More Mobile Escape Analysis for occam-pi

PLAS Research Group Seminar

Martin Ellis
School of Computing, University of Kent, Canterbury

M.C.Ellis@kent.ac.uk

http://www.cs.kent.ac.uk/~me92/

Mobile Escape Analysis

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

Mobility Analysis

- An ID process.

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

$$\textit{mobility}\ \mathrm{ID} = \{\}$$ [1]

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

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

[1] We will come back to this.

Mobility Analysis

- An ID process.

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

$$\textit{mobility}\ \mathrm{ID} = \{\}^1$$

- For an ‘MID’ process that transports/buffers mobiles:

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

[1] We will come back to this.

Mobility Analysis

- An ID process.

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

$$\textit{mobility}\ \mathrm{ID} = \{\}$$[1]

- 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
:
```

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

[1]We will come back to this.

Mobility Analysis

- An ID process.

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

$$\textit{mobility}\ \mathrm{ID} = \{\}$$ [1]

- 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
:
```

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

[1] We will come back to this.

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'
:
```

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

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

- 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
:
```

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

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

- 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
:
```

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

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

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

- 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
:
```

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

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

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

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!)

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
:
```

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

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

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

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!)

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
:
```

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

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

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

$$= \{\langle in?^{u}, out!^{u} \rangle\}$$

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!)

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
:
```

$$mobility\,\mathrm{P} = \{\langle \mathbb{H}?^{x}, out!^{x}\rangle\}$$

$$mobility\,\mathrm{Q} = \{\langle in?^{y}, \mathbb{H}!^{y}\rangle\}$$

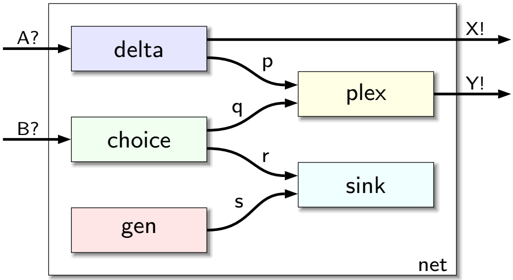
$$mobility\,\mathrm{R} = \{\langle in?^{v}, Lc!^{v}\rangle, \langle Lc?^{w}, out!^{w}\rangle\} \setminus \{Lc\} = \{\langle in?^{u}, out!^{u}\rangle\}$$

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!)

Using Mobile Analysis



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

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

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

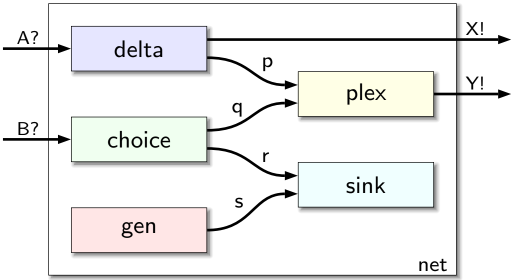
$$mobility\ \mathrm{plex} = \{\langle in0?^a, out!^a\rangle, \langle in1?^b, out!^b\rangle\}$$

$$mobility\ \mathrm{sink} = \{\langle in0?^a\rangle, \langle in1?^b\rangle\}$$

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

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

Using Mobile Analysis



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

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

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

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

$$\textit{mobility}\ \text{sink} = \{\langle in0?^a\rangle, \langle in1?^b\rangle\}$$

- 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

- **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:

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

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

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

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

- 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.

Modelling Non-Mobiles

- All non-mobile occam-pi code can be converted into a pure mobile equivalent.

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

=

Modelling Non-Mobiles

- All non-mobile occam-pi code can be converted into a pure mobile equivalent.

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

=

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

Modelling Non-Mobiles

- All non-mobile occam-pi code can be converted into a pure mobile equivalent.

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

$= \{\}$

=

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

$= \{\langle in?^u, \mathbb{H}!^u \rangle, \langle \mathbb{H}?^z, out!^z \rangle\}$

The Made-From Operator "$\leftarrow$"

- $a \leftarrow \{b, c\}$ means *mobile a* is made from *data a* and *b*.
  - $a \leftarrow \{b\} \equiv a \leftarrow b$
  - $\{\langle out!^a \rangle\} \equiv \{\langle out!^{a \leftarrow \{a\}} \rangle\}$

