Structured Concurrent Programming

Jayadev Misra

Department of Computer Science
University of Texas at Austin

Email: misra@cs.utexas.edu
web: http://www.cs.utexas.edu/users/psp

Collaborators: William Cook, David Kitchin

Example: Airline

- Contact two airlines simultaneously for price quotes.
- Buy ticket from either airline if its quote is at most $300.
- Buy the cheapest ticket if both quotes are above $300.
- Buy any ticket if the other airline does not provide a timely quote.
- Notify client if neither airline provides a timely quote.

Wide-area Computing

Acquire data from remote services.

Calculate with these data.

Invoke yet other remote services with the results.

Additionally

Invoke alternate services for failure tolerance.

Repeatedly poll a service.

Ask a service to notify the user when it acquires the appropriate data.

Download an application and invoke it locally.

Have a service call another service on behalf of the user.

The Nature of Distributed Applications

Three major components in distributed applications:

Persistent storage management

databases by the airline and the hotels.

Specification of sequential computational logic

does ticket price exceed $300?

Methods for orchestrating the computations

We look at only the third problem.

Overview of Orc

- Orchestration language.
  - Invoke services by calling sites
  - Manage time-outs, priorities, and failures
- A Program execution
  - calls sites,
  - publishes values.
- Simple
  - Language has only 3 combinators.
  - Semantics described by labeled transition system and traces.
  - Combinators are (monotonic and) continuous.

Structure of Orc Expression

- Simple: just a site call, $CNN(d)$

  Publishes the value returned by the site.

- composition of two Orc expressions:

do f and g in parallel	$f \mid g$	Symmetric composition
for all x from f do g	$f > x > g$	Piping
for some x from g do f	f where $x :\in g$	Asymmetric composition

Symmetric composition: $f \mid g$

$CNN \mid BBC$: calls **both** CNN and BBC simultaneously.

Publishes values returned by both sites. (0, 1 or 2 values)

- Evaluate f and g independently.
- Publish all values from both.
- No direct communication or interaction between f and g.
  They may communicate only through sites.

Pipe: $f \; > x > \; g$

For all values published by f do g. Publish only the values from g.

- $CNN \; > x > \; Email(address, x)$

  Call CNN. Bind result (if any) to x. Call $Email(address, x)$.

  Publish the value, if any, returned by $Email$.

- $(CNN \mid BBC) \; > x > \; Email(address, x)$

  May call $Email$ twice. Publishes up to two values from $Email$.

Notation

Write $f \gg g$ for $f \;>x>\; g$ if x unused in g.

