Static Analysis of Services

Flemming Nielson and Hanne Riis Nielson

Technical University of Denmark

MT-LAB
A VKR CENTRE OF EXCELLENCE

Types At Work, 2009

Static Analysis of Services

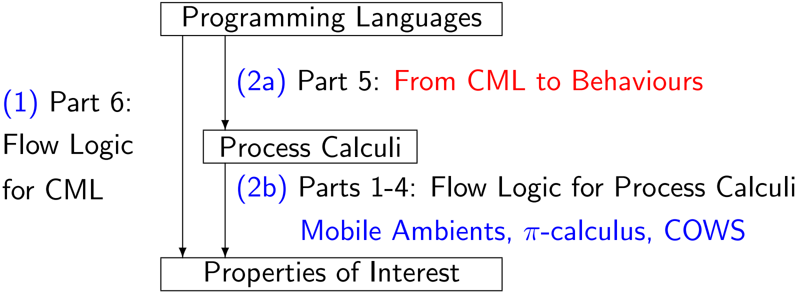
Part 5:

Extracting Process Calculus from Concurrent ML

MT-LAB
A VKR CENTRE OF EXCELLENCE

The Grand View

Recall what we are doing:



Motivation

Concurrent ML

- functions and synchronous operations are first class values
- typed channels and processes are created dynamically

Behaviours

- express the overall communication actions performed
- can serve both as a description of concrete behaviour and as a specification of intended behaviour

Type and Effect Systems

- facilitates extracting behaviours from programs

MT-LAB
A VKR CENTRE OF EXCELLENCE

Syntax of CML subset

Expressions

$$
\begin{array}{lcl}
e & ::= & c \mid x \mid \texttt{fn}\ x\ \texttt{=>}\ e \mid e_1\ e_2 \\
 & \mid & \texttt{let}\ x = e_1\ \texttt{in}\ e_2 \\
 & \mid & \texttt{rec}\ f\ x\ \texttt{=>}\ e \\
 & \mid & \texttt{if}\ e\ \texttt{then}\ e_1\ \texttt{else}\ e_2
\end{array}
$$

Constants:

$$
\begin{array}{lcl}
c & ::= & \texttt{()} \mid \texttt{true} \mid \texttt{false} \mid n \mid \texttt{pair} \mid \texttt{fst} \mid \texttt{snd} \\
 & \mid & \texttt{nil} \mid \texttt{cons} \mid \texttt{hd} \mid \texttt{tl} \mid \texttt{isnil} \mid \texttt{+} \mid \texttt{*} \mid \texttt{=} \mid \cdots \\
 & \mid & \texttt{send} \mid \texttt{receive} \mid \texttt{sync} \\
 & \mid & \texttt{fork}_\pi \mid \texttt{channel}_l \mid \texttt{choose} \mid \texttt{wrap}
\end{array}
$$

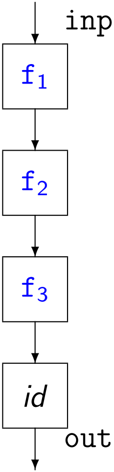
Shorthands:

$$
\begin{array}{lcl}
\texttt{input}\ e & = & \texttt{sync}\ (\texttt{receive}\ e) \\
\texttt{output}\ e & = & \texttt{sync}\ (\texttt{send}\ e)
\end{array}
$$

MT-LAB
A VKR CENTRE OF EXCELLENCE

Example: the pipe function

pipe $[f_1, f_2, f_3]$ inp out



Example: the pipe function

```
let node =
   fn f => fn inp => fn out =>
   fork_π (rec loop d => let v = input inp
                          in output (out, f v);
                             loop d)

in rec pipe fs => fn inp => fn out =>
   if isnil fs
   then node (fn x => x) inp out
   else let ch = channel_r0 ()
        in (node (hd fs) inp ch;
            pipe (tl fs) ch out)
```

MT-LAB
A VKR CENTRE OF EXCELLENCE

The CML primitives

- $\texttt{fork}_\pi$ e spawns a process computing e ()
- $\texttt{channel}_l$ () creates a new channel

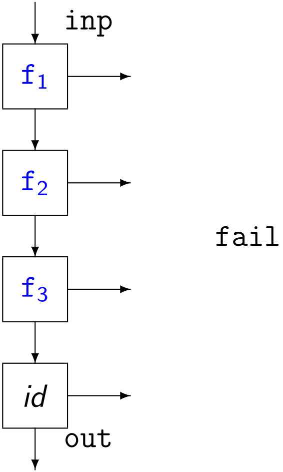
sync, receive and choose do not perform the operations - they produce **delayed** communications (also called events). sync will activate the communications:

- sync (send (ch,v)): sends the value v on the channel ch
- sync (receive ch): receives a value on the channel ch
- sync (choose $[e_1, \cdots, e_n]$): chooses between a list of communication possibilities
- sync (wrap (e_1, e_2)) is "similar" to e_2(sync e_1)

MT-LAB
A VKR CENTRE OF EXCELLENCE

Example: Adding the possibility of failure

`pipe` $[f_1, f_2, f_3]$ `inp out`



Example: Adding the possibility of failure

```
let node =
  fn f => fn inp => fn out =>
  fork_π (rec loop d =>
     sync (choose
              [wrap (receive inp,
                fn x => sync (send (out, f x));
                loop d),
               send(fail,())]))
in rec pipe fs => fn inp => fn out =>
     if isnil fs
     then node (fn x => x) inp out
     else let ch = channel_r0 ()
            in (node (hd fs) inp ch;
                pipe (tl fs) ch out)
```

