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.

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

\[
\text{mobility ID} = \{\}\textsuperscript{1}
\]

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

\[
\text{mobility MID} = \{in?^a, out!^a\}
\]

\textsuperscript{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
:;
```

$$mobility \text{ID} = \{\}^1$$

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

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

---

$^1$We will come back to this.
Mobility Analysis

- An ID process.

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

\[ mobility \, ID = \{\} \]

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

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

\[ mobility \, MID = \{in^a, out^a\} \]

\(^1\)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
  :
```

\[
\text{mobility ID} = \{\\}
\]

- 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
  :
```

\[
\text{mobility MID} = \{\text{in}^a, \text{out}^a\}
\]

\(^1\)We will come back to this.
Generating Models of occam-$\pi$ 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:

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

\[ mobility P = \{\langle out!x \rangle\} \]

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

- Input, output and assignment are largely straightforward:

```plaintext
PROC Q (CHAN MOBILE THING in?)
  MOBILE THING y:
  SEQ
    in ? y
    ... use ‘y’
  :
```

\[ mobility \ P = \{ \langle out!x \rangle \} \]

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

- Input, output and assignment are largely straightforward:

\[
\text{PROC } Q \text{ (CHAN MOBILE THING } \text{ in?)} \\
\text{MOBILE THING } y: \\
\text{SEQ} \\
\text{in } ? \ y \\
\text{... use `y'} \\
: \\
\]

\[
\text{mobility P} = \{\langle out!x \rangle \} \\
\text{mobility Q} = \{\langle in?y \rangle \} \\
\]

- As are choice (\textit{ALT}, \textit{IF}, \textit{CASE}) and parallelism (\textit{PAR}).
  - simply the \textit{set union} of the different branches.
  - \textit{hiding} is more complex – e.g. as above with \textit{`Lc'}.
  - essentially matching \textit{outputs} with \textit{inputs}, and combining those sequences (potentially expansive!)
Generating Models of occam-π Programs

- Input, output and assignment are largely straightforward:

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

- mobility \( P = \{\langle out!^x \rangle\} \)
- mobility \( Q = \{\langle in?!^y \rangle\} \)

- 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-$\pi$ Programs

- Input, output and assignment are largely straightforward:

\[
\text{PROC } R \ (\text{CHAN MOBILE THING } \text{in?}, \text{ out!}) \\
\text{MOBILE THING } v, w: \\
\text{SEQ} \\
\quad \text{in ? } v \\
\quad \text{w := v} \\
\quad \text{out ! w}
\]

\[
\text{mobility } P = \{\langle \text{out!}^x \rangle \} \\
\text{mobility } Q = \{\langle \text{in}?^y \rangle \} \\
\text{mobility } R = \{\langle \text{in}?^v, Lc!^v \rangle, \\
\quad \langle Lc?^w, \text{out!}^w \rangle \} \setminus \{Lc\}
\]

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

- Input, output and assignment are largely straightforward:

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