Precedence: $f \;>x>\; g \mid h \;>y>\; u$
$(f \;>x>\; g) \mid (h \;>y>\; u)$

Schematic of piping

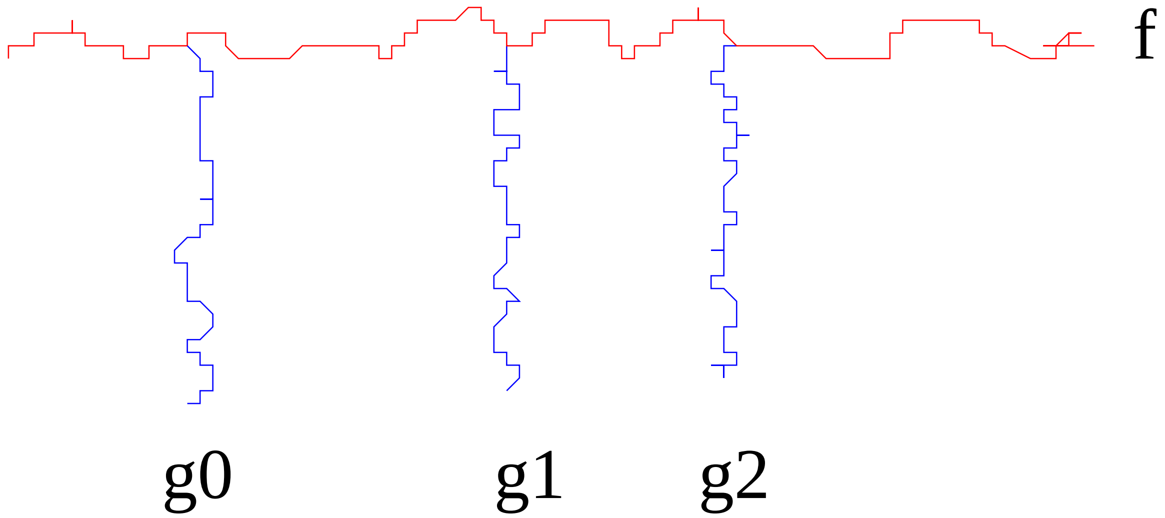


Figure 1: Schematic of $f\ {\color{red}>} x {\color{red}>}\ g$

Asymmetric parallel composition: $(f \text{ where } x{:}\in g)$

For some value published by g do f. Publish only the values from f.

$$Email(address, x) \text{ where } x{:}\in (CNN \mid BBC)$$

Binds x to the first value from $CNN \mid BBC$.

- Evaluate f and g in parallel.
  Site calls that need x are suspended; other site calls proceed.
  $(M \mid N(x)) \text{ where } x{:}\in g$

- When g returns a value, assign it to x and terminate g.
  Resume suspended calls.

- Values published by f are the values of $(f \text{ where } x{:}\in g)$.

Some Fundamental Sites

0: never responds.

$let(x, y, \cdots)$: returns a tuple of its argument values.

$if(b)$: boolean b,
returns a signal if b is true; remains silent if b is false.

$Signal$ returns a signal immediately. Same as $if(\textit{true})$.

$Rtimer(t)$: integer t, $t \geq 0$, returns a signal t time units later.

Centralized Execution Model

- An expression is evaluated on a single machine (client).
- Client communicates with sites by messages.
- All fundamental sites are local to the client.
  All except $Rtimer$ respond immediately.
- Concurrent and distributed executions are derived from an expression.

Expression Definition

$$MailOnce(a) \ \underline{\Delta}$$
$$\quad Email(a, m) \ \text{where}\ m :\in (CNN \mid BBC)$$

$$MailLoop(a, t) \ \underline{\Delta}$$
$$\quad MailOnce(a) \ \gg \ Rtimer(t) \ \gg \ MailLoop(a, t)$$

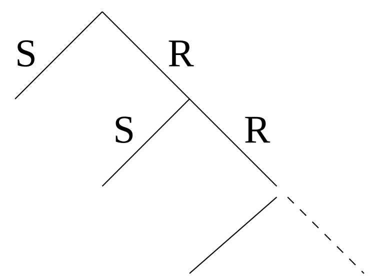
- Expression is called like a procedure.
  May publish many values. $MailLoop$ does not publish a value.

- Site calls are strict; expression calls non-strict.

Metronome

Publish a signal at every time unit.

$$Metronome \ \underline{\Delta} \ Signal \mid (Rtimer(1) \gg Metronome)$$



Publish n signals.

$$\begin{array}{lll} BM(0) & \underline{\Delta} & \mathbf{0} \\ BM(n) & \underline{\Delta} & Signal \mid (Rtimer(1) \gg BM(n-1)) \end{array}$$

Example of Expression call

- Site $Query$ returns a value (different ones at different times).
- Site $Accept(x)$ returns x if x is acceptable;
  it is silent otherwise.
- Produce all acceptable values by calling $Query$ at unit intervals forever.

$$Metronome \gg Query \; >x> \; Accept(x)$$

Time-out

Publish M's response if it arrives before t, and 0 otherwise.

$$
\begin{array}{l}
let(z) \\
\quad \text{where} \\
\qquad z :\in \\
\qquad\qquad M \\
\qquad\quad \mid Rtimer(t) \gg let(0)
\end{array}
$$

Fork-join parallelism

Call M and N in parallel.

Return their values as a tuple after both respond.

$$
\begin{array}{l}
let(u,v) \\
\quad \text{where} \;\; u{:}{\in}\; M \\
\qquad\qquad\;\; v{:}{\in}\; N
\end{array}
$$

This stands for:

$$
\begin{array}{l}
(let(u,v) \\
\quad \text{where} \;\; u{:}{\in}\; M) \\
\qquad\qquad \text{where} \;\; v{:}{\in}\; N
\end{array}
$$

Recursive definition with time-out

Call a list of sites.