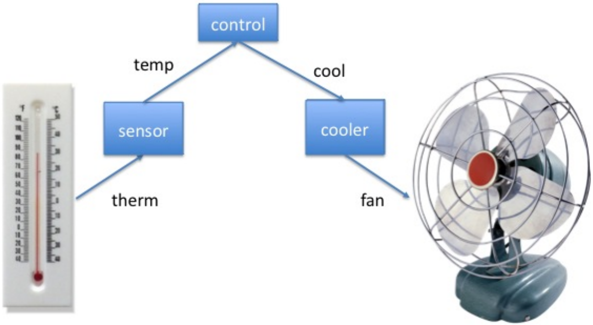
MT-LAB
A VKR CENTRE OF EXCELLENCE

The map function

```
rec map f => fn xs =>
    if isnil xs then nil
    else cons(f (hd xs))(map f (tl xs))
```

Exercise:
A concurrent version?

Controlling a fan



MT-LAB
A VKR CENTRE OF EXCELLENCE

Controlling a fan

In Concurrent ML:

$$
\begin{array}{l}
\texttt{let therm = channel}_{l_t}\ \texttt{();} \\
\quad\ \ \texttt{fan = channel}_{l_f}\ \texttt{();} \\
\quad\ \ \texttt{temp = channel}_{l_1}\ \texttt{();} \\
\quad\ \ \texttt{cool = channel}_{l_2}\ \texttt{()} \\
\texttt{in fork}_{\pi_1}\ \texttt{(... sensor ...);} \\
\quad\ \texttt{fork}_{\pi_2}\ \texttt{(... control ...);} \\
\quad\ \texttt{fork}_{\pi_3}\ \texttt{(... cooling ...)}
\end{array}
$$

MT-LAB
A VKR CENTRE OF EXCELLENCE

Controlling a fan

```
rec sensor (t1,t2) =>
    sync (choose [wrap (receive therm,
                        fn t => sensor(t2,t)),
                  wrap (send (temp, (t1+t2)/2),
                        fn d => sensor (t1,t2))])
```

Controlling a fan

```
rec sensor (t1,t2) =>
    sync (choose [wrap (receive therm,
                        fn t => sensor(t2,t)),
                  wrap (send (temp, (t1+t2)/2),
                        fn d => sensor (t1,t2))])
rec control d =>
    let t = sync (receive temp)
    in ((if t > upper then sync (send(cool,‘‘On’’))
        else if t < lower then sync (send(cool,‘‘Off’’))
             else ‘‘Ok’’); control d)
```

MT-LAB
A VKR CENTRE OF EXCELLENCE

Controlling a fan

```
rec sensor (t1,t2) =>
    sync (choose [wrap (receive therm,
                         fn t => sensor(t2,t)),
                  wrap (send (temp, (t1+t2)/2),
                         fn d => sensor (t1,t2))])
rec control d =>
    let t = sync (receive temp)
    in ((if t > upper then sync (send(cool,``On''))
        else if t < lower then sync (send(cool,``Off''))
             else ``Ok''); control d)
rec cooling state =>
    let new = sync (receive cool)
    in if new = state then cooling state
       else (sync (send(fan, new)); cooling new)
```

MT-LAB
A VKR CENTRE OF EXCELLENCE

Typing system

Behaviours:

$$\begin{array}{lll} b & ::= & \epsilon \mid r!t \mid r?t \mid t\ \mathsf{CHAN}_r \\ & \mid & \mathsf{FORK}_\pi\ b \mid b_1;b_2 \mid b_1+b_2 \\ & \mid & \mathsf{REC}\beta.b \mid \beta \end{array}$$

Types:

$$\begin{array}{lll} t & ::= & \texttt{unit} \mid \texttt{bool} \mid \texttt{int} \mid \alpha \\ & \mid & t_1 \rightarrow^{b} t_2 \mid t_1 \times t_2 \mid t\ \texttt{list} \\ & \mid & t\ \texttt{chan}\ r \mid t\ \texttt{com}\ b \end{array}$$

Regions:

$$\begin{array}{lll} r & ::= & l \mid r_1 + r_2 \qquad \text{(sets of labels)} \\ & \mid & \rho \end{array}$$

Type schemes:

$$ts \ ::= \ t \mid \forall\beta.ts \mid \forall\alpha.ts \mid \forall\rho.ts$$

MT-LAB
A VKR CENTRE OF EXCELLENCE

Example: the pipe function (again)

```
let node =
   fn f => fn inp => fn out =>
   fork_π (rec loop d => let v = input inp
                         in output (out, f v);
                            loop d)

in rec pipe fs => fn inp => fn out =>
   if isnil fs
   then node (fn x => x) inp out
   else let ch = channel_r0 ()
        in (node (hd fs) inp ch;
            pipe (tl fs) ch out)
```

Example: behaviour for node

Assumptions:

$$\begin{array}{rcl} \texttt{f} & : & t_1 \rightarrow^{b} t_2 \\ \texttt{inp} & : & t_1\ \texttt{chan}\ r_1 \\ \texttt{out} & : & t_2\ \texttt{chan}\ r_2 \end{array}$$

$\texttt{node f inp out}$: $\mathsf{FORK}_{\pi}\ (\mathsf{REC}\ \beta.\,(r_1?t_1;\ b;\ r_2!t_2;\ \beta))$