```
\textit{mobility} P = \{\langle out!^x \rangle \}
\textit{mobility} Q = \{\langle in?^y \rangle \}
\textit{mobility} R = \{\langle in?v^v, Lc!^v \rangle, \langle Lc?^w, out!^w \rangle \} \setminus \{Lc\}
= \{\langle in?^u, out!^u \rangle \}
```

- As are choice (\textit{ALT}, \textit{IF}, \textit{CASE}) and parallelism (\textit{PAR}).

  - simply the \textit{set union} of the different branches.
  - \textit{hiding} is more complex – e.g. as above with \textit{‘Lc’}.
  - essentially matching outputs with inputs, and combining those sequences (potentially expansive!)
Generating Models of occam-$\pi$ 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 $P = \{\langle out!^x \rangle\}$

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

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

- 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-$\pi$ Programs

- Input, output and assignment are largely straightforward:

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

based on the equivalence:

\[ x := y \equiv \begin{array}{c}
  \text{CHAN INT } c:
  \text{PAR}
  \quad c ! y
  \quad c ? x
\end{array} \]

- mobility $P = \{\langle \text{out!}^x \rangle\}$
- mobility $Q = \{\langle \text{in?}^y \rangle\}$
- mobility $R = \{\langle \text{in?}^v, \text{Lc!}^v \rangle, \langle \text{Lc?}^w, \text{out!}^w \rangle \} \setminus \{\text{Lc}\}$

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!)
Input, output and assignment are largely straightforward:

\[ \text{PROC } R \text{ (CHAN MOBILE THING in?, out!)} \]
\[ \text{MOBILE THING } v, w: \]
\[ \text{SEQ} \]
\[ \text{in } ? v \]
\[ w := v \]
\[ \text{out } ! w \]
\[ : \]

Based on the equivalence:
\[ x := y \equiv \begin{align*}
\text{CHAN INT } c: \\
\text{PAR} \]
\[ c ! y \]
\[ c ? x \]

\[ \text{mobility } P = \{ \langle \text{in}\?^x, \text{out}!^x \rangle \} \]
\[ \text{mobility } Q = \{ \langle \text{in}\?^y, \text{in}\!^y \rangle \} \]
\[ \text{mobility } R = \{ \langle \text{in}\?^v, \text{Lc}!^v \rangle, \langle \text{Lc}\?^w, \text{out}!^w \rangle \} \setminus \{ \text{Lc} \} \]

As are choice (\text{ALT, IF, CASE}) and parallelism (\text{PAR}).

- simply the \text{set union} of the different branches.
- \text{hiding} is more complex – e.g. as above with ‘Lc’.
- essentially matching \text{outputs} with \text{inputs}, and combining those sequences (potentially expansive!)
Using Mobile Analysis

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

\[
\text{mobility net} = \{ \langle A^a, X^a \rangle, \langle A^b, p^b \rangle, \langle B^c, q^c \rangle, \langle B^d, r^d \rangle, \\
\langle s^e \rangle, \langle p^f, Y^f \rangle, \langle q^g, Y^g \rangle, \langle r^h \rangle, \langle s^h \rangle \} \setminus \{p, q, r, s\}
\]
Using Mobile Analysis

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

$$\textit{mobility net} = \{ (A^a, X^a), (A^b, p^b), (B^c, q^c), (B^d, r^d), s^e, p^f, Y^f, q^g, Y^g, r^h, s^h \} \setminus \{ p, q, r, s \}$$
Using Mobile Analysis

- **Hiding** the internal channels gives:

\[\{p\} \mapsto \{\langle A^a, X^a \rangle, \langle A^b, Y^b \rangle, \langle B^c, q^c \rangle, \langle B^d, r^d \rangle, \langle s^e \rangle, \langle q^g, Y^g \rangle, \langle r^h \rangle, \langle s^h \rangle \}\]

\[\{q\} \mapsto \{\langle A^a, X^a \rangle, \langle A^b, Y^b \rangle, \langle B^c, Y^c \rangle, \langle B^d, r^d \rangle, \langle s^e \rangle, \langle r^h \rangle, \langle s^h \rangle \}\]

\[\{r\} \mapsto \{\langle A^a, X^a \rangle, \langle A^b, Y^b \rangle, \langle B^c, Y^c \rangle, \langle B^d \rangle, \langle s^e \rangle, \langle s^h \rangle \}\]

\[\{s\} \mapsto \{\langle A^a, X^a \rangle, \langle A^b, Y^b \rangle, \langle B^c, Y^c \rangle, \langle B^d \rangle \}\]

- 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:

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

  \[ \{q\} \rightarrow \{\langle A?^a, X!^a \rangle, \langle A?^b, Y!^b \rangle, \langle B?^c, Y!^c \rangle, \langle B?^d, r!^d \rangle, \langle s!^e \rangle, \langle r?^h, s?^h \rangle \} \]

  \[ \{r\} \rightarrow \{\langle A?^a, X!^a \rangle, \langle A?^b, Y!^b \rangle, \langle B?^c, Y!^c \rangle, \langle B?^d \rangle, \langle s!^e \rangle, \langle s?^h \rangle \} \]

  \[ \{s\} \rightarrow \{\langle A?^a, X!^a \rangle, \langle A?^b, Y!^b \rangle, \langle B?^c, Y!^c \rangle, \langle B?^d \rangle \} \]

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:

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

\[
\{q\} \rightarrow \{\langle A?^a, X!^a \rangle, \langle A?^b, Y!^b \rangle, \langle B?^c, Y!^c \rangle, \langle B?^d, r!^d \rangle, \langle s!^e \rangle, \langle r?^h \rangle, \langle s?^h \rangle \}
\]

\[
\{r\} \rightarrow \{\langle A?^a, X!^a \rangle, \langle A?^b, Y!^b \rangle, \langle B?^c, Y!^c \rangle, \langle B?^d \rangle, \langle s!^e \rangle, \langle s?^h \rangle \}
\]

\[
\{s\} \rightarrow \{\langle A?^a, X!^a \rangle, \langle A?^b, Y!^b \rangle, \langle B?^c, Y!^c \rangle, \langle B?^d \rangle \}
\]

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.

