A Compositional Trace Semantics for Orc

Dimitrios Vardoulakis Mitchell Wand

Northeastern University

May 27, 2008

What is Orc, what we did

- Orc is a process calculus for programming concurrent endpoints. Proposed by Jayadev Misra and William Cook (University of Texas at Austin).

What is Orc, what we did

- Orc is a process calculus for programming concurrent endpoints. Proposed by Jayadev Misra and William Cook (University of Texas at Austin).
- Small and simple, but effective. The workflow patterns of van der Aalst can be coded in Orc.

What is Orc, what we did

- Orc is a process calculus for programming concurrent endpoints. Proposed by Jayadev Misra and William Cook (University of Texas at Austin).
- Small and simple, but effective. The workflow patterns of van der Aalst can be coded in Orc.
- We showed that the previous trace semantics was not adequate and we fixed it.

Main Features of Orc

- Site Call: a request to an autonomous computing unit that does arithmetic, printing etc
- Operators to coordinate the execution of site calls e.g. parallel composition
- Recursive declarations to express non-terminating processes

Site Call

$$IsPrime(N)$$

$$Reddit(June\ 3)$$

Site Call

$$IsPrime(N)$$

$$Reddit(June\ 3)$$

A site returns a value by "publishing" it.

Symmetric Composition $(f \mid g)$

Evaluate f and g in parallel, no interaction between them.

$$IsPrime(N) \mid Reddit(June\,3)$$

Sequencing ($f >x> g$)

Start evaluating f. Each time it publishes, spawn a new instance of g in parallel.

$$(IsPrime(N) \mid Reddit(June\,3)) >x> Print(x)$$

Asymmetric Composition (f **where** $x :\in g$)

Evaluate f and g in parallel. When g publishes, it sends the value to f and terminates.

$$\mathit{Print}(x) \textbf{ where } x :\in (\mathit{IsPrime}(N) \mid \mathit{Reddit}(\mathit{June}\,3))$$

Mutually Recursive Declarations

We can express processes that don't terminate.

We define: $DOS(x) = Ping(x) \mid DOS(x)$

And then call: $DOS(ip)$

Syntax

Program $\quad P ::= D_1, \ldots, D_k \ \textbf{in} \ e$

Expression $\quad e ::= \mathbf{0} \mid M(p) \mid \mathit{let}(p) \mid E_i(p) \mid (e_1 \mid e_2) \mid e_1 >x> e_2 \mid e_1 \ \textbf{where} \ x :\in e_2$

Parameter $\quad p ::= x \mid v$

Declaration $\quad D_i ::= E_i(x) \triangleq e$

Operational Semantics

Labeled transitions of the form:

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

Process f takes a step to f' with observable event a:

$$\text{Event} ::= !v \mid \tau \mid M_k(v) \mid k?v$$

Atomic Processes

$$(\textsc{sitecall}) \quad \frac{k \text{ fresh}}{M(v) \xrightarrow{M_k(v)} ?k}$$

$$(\textsc{siteret}) \quad ?k \xrightarrow{k?v} let(v)$$

$$(\textsc{let}) \quad let(v) \xrightarrow{!v} \mathbf{0}$$

Some Composite Processes

$$(\textsc{sym2}) \qquad \frac{g \xrightarrow{a} g'}{f \mid g \xrightarrow{a} f \mid g'}$$

$$(\textsc{seq1v}) \qquad \frac{f \xrightarrow{!v} f'}{f >x> g \xrightarrow{\tau} (f' >x> g) \mid [v/x]g}$$

Some Composite Processes

$$(\textsc{asym1n})\ \frac{f \xrightarrow{a} f'}{f \textbf{ where } x :\in g \xrightarrow{a} f' \textbf{ where } x :\in g}$$

$$(\textsc{asym1v})\ \frac{g \xrightarrow{!v} g'}{f \textbf{ where } x :\in g \xrightarrow{\tau} [v/x]f}$$

What about free variables?

$$f \xrightarrow{[v/x]} [v/x]f$$

What about free variables?

$$f \xrightarrow{[v/x]} [v/x]f$$

$\surd$ $\qquad M(x) \mid let(x) \xrightarrow{[3/x]} M(3) \mid let(3)$

What about free variables?

$$f \xrightarrow{[v/x]} [v/x]f$$

$$M(x) \mid let(x) \xrightarrow{[3/x]} M(3) \mid let(3)$$

$$let(1) \xrightarrow{[2/y]} let(1)$$

What about free variables?

$$f \xrightarrow{[v/x]} [v/x]f$$

$$M(x) \mid let(x) \xrightarrow{[3/x]} M(3) \mid let(3)$$

$$let(1) \xrightarrow{[2/y]} let(1)$$

$$\frac{let(x) \xrightarrow{[3/x]} let(3)}{M(x) \mid let(x) \xrightarrow{[3/x]} M(x) \mid let(3)}$$

