

# 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, Ian Wehrman

## 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 Orchestration language **Orc**

- A Program execution
  - calls **sites**, to invoke services.
  - publishes **values**.
- Orc is simple
  - Language has only 3 combinators.
  - Can handle time-outs, priorities, failures, synchronizations, ...
  - 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 $f$	$f <x< g$	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  $Email$ .

- $(CNN | BBC) > x > Email(address, x)$

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

## Schematic of piping

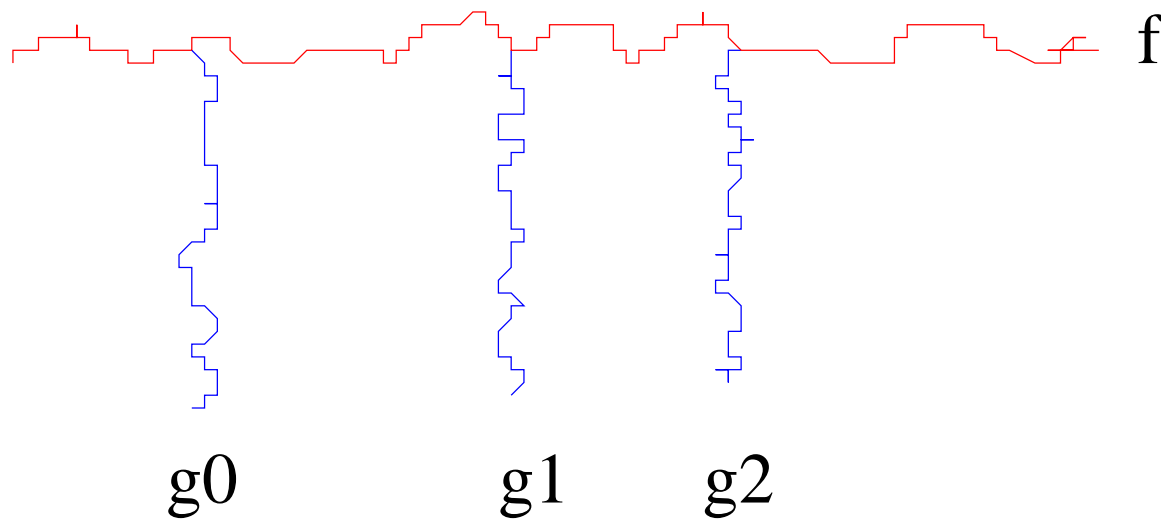


Figure 1: Schematic of  $f > x > g$

## Notation

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

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

## Asymmetric parallel composition: $(f \text{ < } x \text{ < } g)$

For some value published by  $g$  do  $f$ .

- Evaluate  $f$  and  $g$  in parallel.  
Site calls that need  $x$  are suspended; other site calls proceed.

$$(M \mid N(x)) \text{ < } x \text{ < } g$$

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

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

$$\textit{Email}(\textit{address}, x) \text{ < } x \text{ < } (\textit{CNN} \mid \textit{BBC})$$

Binds  $x$  to the first value from  $\textit{CNN} \mid \textit{BBC}$ . Sends at most one email.

## Some Fundamental Sites

$0$ : never responds.

$let(x, y, \dots)$ : 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  $if(true)$ .

$Rtimer(t)$ : 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

$\text{MailOnce}(a)$   $\triangle$   
 $\text{Email}(a, m) < m < (\text{CNN} \mid \text{BBC})$

$\text{MailLoop}(a, d)$   $\underline{\triangle}$   
 $\text{MailOnce}(a) \gg \text{Rtimer}(d) \gg \text{MailLoop}(a, d)$

- Expression is called like a procedure.  
May publish many values. *MailLoop* does not publish a value.
- Site calls are strict; expression calls non-strict.

## Another Strategy

- Start  $MailOnce(a)$  and  $Rtimer(t)$  simultaneously.

$$\begin{array}{c}
 MailLoop(a, d) \triangle \\
 MailOnce(a) \mid Rtimer(d) \gg MailLoop(a, d)
 \end{array}$$

Compare with

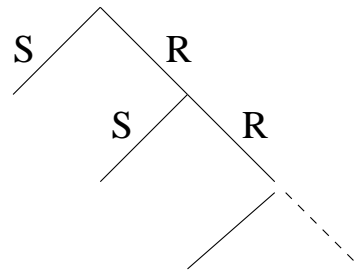
$$\begin{array}{c}
 MailLoop(a, d) \triangle \\
 MailOnce(a) \gg Rtimer(d) \gg MailLoop(a, d)
 \end{array}$$

What if  $MailOnce(a)$  does not respond for 3 days?