```plaintext
PROC R (CHAN THING in?, out!)
  THING v, w:
  SEQ
    in ? v
    w := v
    out ! w
  :
```
All non-mobile occam-pi code can be converted into a pure mobile equivalent.
Modelling Non-Mobiles

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

\[
\text{PROC } R \ (\text{CHAN THING } \text{in?}, \text{out!}) \\
\text{THING } v, w: \\
\text{SEQ} \\
\quad \text{in } ? \ v \\
\quad w := v \\
\quad \text{out } ! \ w \\
\] = {} \\

\[
\text{PROC } MR \ (\text{CHAN MOBILE THING } \text{in?}, \text{out!}) \\
\text{MOBILE THING } v, w: \\
\text{SEQ} \\
\quad \text{in } ? \ v \\
\quad w := \text{CLONE } v \\
\quad \text{out } ! \ \text{CLONE } w \\
\] = \{\langle in?^u, H!^u \rangle, \langle H?^z, out!^z \rangle\}
The Made-From Operator “←”

- \( a \leftarrow \{ b, c \} \) means \textit{mobile} \( a \) is made from \textit{data} \( a \) and \( b \).
  - \( a \leftarrow \{ b \} \equiv a \leftarrow b \)
  - \( \{ \langle \text{out}!^a \rangle \} \equiv \{ \langle \text{out}!^a \leftarrow \{ a \} \rangle \} \)

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

\( \text{mobility MR1} = \{ \langle \text{in}\!_a, \text{out}\!^a \rangle \} \)
\( \text{mobility MR2} = \{ \langle \text{in}\!_a, \text{out}!^b \leftarrow \{ a \} \rangle \} \)
\( \text{mobility MR3} = \{ \langle \text{in}\!_a, \text{out}!^a \leftarrow \{ a, \tau \} \rangle \} \)
\( \text{mobility MR4} = \{ \langle \text{in}\!_a, \text{out}!^b \leftarrow \{ a, \tau \} \rangle \} \)
The Made-From Operator \(" \leftarrow \"")

- \( a \leftarrow \{ b, c \} \) means mobile \( a \) is made from data \( a \) and \( b \).
- \( a \leftarrow \{ b \} \equiv a \leftarrow b \)
- \( \{ (\text{out}!^a) \} \equiv \{ (\text{out}!^{a\leftarrow\{a\}}) \} \)

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

\( \text{mobility} \text{ MR1} = \{ (\text{in}^a, \text{out}^a) \} \)

\( \text{mobility} \text{ MR2} = \{ (\text{in}^a, \text{out}!^{b\leftarrow\{a\}}) \} \)

\( \text{mobility} \text{ MR3} = \{ (\text{in}^a, \text{out}!^{a\leftarrow\{a,\tau\}}) \} \)

\( \text{mobility} \text{ MR4} = \{ (\text{in}^a, \text{out}!^{b\leftarrow\{a,\tau\}}) \} \)
The Made-From Operator “←”

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

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

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

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

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

$\text{mobility } MR4 = \{\langle \text{in}^a, \text{out}!^{b \leftarrow \{a, \tau\}} \rangle \}$
The Made-From Operator “←”

- \( a \leftarrow \{b, c\} \) 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
```