```
PROC MR1 (CHAN MOBILE THING in?, out!)
  MOBILE THING v, w:
  SEQ
    in ? v
    w := v
    out ! w
:
```

$\textit{mobility}\ \mathrm{MR1} = \{\langle in?^a, out!^a \rangle\}$

$\textit{mobility}\ \mathrm{MR2} = \{\langle in?^a, out!^{b \leftarrow \{a\}} \rangle\}$

$\textit{mobility}\ \mathrm{MR3} = \{\langle in?^a, out!^{a \leftarrow \{a, \tau\}} \rangle\}$

$\textit{mobility}\ \mathrm{MR4} = \{\langle in?^a, out!^{b \leftarrow \{a, \tau\}} \rangle\}$

The Made-From Operator "$\leftarrow$"

- $a \leftarrow \{b, c\}$ means *mobile a* is made from *data a* and *b*.
  - $a \leftarrow \{b\} \equiv a \leftarrow b$
  - $\{\langle out!^{a}\rangle\} \equiv \{\langle out!^{a \leftarrow \{a\}}\rangle\}$

```
PROC MR1 (CHAN MOBILE THING in?, out!)
  MOBILE THING v, w:
  SEQ
    in ? v
    w := v
    out ! w
:
```

$mobility\ \mathrm{MR1} = \{\langle in?^{a}, out!^{a}\rangle\}$

$mobility\ \mathrm{MR2} = \{\langle in?^{a}, out!^{b \leftarrow \{a\}}\rangle\}$

$mobility\ \mathrm{MR3} = \{\langle in?^{a}, out!^{a \leftarrow \{a,\tau\}}\rangle\}$

$mobility\ \mathrm{MR4} = \{\langle in?^{a}, out!^{b \leftarrow \{a,\tau\}}\rangle\}$

The Made-From Operator "$\leftarrow$"

- $a \leftarrow \{b, c\}$ means *mobile* a is made from *data* a and b.
  - $a \leftarrow \{b\} \;\equiv\; a \leftarrow b$
  - $\{\langle out!^{a} \rangle\} \;\equiv\; \{\langle out!^{a \leftarrow \{a\}} \rangle\}$

```
PROC MR1 (CHAN MOBILE THING in?, out!)
  MOBILE THING v, w:
  SEQ
    in ? v
    w := v
    out ! w
:
```

$mobility\ \mathrm{MR1} = \{\langle in?^{a}, out!^{a} \rangle\}$

$mobility\ \mathrm{MR2} = \{\langle in?^{a}, out!^{b \leftarrow \{a\}} \rangle\}$

$mobility\ \mathrm{MR3} = \{\langle in?^{a}, out!^{a \leftarrow \{a, \tau\}} \rangle\}$

$mobility\ \mathrm{MR4} = \{\langle in?^{a}, out!^{b \leftarrow \{a, \tau\}} \rangle\}$

The Made-From Operator "$\leftarrow$"

- $a \leftarrow \{b, c\}$ means *mobile a* is made from *data a* and *b*.
  - $a \leftarrow \{b\} \;\equiv\; a \leftarrow b$
  - $\{\langle out!^{a} \rangle\} \;\equiv\; \{\langle out!^{a \leftarrow \{a\}} \rangle\}$

```
PROC MR1 (CHAN MOBILE THING in?, out!)
  MOBILE THING v, w:
  SEQ
    in ? v
    w := v
    out ! w
:
```

$mobility\ \mathrm{MR1} = \{\langle in?^{a}, out!^{a} \rangle\}$

$mobility\ \mathrm{MR2} = \{\langle in?^{a}, out!^{b \leftarrow \{a\}} \rangle\}$

$mobility\ \mathrm{MR3} = \{\langle in?^{a}, out!^{a \leftarrow \{a, \tau\}} \rangle\}$

$mobility\ \mathrm{MR4} = \{\langle in?^{a}, out!^{b \leftarrow \{a, \tau\}} \rangle\}$

The Made-From Operator "$\leftarrow$"

- $a \leftarrow \{b, c\}$ means *mobile a* is made from *data a* and *b*.
  - $a \leftarrow \{b\} \equiv a \leftarrow b$
  - $\{\langle out!^{a} \rangle\} \equiv \{\langle out!^{a \leftarrow \{a\}} \rangle\}$

