Structured Wide-Area Programming: Orc Calculus

Jayadev Misra

Department of Computer Science
University of Texas at Austin

http://orc.csres.utexas.edu

Concurrency

- ubiquitous.
- difficult.
- important.

Some Typical Applications

- Map-Reduce using a server farm
- Thread management in an operating system
- Mashups (Internet Scripting)
- Reactive Programming
- Extended 911:
  Using humans as components
  Components join and leave
  Real-time response

Traditional approaches to handling Concurrency

- Adding concurrency to serial languages:
  - Threads with mutual exclusion using semaphore.
  - Transaction.
- Process Networks.

Features needed in a Concurrent Programming Language

- Describe entities and their interactions.
- Describe passage of time.
- Allow birth and death of entities.
- Allow programming of novel interactions.
- Support hierarchical structure.

Orc

- Initial Goal: Internet scripting language.
- Next: Component integration language.
- Next: A general purpose, structured “concurrent programming language”.
- A very late realization: A simulation language.

Internet Scripting

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

-

Structured Concurrent Programming

- Structured Sequential Programming: Dijkstra circa 1968
  Component Integration in a sequential world.

- Structured Concurrent Programming:
  Component Integration in a concurrent world.

Philosophy of our Language Design

- Start with Concurrency. Add sequential features later.
- Impose hierarchical structure: compose concurrent programs.
- Introduce very few composition mechanisms (Combinators).

Orc Basics

- Site: Basic service or component.
- Concurrency combinators for integrating sites.
- Theory includes nothing other than the combinators.

No notion of data type, thread, process, channel, synchronization, parallelism $\cdots$

New concepts are programmed using new sites.

Examples of Sites

- $+$ $-$ $*$ $\&\&$ $||$ $=$ ...
- `Println`, `Random`, `Prompt`, `Email` ...
- Mutable Ref, Semaphore, Channel, ...
- `Timer`
- External Services: Google Search, MySpace, CNN, ...
- Any Java Class instance, Any Orc Program
- Factory sites; Sites that create sites: `Semaphore`, `Channel` ...
- Humans

...

Sites

- A site is called like a procedure with parameters.
- Site returns at most one value.
- The value is published.

Site calls are strict.

Overview of Orc

- Orc program has
  - a goal expression,
  - a set of definitions.
- The goal expression is executed. Its execution
  - calls sites,
  - publishes values.

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	$f >x> g$	Sequential composition
for some x from g do f	$f <x< g$	Pruning
if f halts without publishing do g	$f \; ; \; g$	Otherwise

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	$f >x> g$	Sequential composition
for some x from g do f	$f <x< g$	Pruning
if f halts without publishing do g	$f \; ; \; g$	Otherwise

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	$f >x> g$	Sequential composition
for some x from g do f	$f <x< g$	Pruning
if f halts without publishing do g	$f \; ; \; g$	Otherwise

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	$f >x> g$	Sequential composition
for some x from g do f	$f <x< g$	Pruning
if f halts without publishing do g	$f \; ; \; g$	Otherwise

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	$f >x> g$	Sequential composition
for some x from g do f	$f <x< g$	Pruning
if f halts without publishing do g	$f \; ; \; g$	Otherwise

Symmetric composition: $f \mid g$

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

  Example: $CNN(d) \mid BBC(d)$

  Calls both CNN and BBC simultaneously.
  Publishes values returned by both sites. (0, 1 or 2 values)

Sequential composition: $f \ >x> \ g$

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

- $CNN(d) \ >x> \ Email(address, x)$
  - Call $CNN(d)$.
  - Bind result (if any) to x.
  - Call $Email(address, x)$.
  - Publish the value, if any, returned by $Email$.

- $(CNN(d) \mid BBC(d)) \ >x> \ Email(address, x)$
  - May call $Email$ twice.
  - Publishes up to two values from $Email$.