- \( \text{mobility } MR1 = \{\langle in?^a, out!^a \rangle\} \)
- \( \text{mobility } MR2 = \{\langle in?^a, out!^{b\leftarrow\{a\}} \rangle\} \)
- \( \text{mobility } MR3 = \{\langle in?^a, out!^{a\leftarrow\{a,\tau\}} \rangle\} \)
- \( \text{mobility } MR4 = \{\langle in?^a, out!^{b\leftarrow\{a,\tau\}} \rangle\} \)
The Made-From Operator "←"

- $a \leftarrow \{b, c\}$ 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
:
```

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

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

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

$mobility\ MR4 = \{\langle in^a, out!^{b\leftarrow\{a,\tau\}} \rangle\}$
The Made-From Operator “←”

- $a \leftarrow \{b, c\}$ 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
```

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

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

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

$mobility$ MR4 = $\{\langle in^a, out!^{b\leftarrow\{a,\tau\}} \rangle\}$
The Made-From Operator "←"

- \( a \leftarrow \{b, c\} \) 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
```

\[ mobility\ MR1 = \{\langle in?^a, out!^a \rangle\} \]
\[ mobility\ MR2 = \{\langle in?^a, out!^{b \leftarrow \{a\}} \rangle\} \]
\[ mobility\ MR3 = \{\langle in?^a, out!^{a \leftarrow \{a, \tau\}} \rangle\} \]
\[ mobility\ MR4 = \{\langle in?^a, out!^{b \leftarrow \{a, \tau\}} \rangle\} \]
The Made-From Operator “←”

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

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

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

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

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

$\text{mobility MR4} = \{\langle\text{in}^a, \text{out}!^b\leftarrow\{a, \tau\}\rangle\}$
The Made-From Operator "←"

- \( a \leftarrow \{ b, c \} \) 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
```

\( mobility \ MR1 = \{\langle in?a, out!a \rangle\} \)
\( mobility \ MR2 = \{\langle in?a, out!b \leftarrow \{a\} \rangle\} \)
\( mobility \ MR3 = \{\langle in?a, out!a \leftarrow \{a, \tau\} \rangle\} \)
\( mobility \ MR4 = \{\langle in?a, out!b \leftarrow \{a, \tau\} \rangle\} \)
The Made-From Operator "←"

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

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

\( \text{mobility MR1} = \{\langle \text{in}^a, \text{out}!^a \rangle\} \)

\( \text{mobility MR2} = \{\langle \text{in}^a, \text{out}!^b \leftarrow \{a\} \rangle\} \)

\( \text{mobility MR3} = \{\langle \text{in}^a, \text{out}!^a \leftarrow \{a, \tau\} \rangle\} \)

\( \text{mobility MR4} = \{\langle \text{in}^a, \text{out}!^b \leftarrow \{a, \tau\} \rangle\} \)
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 from the heap are made from undeclared (\(\sigma\)) data. \(^2\)

\[
PROC \ bar(a) = \{\langle \text{H}?^\beta, a!^\beta \leftarrow \{\sigma}\rangle\}
\]

**From Internal State**  Internal state (often constants) is represented as \(\tau\).

\[
PROC \ foobar(a) = \{\langle \text{H}?^\delta, a!^\delta \leftarrow \{\tau}\rangle\}
\]

\(^2\)This is not usually possible in occam due to default initialisers.
Modelling Non-Mobiles

Reviewing our earlier example...