Fork a process that will
- read a value of type t_1 on a channel in r_1
- do the computation f with behaviour b
- write a value of type t_2 on a channel in r_2, and
- recurse

Example: behaviour for pipe

Assumptions:

$$
\begin{array}{lcl}
\texttt{fs} & : & (t \rightarrow^{b} t)\ \texttt{list} \\
\texttt{inp} & : & t\ \texttt{chan}\ r_1 \\
\texttt{out} & : & t\ \texttt{chan}\ r_2
\end{array}
$$

`pipe fs in out`:

$$
\begin{array}{l}
\text{REC}\beta'.\ (\text{FORK}_\pi\ (\text{REC}\ \beta.((r_1 + r_0)?t;\ \epsilon;\ (r_2 + r_0)!t;\ \beta)) \\
\quad + \quad t\ \text{CHAN}_{r_0}; \\
\qquad\quad \text{FORK}_\pi\ (\text{REC}\ \beta.((r_1 + r_0)?t;\ b;\ (r_2 + r_0)!t;\ \beta)); \\
\qquad\quad \beta')
\end{array}
$$

- fork a process that ... or
- allocate a new channel in r_0,
- fork a process that ... and
- recurse

MT-LAB
A VKR CENTRE OF EXCELLENCE

Example: Adding failure (again)

```
let node =
  fn f => fn inp => fn out =>
  fork_π (rec loop d =>
     sync (choose
              [wrap (receive inp,
                 fn x => sync (send (out, f x));
                 loop d),
               send(fail,())]))
in rec pipe fs => fn inp => fn out =>
     if isnil fs
     then node (fn x => x) inp out
     else let ch = channel_r0 ()
          in (node (hd fs) inp ch;
              pipe (tl fs) ch out)
```

MT-LAB
A VKR CENTRE OF EXCELLENCE

Example: behaviour for node

Assumptions:

$$\begin{array}{rcl} \texttt{f} & : & t_1 \rightarrow^{b} t_2 \\ \texttt{fail} & : & \texttt{unit chan } r_0 \\ \texttt{inp} & : & t_1 \texttt{ chan } r_1 \\ \texttt{out} & : & t_2 \texttt{ chan } r_2 \end{array}$$

`node f inp out`:

$$\mathsf{FORK}_{\pi}\ (\mathsf{REC}\ \beta.\,(r_1?t_1;\ b;\ r_2!t_2;\ \beta + r_0!\mathsf{unit}))$$

Fork a process that will
- read a value of type t_1 on a channel in r_1
- do the computation `f` with behaviour b
- write a value of type t_2 on a channel in r_2, and
- recurse

or

- write a value of type `unit` on a channel in r_0, and
- terminate

Example: behaviour for pipe

Assumptions:

`fs`	:	$(t \to^{b} t)$ `list`
`fail`	:	`unit chan` r_0
`inp`	:	t `chan` r_1
`out`	:	t `chan` r_2

`pipe fs in out`:

$$\mathsf{REC}\beta'.\ (\mathsf{FORK}_\pi\ (\mathsf{REC}\ \beta.((r_1 + r_0)?t;\ \epsilon;\ (r_2 + r_0)!t;\ \beta + r_0!\mathsf{unit}))$$
$$+ \quad t\ \mathsf{CHAN}_{r_0};$$
$$\mathsf{FORK}_\pi\ (\mathsf{REC}\ \beta.((r_1 + r_0)?t;\ b;\ (r_2 + r_0)!t;\ \beta\ + r_0!\mathsf{unit}));$$
$$\beta')$$

- fork a process that ... or
- allocate a new channel in r_0,
- fork a process that ... and
- recurse

MT-LAB A VKR CENTRE OF EXCELLENCE

Exercises

Determine the behaviour of the concurrent map functions

Types for constants: TypeOf(c)

send	$\forall\alpha,\rho.\ (\alpha\ \mathtt{chan}\ \rho)\times\alpha \rightarrow^{\epsilon} (\alpha\ \mathtt{com}\ \rho!\alpha)$
receive	$\forall\alpha,\rho.\ (\alpha\ \mathtt{chan}\ \rho) \rightarrow^{\epsilon} (\alpha\ \mathtt{com}\ \rho?\alpha)$
sync	$\forall\alpha,\beta.\ (\alpha\ \mathtt{com}\ \beta) \rightarrow^{\beta} \alpha$
$\mathtt{fork}_\pi$	$\forall\alpha,\beta.\ (\mathtt{unit} \rightarrow^{\beta} \alpha) \rightarrow^{\mathsf{FORK}_\pi\ \beta} \mathtt{unit}$
$\mathtt{channel}_l$	$\forall\alpha.\ \mathtt{unit} \rightarrow^{\alpha\ \mathsf{CHAN}_l} (\alpha\ \mathtt{chan}\ l)$
choose	$\forall\alpha,\beta.\ (\alpha\ \mathtt{com}\ \beta)\ \mathtt{list} \rightarrow^{\epsilon} (\alpha\ \mathtt{com}\ \beta)$
wrap	$\forall\alpha_1,\alpha_2,\beta_1,\beta_2.(\alpha_1\ \mathtt{com}\ \beta_1)\times(\alpha_1 \rightarrow^{\beta_2} \alpha_2) \rightarrow^{\epsilon} (\alpha_2\ \mathtt{com}\ \beta_1;\beta_2)$
input	$\forall\alpha,\rho.\ (\alpha\ \mathtt{chan}\ \rho) \rightarrow^{\rho?\alpha} \alpha$
output	$\forall\alpha,\rho.\ (\alpha\ \mathtt{chan}\ \rho)\times\alpha \rightarrow^{\rho!\alpha} \alpha$