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

Schematic of Sequential composition

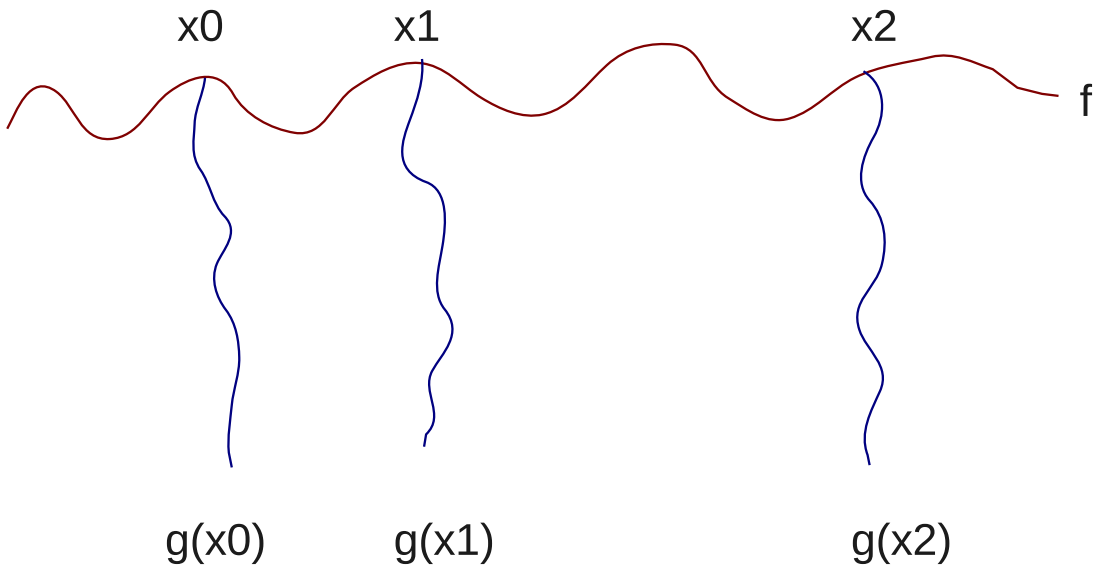


Figure: Schematic of $f >x> g$

Pruning: $f \ {<}x{<}\ g$

For some value published by g do f.

- Evaluate f and g in parallel.
  - Site calls that need x are suspended.
    Consider $(M() \mid N(x)) \ {<}x{<}\ g$
- When g returns a (first) value:
  - Bind the value to x.
  - Kill g.
  - Resume suspended calls.
- Values published by f are the values of $(f \ {<}x{<}\ g)$.

Example of Pruning

$$Email(address, x) \ <x< \ (CNN(d) \mid BBC(d))$$

Binds x to the first value from $CNN(d) \mid BBC(d)$.
Sends at most one email.

Fork-join parallelism

Call M and N in parallel.
Return their values as a tuple after both respond.

$$
\begin{array}{l}
((u, v) \\
\qquad <u< M()) \\
\qquad <v< N()
\end{array}
$$

Otherwise: f ; g

Do f. If f halts without publishing then do g.

- An expression halts if
  - its execution can take no more steps, and
  - all called sites have either responded, or will never respond.
- A site call may respond with a value, indicate that it will never respond (helpful), or do neither.
- All library sites in Orc are helpful.

Examples of $f \,;\, g$

$1 \,;\, 2$ publishes 1

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

If the sites are not helpful, this is equivalent to

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

Some Fundamental Sites

- $Ift(b)$, $Iff(b)$: boolean b,
  Returns a signal if b is true/false; remains silent otherwise.
  Site is helpful: indicates when it will never respond.

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

- *stop* : never responds. Same as $Ift(false)$ or $Iff(true)$.

- *signal* : returns a signal immediately.
  Same as $Ift(true)$ or $Iff(false)$.

Use of Fundamental Sites

- Print all publications of h. When h halts, publish "done".

  $$h \;>x>\; Println(x) \;\gg\; stop \;;\; "done"$$

- Timeout:
  Call site M.
  Publish its response if it arrives within 10 time units.
  Otherwise publish 0.

  $$x \;<x<\; (M() \mid Rwait(10) \;\gg\; 0)$$

Function Definition

$$\begin{array}{l} \mathit{def}\ \ \mathit{MailOnce}(a) = \\ \qquad \mathit{Email}(a,m)\ \ {<}m{<}\ (\mathit{CNN}(d) \mid \mathit{BBC}(d)) \end{array}$$