\[
\begin{align*}
\text{PROC } & \text{R (CHAN THING in?, out!)} \\
& \text{THING v, w:} \\
& \text{SEQ} \\
& \quad \text{in ? v} \\
& \quad w := v \\
& \quad \text{out ! w} \\
& = \{} \\
\end{align*}
\]

\[
\begin{align*}
\text{PROC } & \text{MR (CHAN MOBILE THING in?, out!)} \\
& \text{MOBILE THING v, w:} \\
& \text{SEQ} \\
& \quad \text{in ? v} \\
& \quad w := \text{CLONE v} \\
& \quad \text{out ! CLONE w} \\
& = \{ \langle \text{in}^u, \text{out}!^u \rangle, \\
& \quad \langle \text{in}^v, \text{out}!^v \leftarrow \{u\} \rangle, \\
& \quad \langle \text{in}^w, \text{out}!^w \leftarrow \{v \leftarrow \{u\}\} \rangle \} \\
\end{align*}
\]
Modelling Non-Mobiles

Reviewing our earlier example...

\[
\begin{align*}
\text{PROC } R \ (\text{CHAN THING } \text{in?}, \text{out!}) \\
\text{THING } v, w : \\
\text{SEQ} \\
\quad \text{in } ? \ v \\
\quad w := v \\
\quad \text{out} ! \ w \\
\end{align*}
\]

\[
= \{\} \tag{1}
\]

\[
\begin{align*}
\text{PROC } MR \ (\text{CHAN MOBILE THING } \text{in?}, \text{out!}) \\
\text{MOBILE THING } v, w : \\
\text{SEQ} \\
\quad \text{in } ? \ v \\
\quad w := \text{CLONE } v \\
\quad \text{out} ! \ \text{CLONE } w \\
\end{align*}
\]

\[
= \{\langle \text{in}^?u, \text{out}!u \rangle, \langle \text{in}^?w, \text{out}!w \leftarrow \{u\} \rangle\} \tag{2}
\]
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
  :

{⟨in1?A, out!A←{B,tau}⟩, ⟨in2?B, out!B←{A,B,tau}⟩, 
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
  :

\{ ⟨in1?^A, out!^A{B,tau}⟩, ⟨in2?^B, out!^B{A,B,tau}⟩, \\
Example

More Examples - Flow Control

\[
\begin{array}{l}
\text{PROC } S \ (\text{MOBILE CHAN INT in1?, in2?, out!}) \\
\quad \text{INT } v, w: \\
\quad \text{SEQ} \\
\quad \quad \text{in } ? v \\
\quad \quad \text{in } ? w \\
\quad \quad \text{IF} \\
\quad \quad \quad v = 0 \\
\quad \quad \quad \quad \text{out } ! v \\
\quad \quad \quad w = 0 \\
\quad \quad \quad \quad \text{out } ! v \\
\quad \quad \quad \text{TRUE} \\
\quad \quad \quad \quad \text{out } ! w \\
\quad : \\
\end{array}
\]

\[
\{ \langle \text{in1?}^A, \text{out!}^A \leftarrow \{B, \tau\} \rangle, \langle \text{in2?}^B, \text{out!}^B \leftarrow \{A, B, \tau\} \rangle, \\
\langle \text{in2?}^B, \text{out!}^B \leftarrow \{A, B, \tau\} \rangle, \langle \text{in1?}^A, \text{H!}^A \rangle, \langle \text{in2?}^B, \text{H!}^B \rangle \} 
\]
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
: 

{⟨in1?^A, out!^A←\{B,tau\}⟩, ⟨in2?^B, out!^B←\{A,B,tau\}⟩, 
Mobile Refinement

- $\text{mobility integrate.spec}(\text{in, out}) = \{\langle \text{in}\|^a, \text{out}\mid!^a \leftarrow \{a, \tau\}\rangle\}$
- $\text{mobility integrate}(\text{in, out}) = \{\langle \text{in}\|^a, \text{out}\mid!^b \leftarrow \{a, \tau\}\rangle\}$
- $\text{mobility integrate.spec}(\text{in, out}) \subseteq_M \text{mobility integrate}(\text{in, out})$
- $\{\langle \text{in}\|^a, \text{out}\mid!^a \leftarrow \{a, \tau\}\rangle\} \subseteq_M \{\langle \text{in}\|^a, \text{out}\mid!^b \leftarrow \{a, \tau\}\rangle\}$
**Mobile Refinement**