```
PROC MR2 (CHAN MOBILE THING in?, out!)
  MOBILE THING v, w:
  SEQ
    in ? v
    w := CLONE v
    out ! w
:
```

$$mobility\ \mathrm{MR1} = \{\langle in?^{a}, out!^{a} \rangle\}$$
$$mobility\ \mathrm{MR2} = \{\langle in?^{a}, out!^{b \leftarrow \{a\}} \rangle\}$$
$$mobility\ \mathrm{MR3} = \{\langle in?^{a}, out!^{a \leftarrow \{a, \tau\}} \rangle\}$$
$$mobility\ \mathrm{MR4} = \{\langle in?^{a}, out!^{b \leftarrow \{a, \tau\}} \rangle\}$$

The Made-From Operator "$\leftarrow$"

- $a \leftarrow \{b, c\}$ means *mobile* a is made from *data* a and b.
  - $a \leftarrow \{b\} \;\equiv\; a \leftarrow b$
  - $\{\langle out!^{a} \rangle\} \;\equiv\; \{\langle out!^{a \leftarrow \{a\}} \rangle\}$

```
PROC MR2 (CHAN MOBILE THING in?, out!)
  MOBILE THING v, w:
  SEQ
    in ? v
    w := CLONE v
    out ! w
:
```

$$\textit{mobility}\ \mathrm{MR1} = \{\langle in?^{a}, out!^{a} \rangle\}$$

$$\textit{mobility}\ \mathrm{MR2} = \{\langle in?^{a}, out!^{b \leftarrow \{a\}} \rangle\}$$

$$\textit{mobility}\ \mathrm{MR3} = \{\langle in?^{a}, out!^{a \leftarrow \{a,\tau\}} \rangle\}$$

$$\textit{mobility}\ \mathrm{MR4} = \{\langle in?^{a}, out!^{b \leftarrow \{a,\tau\}} \rangle\}$$

The Made-From Operator "$\leftarrow$"

- $a \leftarrow \{b, c\}$ means *mobile* a is made from *data* a and b.
  - $a \leftarrow \{b\} \;\equiv\; a \leftarrow b$
  - $\{\langle out!^{a} \rangle\} \;\equiv\; \{\langle out!^{a \leftarrow \{a\}} \rangle\}$

```
PROC MR3 (CHAN MOBILE THING in?, out!)
  MOBILE THING v, w:
  SEQ
    in ? v
    v := v + 1
    out ! v
:
```

$$\mathit{mobility}\ \mathrm{MR1} = \{\langle in?^{a}, out!^{a} \rangle\}$$

$$\mathit{mobility}\ \mathrm{MR2} = \{\langle in?^{a}, out!^{b \leftarrow \{a\}} \rangle\}$$

$$\mathit{mobility}\ \mathrm{MR3} = \{\langle in?^{a}, out!^{a \leftarrow \{a, \tau\}} \rangle\}$$

$$\mathit{mobility}\ \mathrm{MR4} = \{\langle in?^{a}, out!^{b \leftarrow \{a, \tau\}} \rangle\}$$

The Made-From Operator "$\leftarrow$"

- $a \leftarrow \{b, c\}$ means *mobile* a is made from *data* a and b.
  - $a \leftarrow \{b\} \equiv a \leftarrow b$
  - $\{\langle out!^{a} \rangle\} \equiv \{\langle out!^{a \leftarrow \{a\}} \rangle\}$

```
PROC MR3 (CHAN MOBILE THING in?, out!)
  MOBILE THING v, w:
  SEQ
    in ? v
    v := v + 1
    out ! v
:
```

$$mobility\ \mathrm{MR1} = \{\langle in?^{a}, out!^{a} \rangle\}$$

$$mobility\ \mathrm{MR2} = \{\langle in?^{a}, out!^{b \leftarrow \{a\}} \rangle\}$$

$$mobility\ \mathrm{MR3} = \{\langle in?^{a}, out!^{a \leftarrow \{a, \tau\}} \rangle\}$$

$$mobility\ \mathrm{MR4} = \{\langle in?^{a}, out!^{b \leftarrow \{a, \tau\}} \rangle\}$$

The Made-From Operator "$\leftarrow$"

- $a \leftarrow \{b, c\}$ means *mobile* a is made from *data* a and b.
  - $a \leftarrow \{b\} \equiv a \leftarrow b$
  - $\{\langle out!^{a} \rangle\} \equiv \{\langle out!^{a \leftarrow \{a\}} \rangle\}$