Count the number of responses received within 10 time units.

$$
\begin{array}{ll}
tally([\,]) & \underline{\Delta}\ \ let(0) \\
tally(M : MS) & \underline{\Delta} \\
& u + v \\
& \quad \text{where} \\
& \qquad u{:}\in\ (M \gg let(1)) \mid (Rtimer(10) \gg let(0)) \\
& \qquad v{:}\in\ tally(MS)
\end{array}
$$

Barrier Synchronization in $M \gg f \mid N \gg g$

f and g start only after both M and N complete.

$$
\begin{array}{ll}
(& let(u, v) \\
& \text{where } u{:}\in M \\
& \qquad\quad\; v{:}\in N\,) \\
\gg & (f \mid g)
\end{array}
$$

Arbitration

In CCS/ Pi-Calculus: $\alpha.P + \beta.Q$

In Orc:

$$if(b) \gg P \mid if(\neg b) \gg Q$$

where

$$b :\in (Alpha \gg let(\textbf{\textit{true}})) \mid (Beta \gg let(\textbf{\textit{false}}))$$

Orc does not permit non-deterministic internal choice.

Priority

- Publish N's response asap, but no earlier than 1 unit from now.

$$Delay \;\underline{\Delta}\; (Rtimer(1) \gg let(u)) \text{ where } u{:\in}\, N$$

- Call M, N together.

  If M responds within one unit, take its response.

  Else, pick the first response.

$$let(x) \text{ where } x{:\in}\, (M \mid Delay)$$

Interrupt f

Evaluation of f can not be directly interrupted.

Introduce two sites:

- $Interrupt.set$: to interrupt f
- $Interrupt.get$: responds after $Interrupt.set$ has been called.

Instead of f, evaluate

$$let(z) \text{ where } z :\in (f \mid Interrupt.get)$$

Parallel or

Sites M and N return booleans. Compute their parallel or.

$ift(b) \triangleq if(b) \gg let(\textit{true})$: returns *true* if b is *true*; silent otherwise.

$$ift(x) \mid ift(y) \mid or(x, y)$$
where
$x{:}\in M,\ y{:}\in N$

To return just one value:

$$let(z)$$
where
$z{:}\in ift(x) \mid ift(y) \mid or(x, y)$
$x{:}\in M$
$y{:}\in N$

Airline quotes: Application of Parallel or

Contact airlines A and B.

Return any quote if it is below c as soon as it is available, otherwise return the minimum quote.

$threshold(x)$ returns x if $x < c$; silent otherwise.
$Min(x, y)$ returns the minimum of x and y.

$let(z)$
where
$z :\in threshold(x) \mid threshold(y) \mid Min(x, y)$
$x :\in A$
$y :\in B$

Sequential Computing

- $(S;\ T)$ is $(S \gg T)$

- **if** b **then** S **else** T

  is

  $$if(b) \gg S \mid if(\neg b) \gg T$$

- **while** $B(x)$ **do** $x{:=}\ S(x)$

  $$loop(x) \ \underline{\Delta} \\ B(x) >b> (if(b) \gg S(x) >y> loop(y) \mid if(\neg b) \gg let(x))$$

Angelic vs. Demonic non-determinism

- for all x from f do g: implements angelic non-determinism.

  All paths of computation are explored.

- for some x from f do g: implements demonic non-determinism.

  Some selected path of computation is explored.

