Orc Verification

Jayadev Misra

Department of Computer Science
University of Texas at Austin

http://orc.csres.utexas.edu

WG 2.3, St. Petersburg
June 5, 2013

Orc Verification has been a disaster

- Concurrency everywhere
- Non-determinacy
  As powerful as any other process calculus
- Real time
  Not just causal ordering among events but temporal ordering
- Basic orc has no mutable variables, but sites do
- Full functional programming (w/o monads)
  plus (active) Objects.

Subset Sum

Given integer n and list of integers xs.

$parsum(n, xs)$ publishes all sublists of xs that sum to n.

`parsum(5,[1,2,1,2]) = [1,2,2], [2,1,2]`

`parsum(5,[1,2,1])` is silent

$$def\ \ parsum(0, [\,]) = [\,]$$

$$def\ \ parsum(n, [\,]) = stop$$

$$\begin{aligned} &def\ \ parsum(n, x : xs) = \\ &\quad parsum(n - x, xs)\ \ >ys>\ x : ys \\ &\quad |\ parsum(n, xs) \end{aligned}$$

Subset Sum (Contd.), Backtracking

Given integer n and list of integers xs.

$seqsum(n, xs)$ publishes the first sublist of xs that sums to n.

"First" is smallest by index lexicographically.

`seqsum(5,[1,2,1,2]) = [1,2,2]`

`seqsum(5,[1,2,1])` is silent

$$def \;\; seqsum(0, [\,]) = [\,]$$

$$def \;\; seqsum(n, [\,]) = stop$$

$$\begin{aligned} def \;\; & seqsum(n, x : xs) = \\ & \quad x : seqsum(n - x, xs) \\ & \quad ; \; seqsum(n, xs) \end{aligned}$$

Subset Sum (Contd.), Concurrent Backtracking

Publish the first sublist of xs that sums to n.

Run the searches concurrently.

$$def \;\; parseqsum(0, [\,]) = [\,]$$

$$def \;\; parseqsum(n, [\,]) = stop$$

$$\begin{aligned} &def \;\; parseqsum(n, x : xs) = \\ &\quad (p \;;\; q) \\ &\qquad <p< x : parseqsum(n - x, xs) \\ &\qquad <q< parseqsum(n, xs) \end{aligned}$$

Note: Neither search in the last clause may succeed.

Semantics

- Tree semantics with Hoare
- Operational semantics with Cook
  1. Traces
  2. Bisimulation can be applied to prove some identities.
  3. Denotational Semantics was difficult. But, it established that:

     Orc combinators are monotonic and continuous.
- But, operational semantics seems ineffective for program proving.
- I failed in applying axiomatic semantics.

A sequence of Verification Problems

- Basic Orc without mutable variables, real time
- add real time
- add mutable variables
- Full Orc language

Denotational semantics with composable proof theories

$$
\begin{array}{lcllcl}
[\![f \mid g]\!] & \triangleq & [\![f]\!] \mid [\![g]\!] & & & \\
[\![f \; >x> \; g]\!] & \triangleq & [\![f]\!] \; >x> \; [\![g]\!], & [\![f \gg g]\!] & \triangleq & [\![f]\!] \gg [\![g]\!] \\
[\![f \; <x< \; g]\!] & \triangleq & [\![f]\!] \; <x< \; [\![g]\!], & [\![f \ll g]\!] & \triangleq & [\![f]\!] \ll [\![g]\!] \\
[\![f \, ; \, g]\!] & \triangleq & [\![f]\!] \, ; \, [\![g]\!] & & &
\end{array}
$$

Simple Expressions

- 1 publishes just 1: $\{1\}$
- $1 \mid 2$ publishes 1 and 2 in either order: $\{1, 2\}$
- $1 \mid 1$ publishes 1 and 1: $[1, 1]$

Publications are unordered.

Simple Expressions

- 1 publishes just 1: $\{1\}$
- $1 \mid 2$ publishes 1 and 2 in either order: $\{1, 2\}$
- $1 \mid 1$ publishes 1 and 1: $[1, 1]$

Publications are unordered.

Simple Expressions

- 1 publishes just 1: $\{1\}$
- $1 \mid 2$ publishes 1 and 2 in either order: $\{1, 2\}$
- $1 \mid 1$ publishes 1 and 1: $[1, 1]$

Publications are unordered.