type environment $\vdash e$: type & behaviour

$$tenv \vdash c : t \,\&\, \epsilon \qquad \text{if TypeOf(c)} \succ t$$

$$tenv[x \mapsto ts] \vdash x : t \,\&\, \epsilon \qquad \text{if } ts \succ t$$

$$\frac{tenv[x \mapsto t] \vdash e : t' \,\&\, b}{tenv \vdash \texttt{fn}\ x \ \Rightarrow\ e : t \rightarrow^{b} t' \,\&\, \epsilon}$$

$$\frac{tenv \vdash e_1 : t \rightarrow^{b} t' \,\&\, b_1 \qquad tenv \vdash e_2 : t \,\&\, b_2}{tenv \vdash e_1\ e_2 : t' \,\&\, b_1; b_2; b}$$

$$\frac{tenv \vdash e_1 : t_1 \,\&\, b_1 \qquad tenv[x \mapsto ts] \vdash e_2 : t_2 \,\&\, b_2}{tenv \vdash \texttt{let}\ x{=}e_1\ \texttt{in}\ e_2 : t_2 \,\&\, b_1; b_2}$$

$$\text{if } ts = \text{gen}(tenv, b_1)t_1$$

Exercise:
Add the inference rules for recursion and conditions.

MT-LAB
A VKR CENTRE OF EXCELLENCE

A design decision

- Early subsumption – or subtyping

  Coercions between types can happen at any time inside any type

  - add the subtyping rule:

$$\frac{tenv \vdash e : t \ \& \ b}{tenv \vdash e : t' \ \& \ b'} \text{ if } t \sqsubseteq t' \text{ and } b \sqsubseteq b'$$

- Late subsumption – or subeffecting

  Generic instantion produce the required instances.

  - In TypeOf(c) we shall use **constrained type schemas** as $\forall \beta : t_1 \ \rightarrow^{\beta} \ t_2 \ [\beta \geq b]$ in stead of $t_1 \ \rightarrow^{b} \ t_2$
  - add the subeffect rule:

$$\frac{tenv \vdash e : t \ \& \ b}{tenv \vdash e : t \ \& \ b'} \text{ if } b \sqsubseteq b'$$

MT-LAB
A VKR CENTRE OF EXCELLENCE

Constraint type schemas: TypeOf(c)

`send`	$\forall \alpha, \rho, \beta_1, \beta_2.\ (\alpha\ \mathtt{chan}\ \rho) \times \alpha \rightarrow^{\beta_1} (\alpha\ \mathtt{com}\ \beta_2)[\epsilon \leq \beta_1, \rho!\alpha \leq \beta_2]$
`receive`	$\forall \alpha, \rho, \beta_1, \beta_2.\ (\alpha\ \mathtt{chan}\ \rho) \rightarrow^{\beta_1} (\alpha\ \mathtt{com}\ \beta_2)[\epsilon \leq \beta_1, \rho?\alpha \leq \beta_2]$
`sync`	$\forall \alpha, \beta_1, \beta_2.\ (\alpha\ \mathtt{com}\ \beta_1) \rightarrow^{\beta_2} \alpha[\beta_1 \leq \beta_2]$
$\mathtt{fork}_\pi$	$\forall \alpha, \beta_1, \beta_2.\ (\mathtt{unit} \rightarrow^{\beta_1} \alpha) \rightarrow^{\beta_2} \mathtt{unit}[\mathsf{FORK}_\pi\ \beta_1 \leq \beta_2]$
$\mathtt{channel}_l$	$\forall \alpha, \beta, \rho.\ \mathtt{unit} \rightarrow^{\beta} (\alpha\ \mathtt{chan}\ l)[\alpha\ \mathsf{CHAN}_\rho \leq \beta, l \leq \rho]$
`choose`	$\forall \alpha, \beta_1, \beta_2, \beta_3.\ (\alpha\ \mathtt{com}\ \beta_1)\ \mathtt{list} \rightarrow^{\beta_2} (\alpha\ \mathtt{com}\ \beta_3)[\epsilon \leq \beta_2, \beta_1 \leq \beta_3]$
`wrap`	$\forall \alpha_1, \alpha_2, \beta_1, \beta_2, \beta_3, \beta_4.\ (\alpha_1\ \mathtt{com}\ \beta_1) \times (\alpha_1 \rightarrow^{\beta_2} \alpha_2) \rightarrow^{\beta_3} (\alpha_2\ \mathtt{com}\ \beta_4)[\epsilon \leq \beta_3, \beta_1; \beta_2 \leq \beta_4]$

MT-LAB
A VKR CENTRE OF EXCELLENCE

Generalisation and instantiation

Generalisation

$$\mathrm{gen}(tenv, b)t = \begin{array}{l} \text{let } \bar{\alpha}\bar{\beta}\bar{\rho} = FV(t) \setminus (FV(tenv) \cup FV(b)) \\ \text{in } \forall \bar{\alpha}\bar{\beta}\bar{\rho}.t \end{array}$$

Generalisation and instantiation

Generalisation

$$\mathrm{gen}(tenv, b)t = \text{ let } \bar{\alpha}\bar{\beta}\bar{\rho} = FV(t) \setminus (FV(tenv) \cup FV(b)) \text{ in } \forall \bar{\alpha}\bar{\beta}\bar{\rho}.t$$

Instantiation

$$\forall \bar{\alpha}\bar{\beta}\bar{\rho}.t[C] \succ t'$$

where C is a (possible empty) set of constraints of the form $\beta \geq b$ and $\rho \geq r$

There exists a substitution θ with $DOM(\theta) = \{\bar{\alpha}\bar{\beta}\bar{\rho}\}$ such that

- $\theta t = t'$ and
- all the constraints of C are satisfied, that is,
  - $\theta\beta \sqsupseteq \theta b$ for all $\beta \geq b$ in C, and
  - $\theta\rho \supseteq \theta r$ for all $\rho \geq r$ in C.