Backtracking: Eight queens

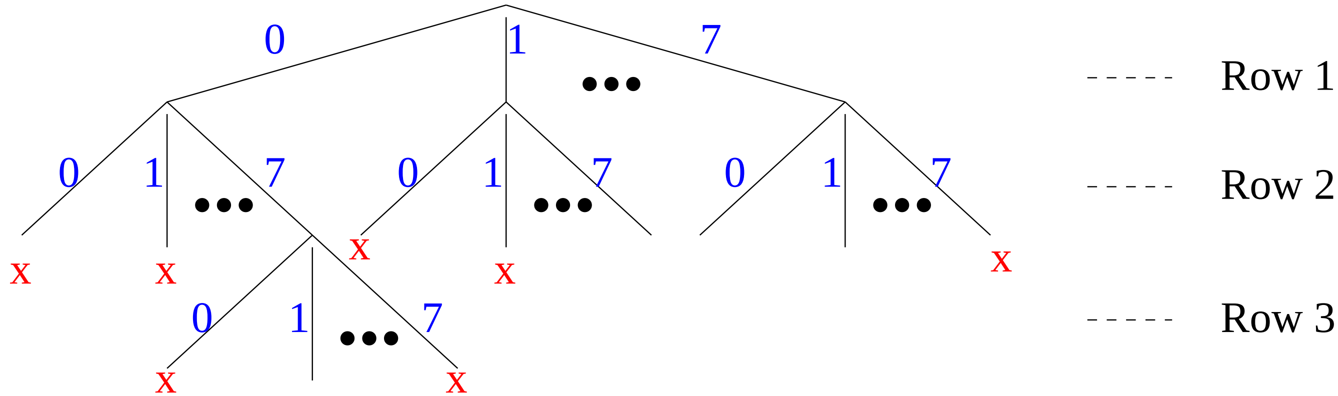


Figure 2: Backtrack Search for Eight queens

Eight queens; contd.

- configuration: placement of queens in the last i rows. Represented by a list of i values from $0..7$
- Valid configuration: no queen captures another.

  $valid(z)$ returns z if configuration z is valid; silent otherwise.
- Produce all valid extensions of z by placing n additional queens:

$$\begin{array}{lll} extend(z,1) & \underline{\Delta} & valid(0{:}z) \mid valid(1{:}z) \mid \cdots \mid valid(7{:}z) \\ extend(z,n) & \underline{\Delta} & extend(z,1) \ {>}y{>}\ extend(y, n-1) \end{array}$$

- Solve the original problem by calling $extend([\,], 8)$.

Processes

- Processes typically communicate via channels.
- For channel c, treat $c.put$ and $c.get$ as site calls.
- In our examples, $c.get$ is blocking and $c.put$ is non-blocking.
- Other kinds of channels can be programmed as sites.

Typical Iterative Process

Forever: Read x from channel c, compute with x, output result on e:

$$P(c,e) \;\underline{\Delta}\; c.get \;>x>\; Compute(x) \;>y>\; e.put(y) \;\gg\; P(c,e)$$

Process (network) to read from both c and d and write on e:

$$Net(c,d,e) \;\underline{\Delta}\; P(c,e) \mid P(d,e)$$

Interaction

Run a dialog with a child.

Forever: child inputs an integer on channel p

Process outputs *true* on channel q iff the number is prime.

Sites: $c.get$ and $c.put$, for channel c.

$Prime?(x)$ returns *true* iff x is prime.

$$
\begin{array}{ll}
Dialog(p,q) \;\underline{\Delta} & \\
\quad p.get & > x > \\
\quad Prime?(x) & > b > \\
\quad q.put(b) & \gg \\
\quad Dialog(p,q) &
\end{array}
$$

Laws of Kleene Algebra

(Zero and $\mid$)	$f \mid 0 = f$
(Commutativity of $\mid$)	$f \mid g = g \mid f$
(Associativity of $\mid$)	$(f \mid g) \mid h = f \mid (g \mid h)$
(Idempotence of $\mid$)	$f \mid f = f$
(Associativity of $\gg$)	$(f \gg g) \gg h = f \gg (g \gg h)$
(Left zero of $\gg$)	$0 \gg f = 0$
(Right zero of $\gg$)	$f \gg 0 = 0$
(Left unit of $\gg$)	$Signal \gg f = f$
(Right unit of $\gg$)	$f > x > let(x) = f$
(Left Distributivity of $\gg$ over $\mid$)	$f \gg (g \mid h) = (f \gg g) \mid (f \gg h)$
(Right Distributivity of $\gg$ over $\mid$)	$(f \mid g) \gg h = (f \gg h \mid g \gg h)$

Laws which do not hold

(Idempotence of $\mid$)	$f \mid f = f$
(Right zero of $\gg$)	$f \gg 0 = 0$
(Left Distributivity of $\gg$ over $\mid$)	$f \gg (g \mid h) = (f \gg g) \mid (f \gg h)$