# Metronome

Publish a signal at every time unit.

$Metronome \triangle Signal \mid (Rtimer(1) \gg Metronome)$



Publish  $n$  signals.

$BM(0) \triangle 0$

$BM(n) \triangle Signal \mid (Rtimer(1) \gg BM(n - 1))$

## 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.

$$\text{let}(z)$$
$$\langle z \langle M \mid (\text{Rtimer}(t) \gg \text{let}(0))$$

## Fork-join parallelism

Call  $M$  and  $N$  in parallel.

Return their values as a tuple after both respond.

$$\begin{aligned} & \text{let}(u, v) \\ & \quad \langle u \rangle \langle M \rangle \\ & \quad \langle v \rangle \langle N \rangle \end{aligned}$$

Notational Convention:

$$\begin{aligned} & \text{let}(u, v) \\ & \quad \langle u \rangle \langle M \rangle \\ & \quad \langle v \rangle \langle N \rangle \end{aligned}$$

## Recursive definition with time-out

Call a list of sites.

Count the number of responses received within 10 time units.

$$\begin{array}{l}
 \textit{tally}([]) \quad \triangle \textit{let}(0) \\
 \textit{tally}(M : MS) \quad \triangle u + v \\
 \quad \langle u \rangle (M \gg \textit{let}(1)) \mid (\textit{Rtimer}(10) \gg \textit{let}(0)) \\
 \quad \langle v \rangle \textit{tally}(MS)
 \end{array}$$

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

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

Rendezvous of CSP or CCS;  $M$  and  $N$  are complementary actions.

$$\begin{aligned} & ( \text{let}(u, v) \\ & \quad \langle u \langle M \\ & \quad \langle v \langle N \rangle \\ \gg & (f \mid g) \end{aligned}$$

## Priority

- Publish  $N$ 's response asap, but no earlier than 1 unit from now.  
Apply fork-join between  $Rtimer(1)$  and  $N$ .

$$Delay \triangle (Rtimer(1) \gg let(u) <u < N$$

- Call  $M$ ,  $N$  together.

If  $M$  responds within one unit, take its response.

Else, pick the first response.

$$let(x) <x < (M \mid Delay)$$

## 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

$$\text{let}(z) <z < (f \mid \text{Interrupt.get})$$

## Parallel or

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

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

$$ift(x) \mid ift(y) \mid or(x, y)$$

$$\langle x \rangle M$$

$$\langle y \rangle N$$

To return just one value:

$$let(z)$$

$$\langle z \rangle ift(x) \mid ift(y) \mid or(x, y)$$

$$\langle x \rangle M$$

$$\langle y \rangle N$$

## 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  $y$ .

$let(z)$

$<z < threshold(x) \mid threshold(y) \mid Min(x, y)$

$<x < A$

$<y < B$

# 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**  $x := S(x)$

$$loop(x) \triangleq B(x) \gg b \gg (if(b) \gg S(x) \gg y \gg loop(y) \mid if(\neg b) \gg let(x))$$