Ordering on behaviours

Axioms and rules that formally state:

- $\sqsubseteq$ is a pre-order
- $\sqsubseteq$ is a pre-congruence
- sequencing $;$ is associative
- sequencing distributes over join

$$(b_1 + b_2); b_3 \;\sqsubseteq\; b_1; b_3 + b_2; b_3$$

$$b_1; b_3 + b_2; b_3 \;\sqsubseteq\; (b_1 + b_2); b_3$$

- ϵ is left and right identity for sequencing
- join $+$ is least upper bound operation
- recursion can be unfolded

$$\mathrm{REC}\ \beta.b \;\sqsubseteq\; b[\beta \mapsto \mathrm{REC}\ \beta.b]$$

$$b[\beta \mapsto \mathrm{REC}\ \beta.b] \;\sqsubseteq\; \mathrm{REC}\ \beta.b$$

MT-LAB
A VKR CENTRE OF EXCELLENCE

Example (1)

```
let node = fn f => fn inp => fn out =>
           forkπ ((rec loop d => let v = input inp
                                  in output (out, f v); loop d) ())
in ···
```

Type for node:

$$\forall\, \alpha_1, \alpha_2, \beta, \rho_1, \rho_2.\ \underbrace{(\alpha_1 \xrightarrow{\beta} \alpha_2)}_{\texttt{f}} \xrightarrow{\epsilon} \underbrace{(\alpha_1\ \texttt{chan}\ \rho_1)}_{\texttt{inp}} \xrightarrow{\epsilon} \underbrace{(\alpha_2\ \texttt{chan}\ \rho_2)}_{\texttt{out}} \xrightarrow{\varphi} \texttt{unit}$$

where $\varphi = \mathsf{FORK}_\pi(\mathsf{REC}\ \beta'.\ (\rho_1?\alpha_1;\ \beta;\ \rho_2!\alpha_2;\ \beta'))$

MT-LAB
A VKR CENTRE OF EXCELLENCE

Example (2)

```
let node = ···
in rec pipe fs => fn inp => fn out =>
        if isnil fs then node (fn x => x) inp out
        else let ch = channel_C ()
             in (node (hd fs) inp ch; pipe (tl fs) ch out)
```

Type for pipe:

$$\forall\, \alpha, \beta, \rho_1, \rho_2.\ \underbrace{((\alpha \xrightarrow{\beta} \alpha)\ \texttt{list})}_{\texttt{fs}} \xrightarrow{\epsilon} \underbrace{(\alpha\ \texttt{chan}\ (\rho_1 \cup \{C\}))}_{\texttt{inp, ch}} \xrightarrow{\epsilon} \underbrace{(\alpha\ \texttt{chan}\ \rho_2)}_{\texttt{out}} \xrightarrow{\varphi} \texttt{unit}$$

where $\varphi =$

$$\mathsf{REC}\ \beta'.\ (\underbrace{\mathsf{FORK}_\pi(\mathsf{REC}\ \beta''.((\rho_1 \cup \{C\})?\alpha;\ \epsilon;\ \rho_2!\alpha;\ \beta''))}_{\texttt{node (fn x => x) ...}}$$

$$+\ \alpha\ \mathsf{CHAN}\ C;\ \underbrace{\mathsf{FORK}_\pi(\mathsf{REC}\ \beta''.\ ((\rho_1 \cup \{C\})?\alpha;\ \beta;\ C!\alpha; \beta''))}_{\texttt{node (hd fs) ...}};\ \beta')$$

MT-LAB
A VKR CENTRE OF EXCELLENCE

Exercises

Determine the type and behaviour of the concurrent map functions and the fan controller.

MT-LAB
A VKR CENTRE OF EXCELLENCE

Theory: overview

Structural operational semantics for

- CML
- behaviours

Subject Reduction Theorem:

- types are preserved by CML-evaluation steps
- steps in CML-semantics can be mimicked in behaviour semantics

The developments

- simplified semantics: only input and output
- complex semantics: also `sync`, `receive` and `send`