Additional Laws

(Distributivity over $\gg$) if g is x-free

$$(f \gg g \text{ where } x{:}\in h) = (f \text{ where } x{:}\in h) \gg g$$

(Distributivity over $|$) if g is x-free

$$(f \mid g \text{ where } x{:}\in h) = (f \text{ where } x{:}\in h) \mid g$$

(Distributivity over where) if g is y-free

$$((f \text{ where } x{:}\in g) \text{ where } y{:}\in h)$$
$$= ((f \text{ where } y{:}\in h) \text{ where } x{:}\in g)$$

(Elimination of where) if f is x-free, for site M

$$(f \text{ where } x{:}\in M) = f \mid M \gg 0$$

Rules for Site Call

$$\frac{k \text{ fresh}}{M(v) \stackrel{M_k(v)}{\longrightarrow} ?k} \qquad (\text{SiteCall})$$

$$?k \stackrel{k?v}{\longrightarrow} let(v) \qquad (\text{SiteRet})$$

$$let(v) \stackrel{!v}{\longrightarrow} 0 \qquad (\text{Let})$$

Symmetric Composition

$$\frac{f \xrightarrow{a} f'}{f \mid g \xrightarrow{a} f' \mid g} \qquad \text{(Sym1)}$$

$$\frac{g \xrightarrow{a} g'}{f \mid g \xrightarrow{a} f \mid g'} \qquad \text{(Sym2)}$$

Sequencing

$$\frac{f \xrightarrow{a} f' \qquad a \neq !v}{f \;{>}x{>}\; g \xrightarrow{a} f' \;{>}x{>}\; g} \qquad \text{(Seq1N)}$$

$$\frac{f \xrightarrow{!v} f'}{f \;{>}x{>}\; g \xrightarrow{\tau} (f' \;{>}x{>}\; g) \mid [v/x].g} \qquad \text{(Seq1V)}$$

Asymmetric Composition

$$\frac{f \xrightarrow{a} f'}{f \text{ where } x{:}\in g \xrightarrow{a} f' \text{ where } x{:}\in g} \quad \text{(Asym1N)}$$

$$\frac{g \xrightarrow{!v} g'}{f \text{ where } x{:}\in g \xrightarrow{\tau} [v/x].f} \quad \text{(Asym1V)}$$

$$\frac{g \xrightarrow{a} g' \qquad a \neq !v}{f \text{ where } x{:}\in g \xrightarrow{a} f \text{ where } x{:}\in g'} \quad \text{(Asym2)}$$

Expression Call

$$\frac{[\![\, E(x) \ \underline{\Delta}\ f \,]\!] \in D}{E(p) \ \stackrel{\tau}{\rightarrow}\ [p/x].f} \qquad \text{(Def)}$$

Rules

$$\frac{k \text{ fresh}}{M(v) \overset{M_k(v)}{\rightarrow} ?k}$$

$$?k \overset{k?v}{\rightarrow} let(v)$$

$$let(v) \overset{!v}{\rightarrow} 0$$

$$\frac{f \overset{a}{\rightarrow} f'}{f \mid g \overset{a}{\rightarrow} f' \mid g}$$

$$\frac{g \overset{a}{\rightarrow} g'}{f \mid g \overset{a}{\rightarrow} f \mid g'}$$

$$\frac{[\![\, E(x) \;\underline{\Delta}\; f \,]\!] \in D}{E(p) \overset{\tau}{\rightarrow} [p/x].f}$$

$$\frac{f \overset{a}{\rightarrow} f' \qquad a \neq !v}{f \;>x>\; g \overset{a}{\rightarrow} f' \;>x>\; g}$$

$$\frac{f \overset{!v}{\rightarrow} f'}{f \;>x>\; g \overset{\tau}{\rightarrow} (f' \;>x>\; g) \mid [v/x].g}$$

$$\frac{f \overset{a}{\rightarrow} f'}{f \text{ where } x{:}{\in}\, g \overset{a}{\rightarrow} f' \text{ where } x{:}{\in}\, g}$$

$$\frac{g \overset{!v}{\rightarrow} g'}{f \text{ where } x{:}{\in}\, g \overset{\tau}{\rightarrow} [v/x].f}$$

$$\frac{g \overset{a}{\rightarrow} g' \qquad a \neq !v}{f \text{ where } x{:}{\in}\, g \overset{a}{\rightarrow} f \text{ where } x{:}{\in}\, g'}$$