```
PROC MR4 (CHAN MOBILE THING in?, out!)
  MOBILE THING v, w:
  SEQ
    in ? v
    w := v + 1
    out ! w
:
```

$$mobility\ \mathrm{MR1} = \{\langle in?^{a}, out!^{a} \rangle\}$$

$$mobility\ \mathrm{MR2} = \{\langle in?^{a}, out!^{b \leftarrow \{a\}} \rangle\}$$

$$mobility\ \mathrm{MR3} = \{\langle in?^{a}, out!^{a \leftarrow \{a, \tau\}} \rangle\}$$

$$mobility\ \mathrm{MR4} = \{\langle in?^{a}, out!^{b \leftarrow \{a, \tau\}} \rangle\}$$

The Made-From Operator "$\leftarrow$"

- $a \leftarrow \{b, c\}$ means *mobile* a is made from *data* a and b.
  - $a \leftarrow \{b\} \;\equiv\; a \leftarrow b$
  - $\{\langle out!^{a} \rangle\} \;\equiv\; \{\langle out!^{a \leftarrow \{a\}} \rangle\}$

```
  PROC MR4 (CHAN MOBILE THING in?, out!)
    MOBILE THING v, w:
    SEQ
      in ? v
      w := v + 1
      out ! w
  :
```

$$\textit{mobility}\ \mathrm{MR1} = \{\langle in?^{a}, out!^{a} \rangle\}$$

$$\textit{mobility}\ \mathrm{MR2} = \{\langle in?^{a}, out!^{b \leftarrow \{a\}} \rangle\}$$

$$\textit{mobility}\ \mathrm{MR3} = \{\langle in?^{a}, out!^{a \leftarrow \{a, \tau\}} \rangle\}$$

$$\textit{mobility}\ \mathrm{MR4} = \{\langle in?^{a}, out!^{b \leftarrow \{a, \tau\}} \rangle\}$$

Sources of data.

External Channel Mobiles read in from external channels contain their own data.

$$PROC\ foo(a, b) = \{\langle a?^{\alpha}, b!^{\alpha \leftarrow \{\alpha\}}\rangle\}$$

From the Heap Mobiles retrieved form the heap are made from undeclared (σ) data. [2]

$$PROC\ bar(a) = \{\langle \mathbb{H}?^{\beta}, a!^{\beta \leftarrow \{\sigma\}}\rangle\}$$

From Internal State Internal state (often constants) is represented as τ.

$$PROC\ foobar(a) = \{\langle \mathbb{H}?^{\delta}, a!^{\delta \leftarrow \{\tau\}}\rangle\}$$

[2] This is not usualy possible in occam due to default initialisers.

Modelling Non-Mobiles

Reviewing our earlier example. . .

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

=

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

$$= \{\}$$

$$= \{ \langle in?^{u}, \mathbb{H}!^{u} \rangle, \langle \mathbb{H}?^{v}, \mathbb{H}!^{v \leftarrow \{u\}} \rangle, \langle \mathbb{H}?^{w}, out!^{w \leftarrow \{v \leftarrow \{u\}\}} \rangle \}$$

Modelling Non-Mobiles

Reviewing our earlier example...

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

$=$

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

$= \{\}$

$$= \{ \langle in?^{u}, \mathbb{H}!^{u} \rangle, \langle \mathbb{H}?^{w}, out!^{w \leftarrow \{u\}} \rangle \}$$

More Examples - Flow Control