A possible denotation of expressions

- Represent an expression by a bag of values.
- $|$ combines two bags.
- Bags may be infinite.

$$\mathit{def}\ \ \mathit{nat}(i) = i \mid \mathit{nat}(i+1)$$

$$\mathit{def}\ \ \mathit{nats}() = \mathit{nat}(0)$$

$$[\![\mathit{nats}()]\!] = [0, 1, \cdots]$$

- Computation may be infinite without any publication.

$$\mathit{def}\ \ \mathit{unend}() = \mathit{signal} \gg \mathit{unend}()$$

$$[\![\mathit{unend}()]\!] = [\,]$$

Bags are not enough

$$[\![\mathit{stop}]\!] = [\,]$$

$$[\![\mathit{unend}()]\!] = [\,]$$

But their behaviors are different:

$$\mathit{stop} \ ; 3 \neq \mathit{unend}() \ ; 3$$

Halting, Waiting

Associate a status, H for halting, W for waiting, to each bag.

$$[\![stop]\!] = H[\,]$$

$$[\![unend()]\!] = W[\,]$$

$$[\![1]\!] = H[1]$$

$$[\![1 \mid unend()]\!] = W[1]$$

$$[\![nats()]\!] = W[0, 1, \cdots]$$

Elementary term: A status and a bag.
The status of an infinite bag is always W.

Combining Elementary Terms with $|$

$$s[m] \mid s'[m'] = (s \cap s')[m \sqcup m']$$

where $H \cap s = s, \;\; W \cap s = W$

$$[\![1 \mid true]\!] = H[1] \mid H[true] = H[1, true]$$

$$[\![1 \mid stop\,]\!] = H[1] \mid H[\,] = H[1]$$

$$[\![1 \mid unend()]\!] = H[1] \mid W[\,] = W[1]$$

$$[\![nats() \mid nats()]\!] \neq [\![nats()]\!]$$

Specification

A specification, spec, is a set of terms, possibly infinitely many.

$$[\![Random(3)]\!] = \{H[0], H[1], H[2]\}$$

$$\begin{aligned} & [\![Random(3) \mid true \mid false]\!] \\ = \; & \{H[0, true, false], H[1, true, false], H[2, true, false]\} \end{aligned}$$

$$[\![anynat()]\!] = \{H[i] \mid \text{natural } i\}$$

Combining specs using $|$

$|$ distributes over each argument set. Take Cartesian product.

- $\{s_0, \cdots s_i, \cdots\} \mid \{t_0, \cdots t_i, \cdots\} = \{(s_0 \mid t_0), \cdots (s_i \mid t_j), \cdots\}$

Guarded Term

- $b \rightarrow s[m]$:
  the set of traces in which the bindings satisfy b and the status and publications satisfy $s[m]$.
- Taking $|$ over guarded terms:
  $$b \rightarrow s[m] \mid b' \rightarrow s'[m'] = (b \wedge b') \rightarrow (s \cap s')[m \sqcup m']$$
- Guards distribute over terms in a spec:
  $$b \rightarrow \{t_0, t_1 \cdots\} = \{b \rightarrow t_0, b \rightarrow t_1 \cdots\}$$

Parameters; Guarded terms

- $[\![not(x)]\!] = \{x = true \rightarrow H[true],\ x = false \rightarrow H[false]\}$
- $[\![x]\!] = \{x = c \rightarrow H[c] \mid \text{for all } c\}$
- $[\![Ift(x)]\!] = \{x = true \rightarrow H[signal],\ x \neq true \rightarrow H[\,]\}$

Often a parameter is known to remain unbound, denoted by $\not!$

$$[\![x]\!] = \{x = \not! \rightarrow H[\,]\} \cup \{x = c \rightarrow H[c] \mid \text{for all } c\}$$

Example

$$[\![Ift(x)]\!] = \{x = true \rightarrow H[signal],\ x \neq true \rightarrow H[\,]\}$$
$$[\![Iff(x)]\!] = \{x = false \rightarrow H[signal],\ x \neq false \rightarrow H[\,]\}$$

$$\begin{aligned} & [\![Ift(x) \mid Iff(x)]\!] \\ = \quad & \{(x = true \wedge x = false \rightarrow \cdots) \\ & ,(x = true \wedge x \neq false \rightarrow H[signal]) \\ & ,(x \neq true \wedge x = false \rightarrow H[signal]) \\ & ,(x \neq true \wedge x \neq false \rightarrow H[\,])\} \end{aligned}$$

$$\begin{aligned} = \quad & \{(x = true \vee x = false \rightarrow H[signal]) \\ & ,(x \neq true \wedge x \neq false \rightarrow H[\,])\} \end{aligned}$$

Notation

Convention:

$s[\cdots f(x,y)\cdots] \;\underline{\Delta}\; \{x = c \wedge y = c' \rightarrow s[\cdots f(c,c')\cdots] \mid \forall c, c'\}$,
for any total function f that is strict in all its arguments.