## 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

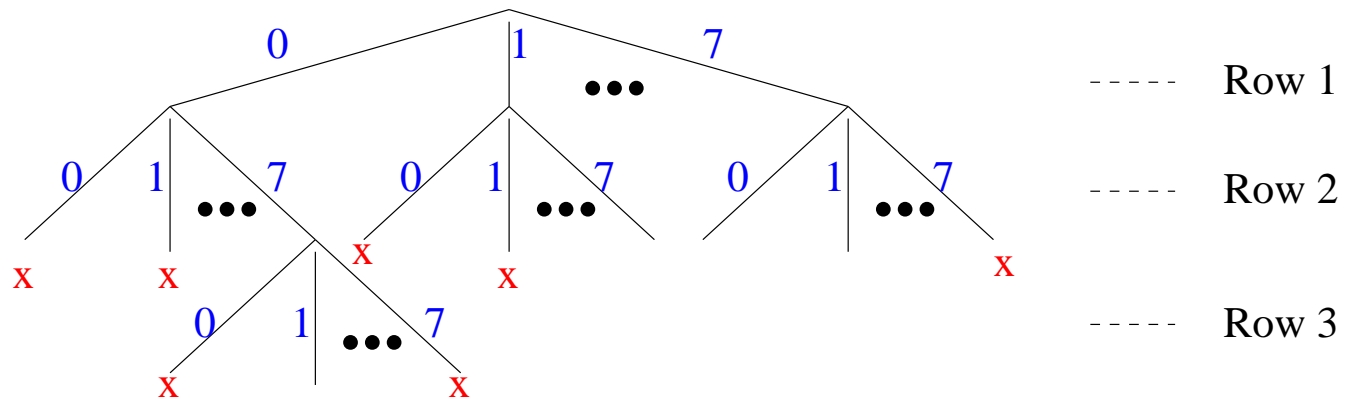


Figure 2: Backtrack Search for Eight queens

## Eight queens; contd.

$extend(z, 1) \triangleq valid(0:z) \mid valid(1:z) \mid \dots \mid valid(7:z)$

$extend(z, n) \triangleq extend(z, 1) > y > extend(y, n - 1)$

- $z$ : partial placement of queens (list of values from 0..7)
- $extend(z, n)$  publishes **all** valid extensions of  $z$  with  $n$  additional queens.
- $valid(z)$  returns  $z$  if  $z$  is valid; silent otherwise.
- 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  $e$ :

$$P(c, e) \triangleq c.get \ >x> \ Compute(x) \ >y> \ e.put(y) \ \gg \ P(c, e)$$

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

$$Net(c, d, e) \triangleq P(c, e) \mid P(d, e)$$

## Interaction: Run a dialog

User inputs an integer on channel  $p$

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

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

$Dialog(p, q) \triangle$   
 $p.get \quad > x >$   
 $Prime?(x) \quad > b >$   
 $q.put(b) \quad \gg$   
 $Dialog(p, q)$

## Laws of Kleene Algebra

(Zero and  $|$ )

$$f | 0 = f$$

(Commutativity of  $|$ )

$$f | g = g | f$$

(Associativity of  $|$ )

$$(f | g) | h = f | (g | h)$$

(Idempotence of  $|$ )

$$f | f = f$$

(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  $\gg$ )

$$\text{Signal} \gg f = f$$

(Right unit of  $\gg$ )

$$f \gg x \text{ let}(x) = f$$

(Left Distributivity of  $\gg$  over  $|$ )

$$f \gg (g | h) = (f \gg g) | (f \gg h)$$

(Right Distributivity of  $\gg$  over  $|$ )

$$(f | g) \gg h = (f \gg h) | (g \gg h)$$

## Laws which do not hold

(Idempotence of  $|$ )

$$f | f = f$$

(Right zero of  $\gg$ )

$$f \gg 0 = 0$$

(Left Distributivity of  $\gg$  over  $|$ )

$$f \gg (g | h) = (f \gg g) | (f \gg h)$$

## Additional Laws

(Distributivity over  $\gg$ ) if  $g$  is  $x$ -free  
 $((f \gg g) \langle x \rangle h) = (f \langle x \rangle h) \gg g$

(Distributivity over  $|$ ) if  $g$  is  $x$ -free  
 $((f | g) \langle x \rangle h) = (f \langle x \rangle h) | g$

(Distributivity over where) if  $g$  is  $y$ -free  
 $((f \langle x \rangle g) \langle y \rangle h)$   
 $= ((f \langle y \rangle h) \langle x \rangle g)$

(Elimination of where) if  $f$  is  $x$ -free, for site  $M$   
 $(f \langle x \rangle M) = f | (M \gg 0)$

## Why Bother with a Formal Semantics?

- Precise definition of execution
- Prove certain identities as reality check
- Guidance for implementation
- Avoid oversight of issues
- Possibility of enhancement of the programming model

# Operational Semantics

**Asynchronous Orc:** No notion of time

- no *Rtimer*
- no requirement that anything be done:  
In  $f \mid g$ ,  $f$  may start whereas  $g$  may never start.

## Events, Transitions

$$f \xrightarrow{a} f'$$

denotes evaluation of expression  $f$  may cause event  $a$  and then transit to  $f'$ . Example:  $let(v) \xrightarrow{!v} 0$

Event:

- **Site Call:**  $M_k(v)$ , call site  $M$  with parameter values  $v$ ; call identified by  $k$ .
- **Site Response:**  $k?m$ , site responds with value  $m$  for call  $k$ .
- **Publication:**  $!v$ , publish  $v$ .
- **Silent (internal):**  $\tau$