```
PROC S (MOBILE CHAN INT in1?, in2?, out!)
  INT v, w:
  SEQ
    in ? v
    in ? w
    IF
      v = 0
        out ! v
      w = 0
        out ! v
      TRUE
        out ! w
:
```

$$\{\langle in1?^{A}, out!^{A\leftarrow\{B,tau\}}\rangle, \langle in2?^{B}, out!^{B\leftarrow\{A,B,tau\}}\rangle,$$
$$\langle in2?^{B}, out!^{B\leftarrow\{A,B,tau\}}\rangle, \langle in1?^{A}, \mathbb{H}!^{A}\rangle, \langle in2?^{B}, \mathbb{H}!^{B}\rangle \quad \}$$

More Examples - Flow Control

```
PROC S (MOBILE CHAN INT in1?, in2?, out!)
  INT v, w:
  SEQ
    in ? v
    in ? w
    IF
      v = 0
        out ! v
      w = 0
        out ! w
      TRUE
        out ! w
:
```

$$\{\langle in1?^{A}, out!^{A \leftarrow \{B,tau\}}\rangle, \langle in2?^{B}, out!^{B \leftarrow \{A,B,tau\}}\rangle,$$
$$\langle in2?^{B}, out!^{B \leftarrow \{A,B,tau\}}\rangle, \langle in1?^{A}, \mathbb{H}!^{A}\rangle, \langle in2?^{B}, \mathbb{H}!^{B}\rangle \quad \}$$

More Examples - Flow Control

```
PROC S (MOBILE CHAN INT in1?, in2?, out!)
  INT v, w:
  SEQ
    in ? v
    in ? w
    IF
      v = 0
        out ! v
      w = 0
        out ! v
      TRUE
        out ! w
:
```

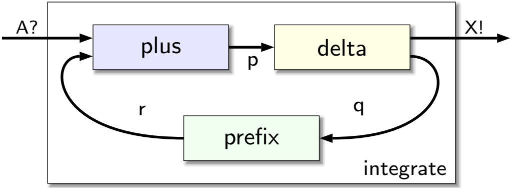
$$\{\langle in1?^{A}, out!^{A \leftarrow \{B,tau\}}\rangle, \langle in2?^{B}, out!^{B \leftarrow \{A,B,tau\}}\rangle,$$
$$\langle in2?^{B}, out!^{B \leftarrow \{A,B,tau\}}\rangle, \langle in1?^{A}, \mathbb{H}!^{A}\rangle, \langle in2?^{B}, \mathbb{H}!^{B}\rangle \quad \}$$

More Examples - Flow Control