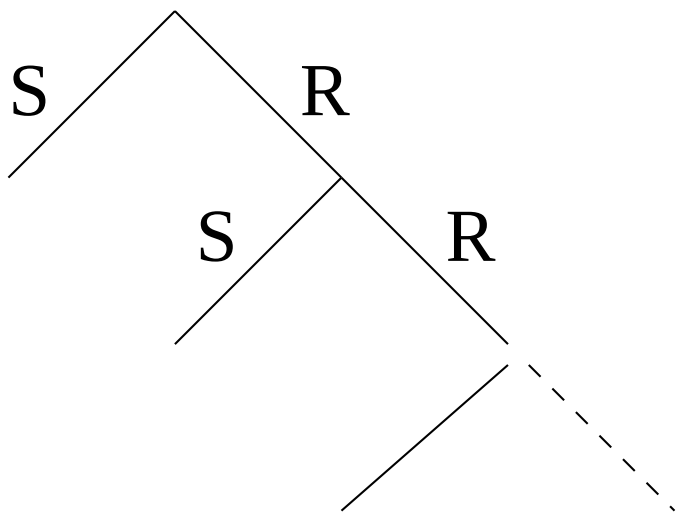
$$\begin{array}{l} \mathit{def}\ \ \mathit{MailLoop}(a,t) = \\ \qquad \mathit{MailOnce}(a)\ \gg\ \mathit{Rwait}(t)\ \gg\ \mathit{MailLoop}(a,t) \end{array}$$

- A function is called like a procedure.
  It may publish many values. *MailLoop* does not publish.
- Site calls are strict; Function calls non-strict.

Example of a Definition: Metronome

Publish a signal every unit.

$$def\ \ Metronome() = \underbrace{signal}_{S} \ \mid\ (\underbrace{Rwait(1) \gg Metronome()}_{R})$$



Example of Function call

- Site $Query()$ returns a value (different ones at different times).
- Site $Accept(x)$ returns x if x is an acceptable value;
  it is silent otherwise.
- Call $Query$ every second forever and publish all its acceptable values.

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

Concurrent function call

- Functions are often called concurrently.
- Each call starts a new instance of function execution.
- If a function accesses shared data, concurrent invocations may interfere.

Example: Publish each of "tick" and "tock" once per second, "tock" after an initial half-second delay.

$$\begin{array}{lll} & Metronome() & \gg\ ”tick” \\ | \ Rwait(500) \gg & Metronome() & \gg\ ”tock” \end{array}$$

Laws about $\mid$ and $\gg$

(Zero and $\mid$) $f \mid stop = f$

(Commutativity of $\mid$) $f \mid g = g \mid f$

(Associativity of $\mid$) $(f \mid g) \mid h = f \mid (g \mid h)$

(Associativity of $\gg$) if h is x-free

$(f >x> g) >y> h = f >x> (g >y> h)$

(Left zero of $\gg$) $stop \gg f = stop$

(Left unit of $\gg$) $signal \gg f = f$

(Right unit of $\gg$) $f >x> x = f$

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

$(f \mid g) >x> h = (f >x> h \mid g >x> h)$

Identities that don't hold

(Idempotence of $\mid$) $f \mid f = f$

(Right zero of $\gg$) $f \gg stop = stop$

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

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

Laws about $\ll$

(Right unit of $\ll$) $f \ll stop = f$

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

$$((f \mid g) \ <x< \ h) = (f \ <x< \ h) \mid g$$

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

$$((f \gg g) \ <x< \ h) = (f \ <x< \ h) \gg g$$

(Distributivity over $\ll$) if g is y-free and h is x-free

$$\begin{aligned} & ((f \ <x< \ g) \ <y< \ h) \\ = \quad & ((f \ <y< \ h) \ <x< \ g) \end{aligned}$$

(Elimination of $\ll$) if f is x-free, for site M

$$(f \ <x< \ M()) = f \mid (M() \gg stop)$$

Laws about ;

(Left unit of ;) $stop \ ; \ f = f$

(Right unit of ;) $f \ ; stop \ = f$

(Associativity of ;) $(f \ ; \ g) \ ; \ h = f \ ; \ (g \ ; \ h)$