- \( \text{mobility integrate.spec(in, out)} = \{ \langle \text{in}^a, \text{out}!^a \leftarrow \{a, \tau\} \rangle \} \)
- \( \text{mobility integrate}(in, out) = \{ \langle \text{in}^a, \text{out}!^b \leftarrow \{a, \tau\} \rangle \} \)
- \( \text{mobility integrate.spec(in, out)} \subseteq_M \text{mobility integrate}(in, out) \)
- \( \{ \langle \text{in}^a, \text{out}!^a \leftarrow \{a, \tau\} \rangle \} \subseteq_M \{ \langle \text{in}^a, \text{out}!^b \leftarrow \{a, \tau\} \rangle \} \)
Mobile Refinement

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

\[ \{\langle \text{in}^a, \text{out}!^{a \leftarrow \{a, \tau\}} \rangle\} \sqsubseteq_{\mathcal{M}} \{\langle \text{in}^a, \text{out}!^{b \leftarrow \{a, \tau\}} \rangle\} \]
Mobile Refinement

- \( \text{mobility integrate.spec}(in, out) = \{\langle in^a, out!^{a\leftarrow\{a,\tau\}} \rangle\} \)
- \( \text{mobility integrate}(in, out) = \{\langle in^a, out!^{b\leftarrow\{a,\tau\}} \rangle\} \)
- \( \text{mobility integrate.spec}(in, out) \subseteq M \text{ mobility integrate}(in, out) \)
- \( \{\langle in^a, out!^{a\leftarrow\{a,\tau\}} \rangle\} \subseteq M \{\langle in^a, out!^{b\leftarrow\{a,\tau\}} \rangle\} \)
Mobile Refinement

- \textit{mobility} integrate.spec(\textit{in, out}) = \{\langle \textit{in}^a, \textit{out}^!a \leftarrow \{a, \tau\} \rangle \}
- \textit{mobility} integrate(\textit{in, out}) = \{\langle \textit{in}^a, \textit{out}^!b \leftarrow \{a, \tau\} \rangle \}
- \textit{mobility} integrate.spec(\textit{in, out}) \not\sqsubseteq \mathcal{M} \textit{mobility} integrate(\textit{in, out})
- \{\langle \textit{in}^a, \textit{out}^!a \leftarrow \{a, \tau\} \rangle \} \not\sqsubseteq \mathcal{M} \{\langle \textit{in}^a, \textit{out}^!b \leftarrow \{a, \tau\} \rangle \}
**Mobile Refinement**

- \( mobility \ integrate.spec(in, out) = \{ \langle in^a, out^{a\leftarrow\{a,\tau}\rangle} \} \)
- \( mobility \ mob.integrate(in, out) = \{ \langle in^a, out^{a\leftarrow\{a,\tau}\rangle} \} \)
- \( mobility \ integrate.spec(in, out) \sqsubseteq_M mobility \ mob.integrate(in, out) \)
- \( \{ \langle in^a, out^{a\leftarrow\{a,\tau}\rangle} \} \sqsubseteq_M \{ \langle in^a, out^{a\leftarrow\{a,\tau}\rangle} \} \)
We can now trace data movement.