```
PROC S (MOBILE CHAN INT in1?, in2?, out!)
  INT v, w:
  SEQ
    in ? v
    in ? w
    IF
      v = 0
        out ! v
      w = 0
        out ! v
      TRUE
        out ! w
:
```

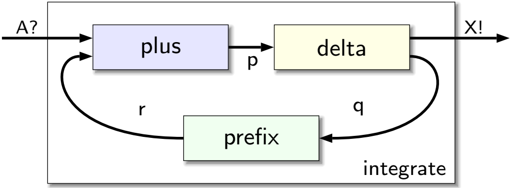
$$\{\langle in1?^{A}, out!^{A\leftarrow\{B,tau\}}\rangle, \langle in2?^{B}, out!^{B\leftarrow\{A,B,tau\}}\rangle,$$
$$\langle in2?^{B}, out!^{B\leftarrow\{A,B,tau\}}\rangle, \langle in1?^{A}, \mathbb{H}!^{A}\rangle, \langle in2?^{B}, \mathbb{H}!^{B}\rangle \quad \}$$

Mobile Refinement



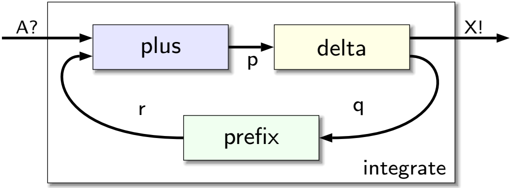
- *mobility* integrate.spec$(in, out) = \{\langle in?^{a}, out!^{a \leftarrow \{a,\tau\}}\rangle\}$
- *mobility* integrate$(in, out) = \{\langle in?^{a}, out!^{b \leftarrow \{a,\tau\}}\rangle\}$
- *mobility* integrate.spec$(in, out) \sqsubseteq_{\mathcal{M}}$ *mobility* integrate(in, out)
- $\{\langle in?^{a}, out!^{a \leftarrow \{a,\tau\}}\rangle\} \sqsubseteq_{\mathcal{M}} \{\langle in?^{a}, out!^{b \leftarrow \{a,\tau\}}\rangle\}$

Mobile Refinement



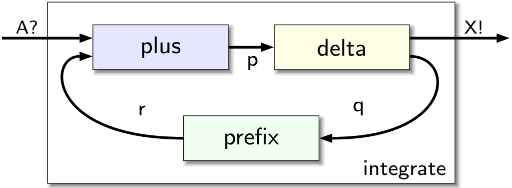
- $\textit{mobility}\ \text{integrate.spec}(\textit{in}, \textit{out}) = \{\langle \textit{in}?^{a}, \textit{out}!^{a \leftarrow \{a,\tau\}} \rangle\}$
- $\textit{mobility}\ \text{integrate}(\textit{in}, \textit{out}) = \{\langle \textit{in}?^{a}, \textit{out}!^{b \leftarrow \{a,\tau\}} \rangle\}$
- $\textit{mobility}\ \text{integrate.spec}(\textit{in}, \textit{out}) \sqsubseteq_{\mathcal{M}} \textit{mobility}\ \text{integrate}(\textit{in}, \textit{out})$
- $\{\langle \textit{in}?^{a}, \textit{out}!^{a \leftarrow \{a,\tau\}} \rangle\} \sqsubseteq_{\mathcal{M}} \{\langle \textit{in}?^{a}, \textit{out}!^{b \leftarrow \{a,\tau\}} \rangle\}$

Mobile Refinement



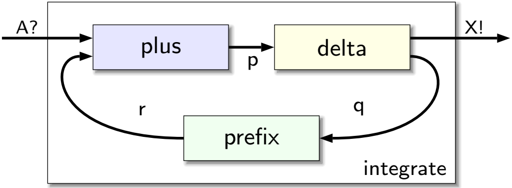
- *mobility* integrate.spec$(in, out) = \{\langle in?^{a}, out!^{a \leftarrow \{a,\tau\}}\rangle\}$
- *mobility* integrate$(in, out) = \{\langle in?^{a}, out!^{b \leftarrow \{a,\tau\}}\rangle\}$
- *mobility* integrate.spec$(in, out) \sqsubseteq_{\mathcal{M}}$ *mobility* integrate(in, out)
- $\{\langle in?^{a}, out!^{a \leftarrow \{a,\tau\}}\rangle\} \sqsubseteq_{\mathcal{M}} \{\langle in?^{a}, out!^{b \leftarrow \{a,\tau\}}\rangle\}$

Mobile Refinement



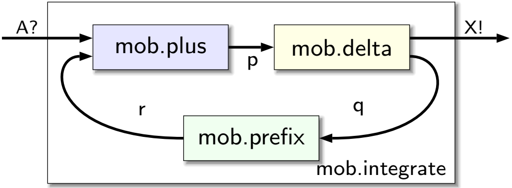
- $\mathit{mobility}$ integrate.spec$(\mathit{in}, \mathit{out}) = \{\langle \mathit{in}?^{a}, \mathit{out}!^{a \leftarrow \{a,\tau\}} \rangle\}$
- $\mathit{mobility}$ integrate$(\mathit{in}, \mathit{out}) = \{\langle \mathit{in}?^{a}, \mathit{out}!^{b \leftarrow \{a,\tau\}} \rangle\}$
- $\mathit{mobility}$ integrate.spec$(\mathit{in}, \mathit{out}) \sqsubseteq_{\mathcal{M}}$ $\mathit{mobility}$ integrate$(\mathit{in}, \mathit{out})$
- $\{\langle \mathit{in}?^{a}, \mathit{out}!^{a \leftarrow \{a,\tau\}} \rangle\} \sqsubseteq_{\mathcal{M}} \{\langle \mathit{in}?^{a}, \mathit{out}!^{b \leftarrow \{a,\tau\}} \rangle\}$