## Rules for Site Call

$$M(v) \xrightarrow{M_k(v)} ?k, \quad k \text{ fresh} \quad (\text{SITECALL})$$

$$?k \xrightarrow{k?v} \text{let}(v) \quad (\text{SITERET})$$

$$\text{let}(v) \xrightarrow{!v} 0 \quad (\text{LET})$$

## Symmetric Composition

$$\frac{f \xrightarrow{a} f'}{f | g \xrightarrow{a} f' | g} \quad (\text{SYM1})$$

$$\frac{g \xrightarrow{a} g'}{f | g \xrightarrow{a} f | g'} \quad (\text{SYM2})$$

# Sequencing

$$\frac{f \xrightarrow{a} f' \quad a \neq !v}{f \langle x \rangle g \xrightarrow{a} f' \langle x \rangle g} \quad (\text{SEQ1N})$$

$$\frac{f \xrightarrow{!v} f'}{f \langle x \rangle g \xrightarrow{\tau} (f' \langle x \rangle g) \mid [v/x].g} \quad (\text{SEQ1V})$$

# Asymmetric Composition

$$\frac{f \xrightarrow{a} f'}{f \langle x \rangle g \xrightarrow{a} f' \langle x \rangle g} \quad (\text{ASYM1N})$$

$$\frac{g \xrightarrow{a} g' \quad a \neq !v}{f \langle x \rangle g \xrightarrow{a} f \langle x \rangle g'} \quad (\text{ASYM2})$$

$$\frac{g \xrightarrow{!v} g'}{f \langle x \rangle g \xrightarrow{\tau} [v/x].f} \quad (\text{ASYM1V})$$

## Expression Call

$$\frac{[[E(x) \triangle f]] \in D}{E(p) \xrightarrow{\tau} [p/x].f} \quad (\text{DEF})$$

## Rules

$$M(v) \xrightarrow{M_k(v)} ?k, \quad k \text{ fresh}$$

$$?k \xrightarrow{k?v} \text{let}(v)$$

$$\text{let}(v) \xrightarrow{!v} 0$$

$$\frac{f \xrightarrow{a} f'}{f \mid g \xrightarrow{a} f' \mid g}$$

$$\frac{g \xrightarrow{a} g'}{f \mid g \xrightarrow{a} f \mid g'}$$

$$\frac{[[E(x) \triangle f]] \in D}{E(p) \xrightarrow{\tau} [p/x].f}$$

$$\frac{f \xrightarrow{a} f' \quad a \neq !v}{f \langle x \rangle g \xrightarrow{a} f' \langle x \rangle g}$$

$$\frac{f \xrightarrow{!v} f'}{f \langle x \rangle g \xrightarrow{\tau} (f' \langle x \rangle g) \mid [v/x].g}$$

$$\frac{f \xrightarrow{a} f'}{f \langle x \rangle g \xrightarrow{a} f' \langle x \rangle g}$$

$$\frac{g \xrightarrow{a} g' \quad a \neq !v}{f \langle x \rangle g \xrightarrow{a} f \langle x \rangle g'}$$

$$\frac{g \xrightarrow{!v} g'}{f \langle x \rangle g \xrightarrow{\tau} [v/x].f}$$

## Example

$$((M(x) \mid \text{let}(x)) \text{ > } y \text{ > } R(y)) \text{ < } x \text{ < } (N \mid S)$$

$$\xrightarrow{S_k} \{\text{Call } S: S \xrightarrow{S_k} ?k; N \mid S \xrightarrow{S_k} N \mid ?k\}$$

$$((M(x) \mid \text{let}(x)) \text{ > } y \text{ > } R(y)) \text{ < } x \text{ < } (N \mid ?k)$$

$$\xrightarrow{N_l} \{\text{Call } N\}$$

$$((M(x) \mid \text{let}(x)) \text{ > } y \text{ > } R(y)) \text{ < } x \text{ < } (?l \mid ?k)$$

$$\xrightarrow{l?5} \{ ?l \xrightarrow{l?5} \text{let}(5); ?l \mid ?k \xrightarrow{l?5} \text{let}(5) \mid ?k \}$$

$$((M(x) \mid \text{let}(x)) \text{ > } y \text{ > } R(y)) \text{ < } x \text{ < } (\text{let}(5) \mid ?k)$$

## Example; contd.

$$((M(x) \mid let(x)) \>y> R(y)) \<x< (let(5) \mid ?k)$$

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

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

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

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

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

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

## Example; contd.

$$\begin{array}{c} ((M(5) \mid 0) > y > R(y)) \mid ?n \\ \xrightarrow{n?7} \{ ?n \xrightarrow{n?7} let(7) \} \end{array}$$

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

$$\xrightarrow{!7} \{ f \mid let(7) \xrightarrow{!7} f \mid 0 \}$$

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

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  $\tau$  events:  $S_k \quad N_l \quad l?5 \quad R_n(5) \quad n?7 \quad !7$

## Executions and Traces

Define  $f \xRightarrow{\epsilon} f$   $\frac{f \xrightarrow{a} f'', f'' \xRightarrow{s} f'}{f \xRightarrow{as} f'}$

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

## Laws, using strong bisimulation

Define  $f \sim g$ : Executions of  $f$  and  $g$  are equal.