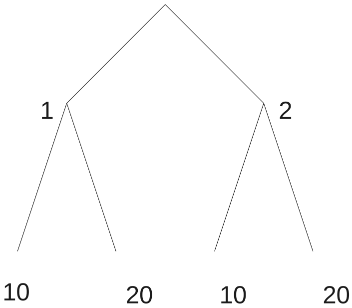
- $[\![choose(x,y)]\!] = \{H[x], H[y]\}$

- $$\begin{aligned} &[\![parallel\_or(x,y)]\!] \\ = \{&(x = true \rightarrow H[true]), \\ &(y = true \rightarrow H[true]), \\ &H[x \vee y]\} \end{aligned}$$

- $(x = true \rightarrow H[true]),\ (y = true \rightarrow H[true])$
  is not the same as
  $(x = true \vee y = true \rightarrow H[true])$
  The first line is satisfied even if just one of x and y is bound.

Sequential Composition

$(1 \mid 2) \gg (10 \mid 20)$: Execute the rhs for every publication of lhs



Should have the spec $H[10, 20, 10, 20]$, construced from $H[1, 2]$ and $H[10, 20]$.

Sequential Composition; contd.

In $s[m] \gg q$, for every value in m one instance of a program with spec q is executed. All such programs are executed in parallel.

Tentative Rule: $s[m] \gg q = \mid [q \mid c \sqsubseteq m]$

$$
\begin{aligned}
& H[1,2] \gg H[10,20] \\
& = \{\text{Tentative Rule}\} \\
& H[10,20] \mid H[10,20] \\
& = \\
& H[10,20,10,20]
\end{aligned}
$$

For $W[1,2] \gg H[10,20]$, the result should be $W[10,20,10,20]$

Exact Rule:

$$
\begin{aligned}
& s[m] \gg q = s[\,] \mid [q \mid c \sqsubseteq m] \\
& p \gg q = \cup_{t \in p}(t \gg q)
\end{aligned}
$$

Sequential Composition with value passing

$$[\![(1 \mid 2) \;\; >x> \; (10 + x \mid 20 - x)]\!] = H[11, 12, 18, 19]$$

Rule:

$$\begin{array}{lll} s[m] \;\; >x> \; q & = & s[\,] \mid [(x \mapsto c)q \mid c \sqsubseteq m] \\ p \;\; >x> \; q & = & \cup_{t \in p}(t \;\; >x> \; q) \end{array}$$

$$\begin{array}{l} H[1,2] \;\; >x> \; H[10 + x, 20 - x] \\ = \\ H[\,] \mid (x \mapsto 1)H[10 + x, 20 - x] \mid (x \mapsto 2)H[10 + x, 20 - x] \\ = \\ H[\,] \mid H[10 + 1, 20 - 1] \mid H[10 + 2, 20 - 2] \\ = \\ H[11, 19, 12, 18] \end{array}$$

Exercise: $[\![nats() \;\; >x> \; x * x]\!]$

Pruning

$$[\![(x \ {<}x{<}\ (1 \mid 2)]\!] = \{H[1], H[2]\}$$

Rule:

$$\begin{aligned} p \ {<}x{<}\ s[m] &= \cup_{(c \in m)}((x \mapsto c)p) \\ p \ {<}x{<}\ q &= \cup_{(t \in q)}(p \ {<}x{<}\ t) \end{aligned}$$

$$[\![i \ {<}i{<}\ nats()]\!]$$

$$=$$

$$H[i] \ {<}i{<}\ W[0, 1, \cdots]$$

$$=$$

$$\cup_{(c \in [0,1,\cdots])}((i \mapsto c)H[i])$$

$$=$$

$$\{H[0], H[1] \cdots\}$$

Recursive Definition

$$\textit{def} \;\; \textit{nat}(i) = \; i \mid \textit{nat}(i+1)$$

$$\textit{def} \;\; \textit{nats}() = \; \textit{nat}(0)$$

Ordering over terms

- $t \leq t$
- $(b \rightarrow W[m]) \leq (b' \rightarrow s[m'])$ if $b' \Rightarrow b, \; m \sqsubseteq m'$

$W[\,]$ is the smallest term.

Prefix closure; Spec Ordering

Define:

- $t^* = \{s \mid s \leq t\}$
- $p^* = \cup_{t \in p}(t^*)$
- $p \leq q \ \underline{\Delta}\ p^* \subseteq q^*$
- $p \equiv q \ \underline{\Delta}\ (p \leq q) \wedge (q \leq p)$
  So, $(p \equiv q) = (p^* = q^*)$

Monotonicity, Continuity

- Every combinator is monotonic in each argument.
- Every combinator is continuous in each argument.
  Take the lub of a chain of specs to be the union of their closures.

Extensions

- Real time needs a surprisingly simple extension.
- Yet to be done: Mutable sites, Orc language