MT-LAB
A VKR CENTRE OF EXCELLENCE

Sequential Semantics of CML

Evaluation in context:

$$(\texttt{fn}\ x\ \texttt{=>}\ e)\ v \longrightarrow e[x \mapsto v]$$

$$\texttt{let}\ x\ \texttt{=}\ v\ \texttt{in}\ e \longrightarrow e[x \mapsto v]$$

$$\texttt{rec}\ f\ x\ \texttt{=>}\ e \longrightarrow (\texttt{fn}\ x\ \texttt{=>}\ e)[f \mapsto (\texttt{rec}\ f\ x\ \texttt{=>}\ e)]$$

$$\texttt{if true then}\ e_1\ \texttt{else}\ e_2 \longrightarrow e_1$$

$$\texttt{if false then}\ e_1\ \texttt{else}\ e_2 \longrightarrow e_2$$

$$v_1\ v_2 \longrightarrow v_3 \qquad \text{if}(v_1, v_2, v_3) \in \delta$$

MT-LAB A VKR CENTRE OF EXCELLENCE

Sequential Semantics of CML

Evaluation in context:

$$(\texttt{fn}\ x\ \texttt{=>}\ e)\ v \longrightarrow e[x \mapsto v]$$

$$\texttt{let}\ x\ \texttt{=}\ v\ \texttt{in}\ e \longrightarrow e[x \mapsto v]$$

$$\texttt{rec}\ f\ x\ \texttt{=>}\ e \longrightarrow (\texttt{fn}\ x\ \texttt{=>}\ e)[f \mapsto (\texttt{rec}\ f\ x\ \texttt{=>}\ e)]$$

$$\texttt{if}\ \texttt{true}\ \texttt{then}\ e_1\ \texttt{else}\ e_2 \longrightarrow e_1$$

$$\texttt{if}\ \texttt{false}\ \texttt{then}\ e_1\ \texttt{else}\ e_2 \longrightarrow e_2$$

$$v_1\ v_2 \longrightarrow v_3 \qquad \text{if}(v_1, v_2, v_3) \in \delta$$

Evaluation contexts:

$$E ::= [\,] \mid E\ e \mid v\ E \mid \texttt{let}\ x\ \texttt{=}\ E\ \texttt{in}\ e \mid \texttt{if}\ E\ \texttt{then}\ e_1\ \texttt{else}\ e_2$$

$$v ::= c' \mid x \mid \texttt{fn}\ x \Rightarrow e \mid (c', v_1) \mid \cdots \mid (c', v_1, \ldots, v_k)$$

where c' can be any constant except `sync`, `channel` and `fork`.

MT-LAB
A VKR CENTRE OF EXCELLENCE

Concurrent Semantics of CML

$$CP, PP[pi \mapsto E[e_1]] \Rightarrow_{pi}^{\epsilon} CP, PP[pi \mapsto E[e_2]]$$
$$\text{if } e_1 \longrightarrow e_2$$

$$CP, PP[pi \mapsto E[\texttt{channel}_l()]] \Rightarrow_{pi}^{\mathsf{CHAN}_l ci} CP \cup \{ci\}, PP[pi \mapsto E[ci]]$$
$$\text{if } ci \notin CP$$

$$CP, PP[pi \mapsto E[\texttt{fork}_\pi\ e_0]] \Rightarrow_{pi}^{\mathsf{FORK}_\pi pi_0} CP, PP[pi \mapsto E[()]][pi_0 \mapsto e_0\ ()]$$
$$\text{if } pi_0 \notin dom(PP) \cup \{pi\}$$

Annotation of $\Rightarrow_{pi}^{ev}$:

$$ev ::= \epsilon \mid (ci!, ci?) \mid \mathsf{CHAN}_r\ ci \mid \mathsf{FORK}_\pi\ pi$$

MT-LAB
A VKR CENTRE OF EXCELLENCE

Semantics of communication

The simple case:

$$CP, PP[p_1 : E_1[\texttt{output(ci, v)}]][p_2 : E_2[\texttt{input}\ ci]]$$
$$\Rightarrow^{(ci!,ci?)}_{p_1,p_2}\ CP, PP[p_1 : E_1[v]][p_2 : E_2[v]]$$
$$\text{if } p_1 \neq p_2$$

MT-LAB
A VKR CENTRE OF EXCELLENCE

Semantics of communication

The simple case:

$$CP, PP[p_1 : E_1[\mathtt{output(ci,\ v)}]][p_2 : E_2[\mathtt{input}\ ci]]$$
$$\Rightarrow^{(ci!,ci?)}_{p_1,p_2}\ CP, PP[p_1 : E_1[v]][p_2 : E_2[v]]$$
$$\text{if } p_1 \neq p_2$$

More generally:

$$\frac{(v_1, v_2) \overset{(ci!,ci?)}{\rightsquigarrow} (e_1, e_2)}{[p_1 \mapsto E_1[\mathtt{sync}\ v_1], p_2 \mapsto E_2[\mathtt{sync}\ v_2]]}$$
$$\Rightarrow^{(ci!,ci?)}_{p_1,p_2}\ [p_1 \mapsto E_1[e_1], p_2 \mapsto E_2[e_2]]$$
$$\text{if } p_1 \neq p_2$$

MT-LAB
A VKR CENTRE OF EXCELLENCE

Matching

$$((\texttt{send}(ci, v)), (\texttt{receive}\ ci)) \overset{(ci!, ci?)}{\rightsquigarrow} (v, v)$$

$$\frac{(v_1, v_3) \overset{(d_1, d_2)}{\rightsquigarrow} (e_1, e_2)}{(\texttt{wrap}(v_1, v_2), v_3) \overset{(d_1, d_2)}{\rightsquigarrow} (v_2\ e_1, e_2)}$$