- $f \mid 0 \sim f$
- $f \mid g \sim g \mid f$
- $f \mid (g \mid h) \sim (f \mid g) \mid h$
- $f \langle x \rangle (g \langle y \rangle h) \sim (f \langle x \rangle g) \langle y \rangle h$ , if  $h$  is  $x$ -free.
- $0 \langle x \rangle f \sim 0$
- $(f \mid g) \langle x \rangle h \sim f \langle x \rangle h \mid g \langle x \rangle h$
- $(f \mid g) \langle x \rangle h \sim (f \langle x \rangle h) \mid g$ , if  $g$  is  $x$ -free.
- $(f \langle y \rangle g) \langle x \rangle h \sim (f \langle x \rangle h) \langle y \rangle g$ , if  $g$  is  $x$ -free.
- $(f \langle x \rangle g) \langle y \rangle h \sim (f \langle y \rangle h) \langle x \rangle g$ , if  $g$  is  $y$ -free  
 $h$  is  $x$ -free.

## Relation $\sim$ is an equality

Given  $f \sim g$ , show

$$1. \quad \begin{array}{l} f \mid h \sim g \mid h \\ h \mid f \sim h \mid g \end{array}$$

$$2. \quad \begin{array}{l} f > x > h \sim g > x > h \\ h > x > f \sim h > x > g \end{array}$$

$$3. \quad \begin{array}{l} f < x < h \sim g < x < h \\ h < x < f \sim h < x < g \end{array}$$

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

- If we work with only open expressions, we can't show

$let(x) | let(y) \sim let(y) | let(x)$

## Substitution Event

$$f \xrightarrow{[v/x]} [v/x].f \quad (\text{SUBST})$$

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

So,  $let(x) \neq 0$

- Earlier rules apply to base events only.

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

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

## Traces as Denotations

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

$$S \mid T, S \succ x \succ T, S \prec x \prec T.$$

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

### Theorem

$$\begin{aligned} \langle f \mid g \rangle &= \langle f \rangle \mid \langle g \rangle \\ \langle f \succ x \succ g \rangle &= \langle f \rangle \succ x \succ \langle g \rangle \\ \langle f \prec x \prec g \rangle &= \langle f \rangle \prec x \prec \langle g \rangle \end{aligned}$$

## Trace equality is Equality

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.  $f < x < h \cong g < x < h$   
 $h < x < f \cong h < x < g$

## Assigning Meaning to Recursion

$M \triangleq S \mid R \gg M$  — Metronome

To compute executions (traces) of  $M$ , define

$$M_0 \triangleq 0$$

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

Take  $M$  to be the least upper bound of the chain  $M_0, M_1, \dots$ .

We need:

- (Monotonicity) A partial order  $\sqsubseteq$  such that  $M_0 \sqsubseteq M_1 \sqsubseteq \dots$
- (Continuity) Least upper bound of this chain exists and has desirable properties.

## Monotonicity, Continuity

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

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

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

- Given chain  $f: f_0 \sqsubseteq f_1, \dots, f_i \sqsubseteq f_{i+1}, \dots$ , its least upper bound exists. Write it as  $\sqcup f$ .

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

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

## Treatment of Time

$$f \xrightarrow{t,a} f'$$

Evaluation of  $f$  may cause event  $a$  at  $t$  units after the evaluation starts. No event precedes  $a$ . Transition to  $f'$ .

$$\text{let}(7) \xrightarrow{0,!7} 0$$

$$\text{Rtimer}(3) \xrightarrow{3,!} 0$$

$$\text{Rtimer}(3) \mid \text{Rtimer}(5) \xrightarrow{3,!} 0 \mid \text{Rtimer}(2)$$

## Time-Shifted Expression

$f^t$ : expression resulting from  $f$  after  $t$  units have elapsed **without occurrence of an event**.