Mobile Refinement



- *mobility* integrate.spec$(in, out) = \{\langle in?^a, out!^{a \leftarrow \{a,\tau\}} \rangle\}$
- *mobility* integrate$(in, out) = \{\langle in?^a, out!^{b \leftarrow \{a,\tau\}} \rangle\}$
- *mobility* integrate.spec$(in, out) \not\sqsubseteq_{\mathcal{M}}$ *mobility* integrate(in, out)
- $\{\langle in?^a, out!^{a \leftarrow \{a,\tau\}} \rangle\} \not\sqsubseteq_{\mathcal{M}} \{\langle in?^a, out!^{b \leftarrow \{a,\tau\}} \rangle\}$

Mobile Refinement

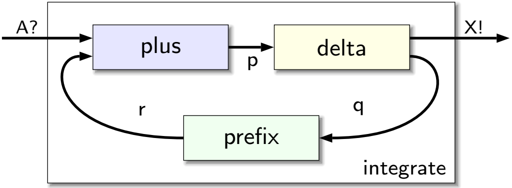


- *mobility* integrate.spec$(in, out) = \{\langle in?^{a}, out!^{a \leftarrow \{a,\tau\}}\rangle\}$
- *mobility* mob.integrate$(in, out) = \{\langle in?^{a}, out!^{a \leftarrow \{a,\tau\}}\rangle\}$
- *mobility* integrate.spec$(in, out) \sqsubseteq_{\mathcal{M}}$ *mobility* mob.integrate(in, out)
- $\{\langle in?^{a}, out!^{a \leftarrow \{a,\tau\}}\rangle\} \sqsubseteq_{\mathcal{M}} \{\langle in?^{a}, out!^{a \leftarrow \{a,\tau\}}\rangle\}$

Data Refinement

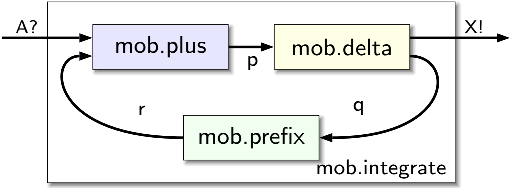
- We can now trace data movement.
- New operator $\sqsubseteq_{\mathcal{D}}$ as data refinement.
- Only concerned about things on the right hand side of the $\leftarrow$ operator.

Data Refinement



- *mobility* $\text{integrate.spec}(\mathit{in}, \mathit{out}) = \{\langle \mathit{in}?^{a}, \mathit{out}!^{a \leftarrow \{a,\tau\}} \rangle\}$
- *mobility* $\text{integrate}(\mathit{in}, \mathit{out}) = \{\langle \mathit{in}?^{a}, \mathit{out}!^{b \leftarrow \{a,\tau\}} \rangle\}$
- *mobility* $\text{integrate.spec}(\mathit{in}, \mathit{out}) \not\sqsubseteq_{\mathcal{M}}$ *mobility* $\text{integrate}(\mathit{in}, \mathit{out})$
- $\{\langle \mathit{in}?^{a}, \mathit{out}!^{a \leftarrow \{a,\tau\}} \rangle\} \not\sqsubseteq_{\mathcal{M}} \{\langle \mathit{in}?^{a}, \mathit{out}!^{b \leftarrow \{a,\tau\}} \rangle\}$
- $\{\langle \mathit{in}?^{a}, \mathit{out}!^{a \leftarrow \{a,\tau\}} \rangle\} \sqsubseteq_{\mathcal{D}} \{\langle \mathit{in}?^{a}, \mathit{out}!^{b \leftarrow \{a,\tau\}} \rangle\}$
- $\{\langle \mathit{in}?^{\{\mathbf{a}\}}, \mathit{out}!^{\{\mathbf{a},\boldsymbol{\tau}\}} \rangle\} \sqsubseteq_{\mathcal{D}} \{\langle \mathit{in}?^{\{\mathbf{a}\}}, \mathit{out}!^{\{\mathbf{a},\boldsymbol{\tau}\}} \rangle\}$

Data Refinement



- $\textit{mobility}\ \mathrm{integrate.spec}(\textit{in}, \textit{out}) = \{\langle \textit{in}?^{a}, \textit{out}!^{a \leftarrow \{a,\tau\}}\rangle\}$
- $\textit{mobility}\ \mathrm{mob.integrate}(\textit{in}, \textit{out}) = \{\langle \textit{in}?^{a}, \textit{out}!^{a \leftarrow \{a,\tau\}}\rangle\}$
- $\textit{mobility}\ \mathrm{integrate.spec}(\textit{in}, \textit{out}) \sqsubseteq_{\mathcal{M}} \textit{mobility}\ \mathrm{mob.integrate}(\textit{in}, \textit{out})$