Traces

- Traces defined in an operational way
- The sequence of zero or more transitions is an *Execution*
- Remove the τ's and you get a *Trace*

For example,
$\tau\,!1$ is an execution of $let(1) >x> let(x)$
$!1$ is a trace of $let(1) >x> let(x)$

Compositionality

Let $\langle f \rangle$ denote the trace set of a process f.
They overload the operators to work on trace sets and claim:

- $\langle f \mid g \rangle = \langle f \rangle | \langle g \rangle$
- $\langle f >x> g \rangle = \langle f \rangle >x> \langle g \rangle$
- $\langle f \textbf{ where } x :\in g \rangle = \langle f \rangle \textbf{ where } x :\in \langle g \rangle$

But there are a few problems, e.g.

$\langle let(2)\rangle >x> \langle let(x)\rangle \neq \langle let(2) >x> let(x)\rangle$

($[1/x]\,!1$ in lhs, not in rhs)

$\langle let(x)\rangle$ **where** $x :\in \langle \mathbf{0}\rangle \neq \langle let(x)$ **where** $x :\in \mathbf{0}\rangle$

($[2/x]\,!2$ in lhs, not in rhs)

Our Fix: Operational Semantics

Restrict the way receive events happen:

- Add an environment for variables in the operational semantics
- A process f can't take a $[v/x]$ step if x is not free in f

$$\Delta, \Gamma \vdash f \xrightarrow{a} f'$$

Our Fix: Operational Semantics

For example,

$$\frac{\Delta, \{(x, 1)\} \vdash \ \mathit{let}(x) \xrightarrow{[1/x]} \mathit{let}(1)}{\Delta, \{(x, 1)\} \vdash \ \mathit{let}(x) \mid M(x) \xrightarrow{[1/x]} \mathit{let}(1) \mid M(x)}$$

Our Fix: Denotational Semantics

Define the traces using complete partial orders:

$$[\![f]\!] : [Fenv \longrightarrow [Env \longrightarrow P]]$$

A few equations

$$[\![M(v)]\!] = \lambda\varphi.\lambda\rho.\{\, M_k(v)\ k?w\ !w \mid k \text{ fresh}, w \in Val\}_{\mathrm{p}}$$

$$[\![h \mid g]\!] = \lambda\varphi.\lambda\rho.\ [\![h]\!]\varphi\rho \parallel [\![g]\!]\varphi\rho$$

Adequacy

Our trace semantics is compositional by definition, but we must prove its adequacy with respect to the operational semantics:

$$t \in [\![f]\!][\![\Delta]\!]\rho \quad \text{iff} \quad \exists f'.\, \Delta, \Gamma \vdash \; f \; \xrightarrow{t}{}^{*} \; f'$$

Improvements over the previous semantics

- previous treatment of free variables is fragile
  - processes take extraneous receive steps
  - global rule in operational semantics

Improvements over the previous semantics

- previous treatment of free variables is fragile
  - processes take extraneous receive steps
  - global rule in operational semantics
- the traces of recursive definitions were defined by example before

Improvements over the previous semantics

- previous treatment of free variables is fragile
  - processes take extraneous receive steps
  - global rule in operational semantics
- the traces of recursive definitions were defined by example before
- plus a few new results

Bisimulation

- Strong bisimulation in Orc is a congruence relation.

Bisimulation

- Strong bisimulation in Orc is a congruence relation.
- Useful congruences:

$$f \mid (g \mid h) \sim (f \mid g) \mid h$$

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

$$(f \mid g) \textbf{ where } x :\in h \sim (f \textbf{ where } x :\in h) \mid g \qquad \text{if } x \notin \mathrm{fv}(g)$$

Bisimulation

- Strong bisimulation in Orc is a congruence relation.
- Useful congruences:

  $f \mid (g \mid h) \sim (f \mid g) \mid h$

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

  $(f \mid g)$ **where** $x :\in h \sim (f$ **where** $x :\in h) \mid g$ if $x \notin \mathrm{fv}(g)$
- Bisimulation is more discriminating than trace equivalence.

What to remember about Orc

- Abstracts computation away, focuses on communication
- Small but expressive
- Our semantics helps to reason effectively about Orc programs

Related Work

- Misra and Cook. "Computation Orchestration: a basis for wide-area computing", Software and Systems Modeling (2007)
- Kitchin, Cook and Misra. "A language for task orchestration and its semantic properties", CONCUR (2006)
- van der Aalst et al."Workflow Patterns", Distributed and Parallel Databases (2003)
- Cook, Patwardhan and Misra. "Workflow patterns in Orc", COORD (2006)

The End

Thank you!

dimvar@ccs.neu.edu