plus rules for `choose` and swopping input and output

MT-LAB
A VKR CENTRE OF EXCELLENCE

Subject reduction theorem

$$CP, PP[pi \mapsto e] \longrightarrow^{ev}_{ps} CP', PP'[pi \mapsto e']$$

$$[\,] \vdash e : t \;\&\; b$$

$$PB[pi \mapsto b]$$

Subject reduction theorem

$$CP, PP[pi \mapsto e] \longrightarrow^{ev}_{ps} CP', PP'[pi \mapsto e']$$

$$[\,] \vdash e : t \;\&\; b \qquad\qquad [\,] \vdash e' : t \;\&\; b'$$

$$PB[pi \mapsto b]$$

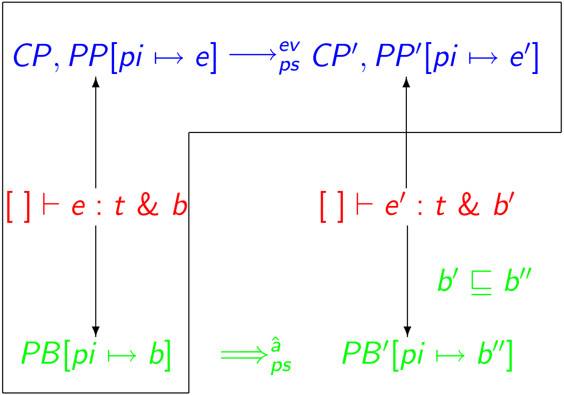
Subject reduction theorem

$$CP, PP[pi \mapsto e] \longrightarrow^{ev}_{ps} CP', PP'[pi \mapsto e']$$

$$[\,] \vdash e : t \;\&\; b \qquad\qquad [\,] \vdash e' : t \;\&\; b'$$

$$PB[pi \mapsto b] \quad \Longrightarrow^{\hat{a}}_{ps} \quad PB'[pi \mapsto b'']$$

Subject reduction theorem



where a is 'the translation of' ev

Semantics of behaviours: sequential

$p \Rightarrow^{p} \epsilon$ $\qquad$ $\epsilon \Rightarrow^{\epsilon} \surd$

$b \Rightarrow^{\epsilon} b$

$$\frac{b_1 \Rightarrow^{p} b_1'}{b_1; b_2 \Rightarrow^{p} b_1'; b_2} \qquad \frac{b_1 \Rightarrow^{p} \surd}{b_1; b_2 \Rightarrow^{p} b_2}$$

plus rules for choice and recursion

Annotations of $\Rightarrow^{p}$:

$$p \quad ::= \quad \epsilon \mid r!t \mid r?t \mid t\ \mathsf{CHAN}_r \mid \mathsf{FORK}_\pi\ b$$

MT-LAB
A VKR CENTRE OF EXCELLENCE

Semantics of behaviours: concurrent

$$\frac{b \Rightarrow^{a} b'}{PB[pi \mapsto b] \Longrightarrow^{a}_{pi} PB[pi \mapsto b']}$$

$$\frac{b_1 \Rightarrow^{r!t} b_1' \quad b_2 \Rightarrow^{r?t} b_2'}{[pi_1 \mapsto b_1, pi_2 \mapsto b_2] \Longrightarrow^{r!t?r}_{pi_1, pi_2} [pi_1 \mapsto b_1', pi_2 \mapsto b_2']} \quad \text{if } pi_1 \neq pi_2$$

Annotations on $\Longrightarrow^{a}$:

$$\begin{array}{rcl} a & ::= & \epsilon \mid r!t?r \\ & \mid & t\ \mathsf{CHAN}_r \mid \mathsf{FORK}_\pi\ b \end{array}$$

MT-LAB
A VKR CENTRE OF EXCELLENCE

Simulation on behaviours

$\mathcal{S}$ is a simulation on (closed) behaviours if

- $\surd\ \mathcal{S}\ b$ if and only if $b = \surd$
- if $b_1 \overset{p_1}{\Rightarrow} b'_1$ and $b_1\ \mathcal{S}\ b_2$ then there exists b_2 and p_2 such that
  - $b_2 \overset{\widehat{p_2}}{\Rightarrow} b'_2$,
  - $p_1\ \mathcal{S}^{\partial}\ p_2$
  - $b'_1\ \mathcal{S}\ b'_2$.

where

$$\mathcal{S}^{\partial} = \{(p,p) \mid p \in \{\epsilon, r!t, r?t, t\ \mathrm{CHAN}_r\}\} \cup \{(\mathrm{FORK}_\pi\ b, \mathrm{FORK}_\pi\ b') \mid b\ \mathcal{S}\ b'\}$$

Define $\precsim$ as the largest simulation.

> Lemma: Soundness of Ordering
>
> $\sqsubseteq$ is a simulation.

The ordering $\precsim$ is undecidable.

MT-LAB A VKR CENTRE OF EXCELLENCE

Subject reduction theorem

$$CP, PP[pi \mapsto e] \longrightarrow^{ev}_{ps} CP', PP'[pi \mapsto e']$$

$$[\,] \vdash e : t \;\&\; b \qquad\qquad [\,] \vdash e' : t \;\&\; b'$$

$$b' \sqsubseteq b''$$

$$PB[pi \mapsto b] \quad \Longrightarrow^{\hat{a}}_{ps} \quad PB'[pi \mapsto b'']$$

where a is 'the translation of' ev

MT-LAB
A VKR CENTRE OF EXCELLENCE

Algorithm

The algorithm is based on Milner's $\mathcal{W}$

- use unification for the types
- collect constraints for the behaviours

Problem: Constraints need not have principal solutions: incomparable solutions could be mixed.