- $\{\langle \textit{in}?^{a}, \textit{out}!^{a \leftarrow \{a,\tau\}}\rangle\} \sqsubseteq_{\mathcal{M}} \{\langle \textit{in}?^{a}, \textit{out}!^{a \leftarrow \{a,\tau\}}\rangle\}$
- $\{\langle \textit{in}?^{a}, \textit{out}!^{a \leftarrow \{a,\tau\}}\rangle\} \sqsubseteq_{\mathcal{D}} \{\langle \textit{in}?^{a}, \textit{out}!^{a \leftarrow \{a,\tau\}}\rangle\}$
- $\{\langle \textit{in}?^{\{\mathbf{a}\}}, \textit{out}!^{\{\mathbf{a},\boldsymbol{\tau}\}}\rangle\} \sqsubseteq_{\mathcal{D}} \{\langle \textit{in}?^{\{\mathbf{a}\}}, \textit{out}!^{\{\mathbf{a},\boldsymbol{\tau}\}}\rangle\}$

Data Refinement

- Security
  - *mobility* secure.spec(*in*, *out*) =

$$\{\langle \mathit{private.data}?^{a}, \mathit{output}!^{a \leftarrow \{a,\tau\}} \rangle, \langle \mathit{input}?^{b}, \mathbb{H}!^{b} \rangle,$$
$$\langle \mathbb{H}?^{c}, \mathit{private.output}!^{c \leftarrow \{a,b,\tau\}} \rangle\}$$

  - *mobility* secure.spec $\sqsubseteq_{\mathcal{D}}$ *mobility* secure

Minimal Componants

→ BUFFER →	$PROC\ BUFFER(in, out) = \{\langle in?^a, out!^a \rangle\}$
GENERATOR →	$PROC\ GENERATOR(out) = \{\langle \mathbb{H}?^a, out!^{a \leftarrow \{\tau\}} \rangle\}$
→ BLACKHOLE	$PROC\ BLACKHOLE(out) = \{\langle in?^a, \mathbb{H}!^a \rangle\}$
→ → PLUS →	$PROC\ PLUS(in1, in2, out) = \{\langle in1?^a, out!^{a \leftarrow \{a,b\}} \rangle, \langle in2?^b, out!^{a \leftarrow \{a,b\}} \rangle\}$

More Minimal Componants

	$PROC\ DELTA(in, out1, out2) = \{\langle in?^{a}, out1!^{a}\rangle, \langle in?^{a}, out2!^{a}\rangle\}$[3]
	$PROC\ MUX(in1, in2, out) = \{\langle in1?^{a}, out!^{a}\rangle, \langle in2?^{b}, out!^{b}\rangle\}$
	$PROC\ ROUTE(in, out1, out2) = \{\langle in?^{a}, out1!^{a}\rangle, \langle in?^{b}, out2!^{b}\rangle\}$
	$PROC\ REPLACE(in1, in2, out) = \{\langle in1?^{a}, out!^{b\leftarrow\{a\}}\rangle, \langle in2?^{b}, out!^{b\leftarrow\{a\}}\rangle\}$

[3] Though this is not valid in traditional Occam it is possible with shared mobiles.

Conclusions

- Detect mobile escape:
  - For optimisation.
- Detect data escape:
  - For security.
  - For consistancy checking.

Limitations:

- Data escape is a over approximation.
- Structured data is modelled as single item.
- Everything escapes everywhere.

The End

- **Any questions**?

References

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:

$$P \sqsubseteq_M Q \quad \equiv \quad mobility\ Q \subseteq mobility\ P$$

- 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:

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

Denotational Semantics

- Choice:

$$mobility\ (P \Box Q) = (mobility\ P) \cup (mobility\ Q)$$
$$mobility\ (P \sqcap Q) = (mobility\ P) \cup (mobility\ Q)$$

- Interleaving and parallelism:

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

- Hiding:

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