$f^t = \perp$  if  $t$  units can not elapse without an event.

$$Rtimer(5)^0 = Rtimer(5)$$

$$Rtimer(5)^3 = Rtimer(2)$$

$$Rtimer(5)^6 = \perp$$

$$let(v)^1 = \perp$$

Expression containing  $\perp$  is  $\perp$ .

$$f \xrightarrow{t,a} \perp$$

means  $f$  does not engage in  $(t, a)$ .

## Definition of Time-shifted Expressions

$$\begin{array}{lll}
 (f \mid g)^t & \triangleq & f^t \mid g^t \\
 (f \succ x \succ g)^t & \triangleq & f^t \succ x \succ g \\
 (f \prec x \prec g)^t & \triangleq & f^t \prec x \prec g^t \\
 M(x)^t & \triangleq & M(x) \\
 M(m)^t & \triangleq & \begin{cases} M(m) & \text{if } t = 0 \\ \perp & \text{otherwise.} \end{cases}
 \end{array}$$

## Facts about Time-Shifted Expressions

$$f^0 = f$$

$$(f^s)^t = f^{s+t}$$

$$f^s \xrightarrow{t,a} h \equiv f \xrightarrow{s+t,a} h$$

## Rules

$$\begin{array}{c}
 M(v) \xrightarrow{0, M_k(v)} ?k, \quad k \text{ fresh} \\
 \\
 ?k \xrightarrow{t, k?v} \text{let}(v) \\
 \\
 \text{let}(v) \xrightarrow{0, !v} 0 \\
 \\
 \frac{f \xrightarrow{t, a} f'}{f \mid g \xrightarrow{t, a} f' \mid g^t} \\
 \\
 \frac{g \xrightarrow{t, a} g'}{f \mid g \xrightarrow{t, a} f^t \mid g'} \\
 \\
 \frac{[[E(x) \triangle f]] \in D}{E(p) \xrightarrow{0, \tau} [p/x].f} \\
 \\
 \frac{f \xrightarrow{t, a} f' \quad a \neq !v}{f \langle x \rangle g \xrightarrow{t, a} f' \langle x \rangle g} \\
 \\
 \frac{f \xrightarrow{t, !v} f'}{f \langle x \rangle g \xrightarrow{t, \tau} (f' \langle x \rangle g) \mid [v/x].g} \\
 \\
 \frac{f \xrightarrow{t, a} f'}{f \langle x \rangle g \xrightarrow{t, a} f' \langle x \rangle g^t} \\
 \\
 \frac{g \xrightarrow{t, a} g' \quad a \neq !v}{f \langle x \rangle g \xrightarrow{t, a} f^t \langle x \rangle g'} \\
 \\
 \frac{g \xrightarrow{t, !v} g'}{f \langle x \rangle g \xrightarrow{t, \tau} [v/x].f^t}
 \end{array}$$

# Execution, Trace, Substitution

## Execution

$$f \xRightarrow{\epsilon} f$$

$$\frac{f \xrightarrow{(t,a)} f'', f'' \xRightarrow{u} f'}{f \xRightarrow{(t,a)u_t} f'}$$

## Substitution Event

$$f \xrightarrow{t, [v/x]} [v/x].(f^t)$$

## Example of Execution

$$\begin{aligned}
 & (Rtimer(3) \gg M(x)) \mid N \\
 \xrightarrow{0, N_k} & \{ (SYM2), (CALL) \} \\
 & (Rtimer(3) \gg M(x)) \mid ?k \\
 \xrightarrow{2, k?7} & \{ (SYM2), (RETURN), \text{response for } ?k \text{ is } (2, 7) \} \\
 & (Rtimer(3)^2 \gg M(x)) \mid let(7) \\
 \xrightarrow{0, !7} & \{ \text{Rule for } let \} \\
 & (Rtimer(3)^2 \gg M(x)) \mid 0 \\
 \xrightarrow{1, \tau} & \{ (SYM1), (SEQ1V), (RETURN), Rtimer(3)^2 = Rtimer(1) \} \\
 & ((0 \gg M(x)) \mid M(x)) \mid 0 \\
 = & \{ \text{simplifying} \} \\
 & M(x)
 \end{aligned}$$

$$\begin{aligned}
 u = & \quad (0, N_k) \quad (2, k?7) \quad (2, !7) \quad (3, \tau) \\
 trace(u) = & \quad \quad \quad \quad \quad \quad \quad (2, !7)
 \end{aligned}$$