- New operator $\subseteq_D$ as data refinement.
- Only concerned about things on the right hand side of the $\leftarrow$ operator.
Data Refinement

- \textit{mobility} \ \text{integrate}.\text{spec}(in, out) = \{\langle in^?a, out!a \leftarrow \{a, \tau\} \rangle\}
- \textit{mobility} \ \text{integrate}(in, out) = \{\langle in^?a, out!b \leftarrow \{a, \tau\} \rangle\}
- \textit{mobility} \ \text{integrate}.\text{spec}(in, out) \not\subseteq \mathcal{M} \ \textit{mobility} \ \text{integrate}(in, out)
- \{\langle in^?a, out!a \leftarrow \{a, \tau\} \rangle\} \not\subseteq \mathcal{M} \ \{\langle in^?a, out!b \leftarrow \{a, \tau\} \rangle\}
- \{\langle in^?a, out!a \leftarrow \{a, \tau\} \rangle\} \subseteq \mathcal{D} \ \{\langle in^?a, out!b \leftarrow \{a, \tau\} \rangle\}
- \{\langle in^?\{a\}, out!\{a, \tau\} \rangle\} \subseteq \mathcal{D} \ \{\langle in^?\{a\}, out!\{a, \tau\} \rangle\}
Data Refinement

- \( \text{mobility} \) integrate.spec\((in, out)\) = \{\langle in\?^a, out!^{a\leftarrow\{a,\tau}\rangle}\}\)
- \( \text{mobility} \) mob.integrate\((in, out)\) = \{\langle in\?^a, out!^{a\leftarrow\{a,\tau}\rangle}\}\)
- \( \text{mobility} \) integrate.spec\((in, out)\) \(\sqsubseteq\mathcal{M}\) \( \text{mobility} \) 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}\}\)
- \{\langle in\?^a, out!^{a\leftarrow\{a,\tau}\rangle}\}\) \(\sqsubseteq\mathcal{D}\) \{\langle in\?^a, out!^{a\leftarrow\{a,\tau}\rangle}\}\)
- \{\langle in?\{a\}, out!\{a,\tau\}\rangle\}\) \(\sqsubseteq\mathcal{D}\) \{\langle in?\{a\}, out!\{a,\tau\}\rangle\}\)
Data Refinement

Security

- mobility secure.spec(in, out) =
  \{ ⟨private.data?^a, output!^a←\{a,τ⟩⟩, ⟨input?^b, H!^b⟩, 
  ⟨H?^c, private.output!^c←\{a,b,τ⟩⟩⟩ \}

- mobility secure.spec ⊑_D mobility secure
## Minimal Components

<table>
<thead>
<tr>
<th>Component</th>
<th>Description</th>
<th>Equation</th>
</tr>
</thead>
<tbody>
<tr>
<td>BUFFER</td>
<td>PROC BUFFER((in, out)) = {\langle in^a, out^a \rangle }</td>
<td></td>
</tr>
<tr>
<td>GENERATOR</td>
<td>PROC GENERATOR((out)) = {\langle H^a, out^a \leftarrow {\tau} \rangle }</td>
<td></td>
</tr>
<tr>
<td>BLACKHOLE</td>
<td>PROC BLACKHOLE((out)) = {\langle in^a, \overline{H}^a \rangle }</td>
<td></td>
</tr>
<tr>
<td>PLUS</td>
<td>PROC PLUS((in1, in2, out)) = {\langle in1^a, out^a \leftarrow {a, b} \rangle, \langle in2^b, out^a \leftarrow {a, b} \rangle }</td>
<td></td>
</tr>
</tbody>
</table>
More Minimal Components

<table>
<thead>
<tr>
<th>Component</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DELTA</td>
<td>$\text{PROC DELTA}(\text{in}, \text{out1}, \text{out2}) = {\langle \text{in}^a, \text{out1}^a \rangle, \langle \text{in}^a, \text{out2}^a \rangle}$</td>
</tr>
<tr>
<td>MUX</td>
<td>$\text{PROC MUX}(\text{in1}, \text{in2}, \text{out}) = {\langle \text{in1}^a, \text{out}^a \rangle, \langle \text{in2}^b, \text{out}^b \rangle}$</td>
</tr>
<tr>
<td>ROUTE</td>
<td>$\text{PROC ROUTE}(\text{in}, \text{out1}, \text{out2}) = {\langle \text{in}^a, \text{out1}^a \rangle, \langle \text{in}^b, \text{out2}^b \rangle}$</td>
</tr>
<tr>
<td>REPLACE</td>
<td>$\text{PROC REPLACE}(\text{in1}, \text{in2}, \text{out}) = {\langle \text{in1}^a, \text{out}^b \leftarrow {a} \rangle, \langle \text{in2}^b, \text{out}^b \leftarrow {a} \rangle}$</td>
</tr>
</tbody>
</table>

$^3$ 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 consistency checking.