Example

$$((M(x) \mid let(x)) \;>y>\; R(y)) \text{ where } x{:}\in (N \mid S)$$

$\overset{S_k}{\rightarrow}$ {Call S: $S \overset{S_k}{\rightarrow} ?k$; $N \mid S \overset{S_k}{\rightarrow} N \mid ?k$}

$$((M(x) \mid let(x)) \;>y>\; R(y)) \text{ where } x{:}\in (N \mid ?k)$$

$\overset{N_l}{\rightarrow}$ {Call N}

$$((M(x) \mid let(x)) \;>y>\; R(y)) \text{ where } x{:}\in (?l \mid ?k)$$

$\overset{l?5}{\rightarrow}$ { $?l \overset{l?5}{\rightarrow} let(5)$; $?l \mid ?k \overset{l?5}{\rightarrow} let(5) \mid ?k$}

$$((M(x) \mid let(x)) \;>y>\; R(y)) \text{ where } x{:}\in (let(5) \mid ?k)$$

Example; contd.

$$((M(x) \mid let(x)) \;>y>\; R(y)) \text{ where } x{:}{\in}\; (let(5) \mid ?k)$$

$\xrightarrow{\tau}\{\; let(5) \xrightarrow{!5} 0;\;\; let(5) \mid ?k \xrightarrow{!5} 0 \mid ?k\,\}$

$$(M(5) \mid let(5)) \;>y>\; R(y)$$

$\xrightarrow{\tau}\{\; let(5) \xrightarrow{!5} 0;\;\; M(5) \mid let(5) \xrightarrow{!5} M(5) \mid 0;$
$f \xrightarrow{!v} f'$ implies $f \;>y>\; g \xrightarrow{\tau} (f' \;>y>\; g) \mid [v/y].g\,\}$

$$((M(5) \mid 0) \;>y>\; R(y)) \mid R(5)$$

$\xrightarrow{R_n(5)}$ {call R with argument (5)}

$$((M(5) \mid 0) \;>y>\; R(y)) \mid ?n$$

Example; contd.

$$((M(5) \mid 0) \;>y>\; R(y)) \mid ?n$$

$$\stackrel{n?7}{\rightarrow} \{\; ?n \;\stackrel{n?7}{\rightarrow}\; let(7) \;\}$$

$$((M(5) \mid 0) \;>y>\; R(y)) \mid let(7)$$

$$\stackrel{!7}{\rightarrow} \{\; f \mid let(7) \;\stackrel{!7}{\rightarrow}\; f \mid 0 \;\}$$

$$((M(5) \mid 0) \;>y>\; R(y)) \mid 0$$

The sequence of events:	$S_k \quad N_l \quad l?5 \quad \tau \quad \tau \quad R_n(5) \quad n?7 \quad !7$
The sequence minus τ events:	$S_k \quad N_l \quad l?5 \quad R_n(5) \quad n?7 \quad !7$

Executions and Traces

Define $f \overset{\epsilon}{\Rightarrow} f$ $\qquad \dfrac{f \overset{a}{\rightarrow} f'',\ f'' \overset{s}{\Rightarrow} f'}{f \overset{as}{\Rightarrow} f'}$

- Given $f \overset{s}{\Rightarrow} f'$, s is an execution of f.
- A trace is an execution minus τ events.
- The set of executions of f (and traces) are prefix-closed.

Laws, using strong bisimulation

- $f \mid 0 \;\sim\; f$
- $f \mid g \;\sim\; g \mid f$
- $f \mid (g \mid h) \;\sim\; (f \mid g) \mid h$
- $f \;>x>\; (g \;>y>\; h) \;\sim\; (f \;>x>\; g) \;>y>\; h$, if h is x-free.
- $0 \;>x>\; f \;\sim\; 0$
- $(f \mid g) \;>x>\; h \;\sim\; f \;>x>\; h \mid g \;>x>\; h$
- $(f \mid g)$ where $x{:}\in h \;\sim\; (f$ where $x{:}\in h) \mid g$, if g is x-free.
- $(f \;>y>\; g)$ where $x{:}\in h \;\sim\; (f$ where $x{:}\in h) \;>y>\; g$, if g is x-free.
- $(f$ where $x{:}\in g)$ where $y{:}\in h \;\sim\; (f$ where $y{:}\in h)$ where $x{:}\in g$, if g is y-free, h is x-free.