$\mathcal{W}(e) = (s, d, C, S)$ where

s: simple type: only behaviour variables

d: simple behaviour: no REC behaviours

C: set of constraints

S: set of solution restrictions

MT-LAB
A VKR CENTRE OF EXCELLENCE

Analysing behaviours

`pipe fs in out`:

$$\mathrm{REC}\beta'.\ (\mathrm{FORK}_\pi\ (\mathrm{REC}\ \beta.((r_1+r_0)?t;\ \epsilon;\ (r_2+r_0)!t;\ \beta))$$
$$+\quad t\ \mathrm{CHAN}_{r_0};$$
$$\mathrm{FORK}_\pi\ (\mathrm{REC}\ \beta.((r_1+r_0)?t;\ b;\ (r_2+r_0)!t;\ \beta));$$
$$\beta')$$

Finite communication topology?

- how many channels might be created?
- how many processes might be created?

MT-LAB
A VKR CENTRE OF EXCELLENCE

Analysis for finite communication topology

Judgements: $\vdash b : (\text{no. of channels, no. of processes})$

Definition:

$\vdash \epsilon : (0,0)$ $\quad\vdash r!t : (0,0)$ $\quad\vdash t?t : (0,0)$

$\vdash t\ \mathsf{CHAN}_r : (1,0)$ $\quad\dfrac{\vdash b : (n,m)}{\vdash \mathsf{FORK}_\pi b : (n, m+1)}$

$\vdash \beta : (0,0)$ $\quad\dfrac{\vdash b : (n,m)}{\vdash \mathsf{REC}\ \beta.b : (n,m)}$ provided that ...

$$\frac{\vdash b_1 : (n_1, m_1) \quad \vdash b_2 : (n_2, m_2)}{\vdash b_1; b_2 : (n_1 + n_2, m_1 + m_2)}$$ provided that ...

$$\frac{\vdash b_1 : (n_1, m_1) \quad \vdash b_2 : (n_2, m_2)}{\vdash b_1 + b_2 : (\max(n_1, n_2), \max(m_1, m_2))}$$ provided that ...

MT-LAB
A VKR CENTRE OF EXCELLENCE

Examples: Do we have a finite communication topology?

1. REC $\beta.t$ CHAN$_r + (r!t; \beta)$
2. REC $\beta.r?t \ + (t$ CHAN$_r; \beta)$
3. REC $\beta.t$ CHAN$_r + (r!t; \beta; \beta)$
4. REC $\beta.\epsilon + (r!t; \beta; \beta)$
5. REC $\beta.t$ CHAN$_r +$ FORK$_\pi(r!t; \beta)$

MT-LAB
A VKR CENTRE OF EXCELLENCE

Exercise

Determine whether or not the concurrent map functions and the fan controller have a finite communication topology.

MT-LAB
A VKR CENTRE OF EXCELLENCE

Analysing behaviours

`pipe fs in out`:

$$
\begin{aligned}
&\mathrm{REC}\,\beta'.\ (\mathrm{FORK}_\pi\ (\mathrm{REC}\ \beta.((r_1 + r_0)?t;\ \epsilon;\ (r_2 + r_0)!t;\ \beta))\\
&\quad +\quad t\ \mathrm{CHAN}_{r_0};\\
&\qquad\quad \mathrm{FORK}_\pi\ (\mathrm{REC}\ \beta.((r_1 + r_0)?t;\ b;\ (r_2 + r_0)!t;\ \beta));\\
&\qquad\quad \beta')
\end{aligned}
$$

Static processor allocation: Assume that all instances of processes labelled π will be running on the same processor - which requirements does this put on the processor?

- how many channels labelled r might be created?
- how many processes labelled π might be created?
- how many times might a channel labelled r be used for input/output?

MT-LAB A VKR CENTRE OF EXCELLENCE

Analysing behaviours

`pipe fs in out:`

$$\mathrm{REC}\beta'.\ (\mathrm{FORK}_\pi\ (\mathrm{REC}\ \beta.((r_1 + r_0)?t;\ \epsilon;\ (r_2 + r_0)!t;\ \beta))$$
$$+\quad t\ \mathrm{CHAN}_{r_0};$$
$$\mathrm{FORK}_\pi\ (\mathrm{REC}\ \beta.((r_1 + r_0)?t;\ b;\ (r_2 + r_0)!t;\ \beta));$$
$$\beta')$$

Dynamic processor allocation: Which requirements does this put on the processor?

- how many channels labelled r might be created?
- how many processes labelled π might be created?
- how many times might a channel labelled r be used for input/output?

MT-LAB
A VKR CENTRE OF EXCELLENCE

Suggested Reading (1)

- T. Amtoft, H. Riis Nielson, F. Nielson: Behavior Analysis for Validating Communication Patterns. In **Journal on Software Tools for Technology Transfer**, vol.2, pages 13-28, Springer, 1998.

  Gives an overview of the development and of a computer system for carrying out the analysis.

- H. Riis Nielson and F. Nielson: Communication Analysis for Concurrent ML. In **ML with Concurrency**. Monographs in Computer Science, pages 185-235, Springer, 1997.

MT-LAB
A VKR CENTRE OF EXCELLENCE

Suggested Reading (2)

- F. Nielson and H. Riis Nielson: From CML to its Process Algebra. Theoretical Computer Science, vol. 155, pages 179-219, 1996.
- T. Amtoft, F. Nielson, H. Riis Nielson: Type and Effect Systems: Behaviours for Concurrency. Imperial College Press, 1999.

  This book contains the full development.
- H. Riis Nielson, F. Nielson: Static and Dynamic Processor Allocation for Higher-Order Concurrent Languages. TAPSOFT'95, LNCS 915.

MT-LAB
A VKR CENTRE OF EXCELLENCE