Limitations:
- Data escape is an overapproximation.
- Structured data is modelled as a single item.
- Everything escapes everywhere.
The End

- Any questions?
References

R. Milner.

*Communicating and Mobile Systems: the Pi-Calculus.*
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:

\[ P \sqsubseteq_M Q \equiv \text{mobility } Q \subseteq \text{mobility } P \]

■ 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.

\[
\{\langle A?a, c!^a \rangle, \langle B?b, c!^b \rangle, \langle c?^f, X!^f \rangle, \langle c?^g, Y!^g \rangle, \langle c?^h, Z!^h \rangle\} \\
\{\langle A?a, X!^a \rangle, \langle A?a, Y!^a \rangle, \langle A?a, Z!^a \rangle, \\
\langle B?b, X!^b \rangle, \langle B?b, Y!^b \rangle, \langle B?b, Z!^b \rangle\}
\]

- 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: $\Sigma_t$, where $t$ is a valid *occam-π* protocol and $t \in T$, where $T$ is the set of valid *occam-π* protocols.  
    - following on: $\Sigma_t = \Sigma^!_t \cup \Sigma^?_t$, and $\forall t : T \cdot \Sigma_t \subseteq \Sigma$.  
  - for *shared* mobiles: $\Sigma_+ = \Sigma^!_+ \cup \Sigma^?_+$.  

- Primitive processes:

  mobility SKIP = $\langle \rangle$

  mobility STOP = $\langle \rangle$

  mobility div = mobility CHAOS =

  $$\{ \langle C^! \rangle \mid C \in \Sigma^! \} \cup \{ \langle D^? \rangle \mid D \in \Sigma^? \} \cup$$

  $$\{ \langle C^? , D^! \rangle \mid \forall t : T \cdot (C, D) \in \Sigma^?_t \times \Sigma^!_t \}$$
Denotational Semantics

- Choice:

\[
\text{mobility } (P \sqcup Q) = (\text{mobility } P) \cup (\text{mobility } Q) \\
\text{mobility } (P \sqcap Q) = (\text{mobility } P) \cup (\text{mobility } Q)
\]

- Interleaving and parallelism:

\[
\text{mobility } (P \parallel Q) = (\text{mobility } P) \cup (\text{mobility } Q)
\]

- Hiding:

\[
\text{mobility } (P \setminus x) = \{ M^\downarrow N[\alpha/\beta] \mid \\
(M^\downarrow \langle x!^{\alpha} \rangle, \langle x?^{\beta} \rangle^\downarrow N) \in \text{mobility } P \times \text{mobility } P \} \cup \\
((\text{mobility } P) - \{ F^\downarrow \langle x!^{\alpha} \rangle \mid F^\downarrow \langle x!^{\alpha} \rangle \in \text{mobility } P \} \\
\cup \{ \langle x?^{\beta} \rangle^\downarrow G \mid \langle x?^{\beta} \rangle^\downarrow G \in \text{mobility } P \}) \cup \\
\{ H \mid (H^\downarrow \langle x!^{\alpha} \rangle) \in \text{mobility } P \land (\langle x?^{\beta} \rangle^\downarrow I) \notin \text{mobility } P \land H \neq \langle \rangle \} \cup \\
\{ J \mid (\langle x?^{\beta} \rangle^\downarrow J) \in \text{mobility } P \land (J^\downarrow \langle x!^{\alpha} \rangle) \notin \text{mobility } P \land J \neq \langle \rangle \}
\]