Relation $\sim$ is an equality

Given $f \sim g$, show

1. $f \mid h \sim g \mid h$
   $h \mid f \sim h \mid g$

2. $f > x > h \sim g > x > h$
   $h > x > f \sim h > x > g$

3. $f \text{ where } x :\in h \sim g \text{ where } x :\in h$
   $h \text{ where } x :\in f \sim h \text{ where } x :\in g$

Treatment of Free Variables

Closed expression: No free variable.
Open expression: Has free variable.

- Law $f \sim g$ holds only if both f and g are closed.

  Otherwise: $let(x) \sim 0$

  But $let(1) > x > 0 \neq let(1) > x > let(x)$

- Then we can't show $let(x) \mid let(y) \sim let(y) \mid let(x)$

Substitution Event

$$f \stackrel{[v/x]}{\longrightarrow} [v/x].f \qquad \text{(SUBST)}$$

- Now, $let(x) \stackrel{[1/x]}{\longrightarrow} let(1)$.

  So, $let(x) \neq 0$

- Earlier rules apply to base events only.

  From $f \stackrel{[v/x]}{\longrightarrow} [v/x].f$, we can not conclude:

  $f \mid g \stackrel{[v/x]}{\longrightarrow} [v/x].f \mid g$

Traces as Denotations

Define Orc combinators over trace sets, S and T. Define:

$$S \mid T, \quad S \;>x>\; T, \quad S \text{ where } x{:}\!\in T.$$

Notation: $\langle f \rangle$ is the set of traces of f.

Theorem

$$\begin{array}{lcl}
\langle f \mid g \rangle & = & \langle f \rangle \mid \langle g \rangle \\
\langle f \;>x>\; g \rangle & = & \langle f \rangle \;>x>\; \langle g \rangle \\
\langle f \text{ where } x{:}\!\in g \rangle & = & \langle f \rangle \text{ where } x{:}\!\in \langle g \rangle
\end{array}$$

Expressions are equal if their trace sets are equal

Define: $f \cong g$ if $\langle f \rangle = \langle g \rangle$.

Theorem (Combinators preserve $\cong$)

Given $f \cong g$ and any combinator $*$: $f * h \cong g * h$, $h * f \cong h * g$

Specifically, given $f \cong g$

1. $f \mid h \cong g \mid h$
   $h \mid f \cong h \mid g$

2. $f > x > h \cong g > x > h$
   $h > x > f \cong h > x > g$

3. $f \text{ where } x{:}\in h \cong g \text{ where } x{:}\in h$
   $h \text{ where } x{:}\in f \cong h \text{ where } x{:}\in g$

Monotonicity, Continuity

- Define: $f \sqsubseteq g$ if $\langle f \rangle \subseteq \langle g \rangle$.

  Theorem (Monotonicity) Given $f \sqsubseteq g$ and any combinator $*$

  $$f * h \sqsubseteq g * h, \quad h * f \sqsubseteq h * g$$

- Chain f: $f_0 \sqsubseteq f_1, \cdots f_i \sqsubseteq f_{i+1}, \cdots$.

  Theorem: $\sqcup(f_i * h) \cong (\sqcup f) * h$.

  Theorem: $\sqcup(h * f_i) \cong h * (\sqcup f)$.

Least Fixed Point

$$M \quad \underline{\Delta} \quad S \mid R \gg M$$

$$M_0 \cong 0$$
$$M_{i+1} \cong S \mid R \gg M_i\,, \quad i \geq 0$$

M is the least upper bound of the chain $M_0 \sqsubseteq M_1 \sqsubseteq \cdots$

Weak Bisimulation

$$signal \gg f \quad \cong \quad f$$
$$f \;>x>\; let(x) \quad \cong \quad f$$