Mon Nov 29, 2004

Locations
- reply?
- lookup/apply metaobject
- event log
- parent?

Methods: mobile λ-terms? (closures)

State: on ports (?) - wellformedness check making sure <u>exactly</u> one reachable restate
- if allow oneshots, zero/one restate. or consistent zero vs. consistent one.

metaop required for I/O crossing location boundaries
- put in comm rule?

Continuations: ports or full method objects?

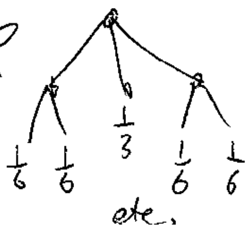
Reply. Is this related to Malenfant et al. reflection over the continuation?
- either direct output on a port OR
- reply via dispatch mechanism

Reply could lift locations back to the receiver
→ aborts in-progress computation
collection select: [:a | a isEven ifTrue: [^ a]]
Also has benefit of clearly disambiguating the reply operator. "^" would mean "reply to nearest lexically enclosing location". Of course, block invocation would then have to <u>not</u> create a fresh location, whereas method invocation would

Upward funargs - after return, what would "^" do? only really sensible for located funargs
how about partial continuations? → actually equivalent to invoking captured continuation
↓
shift/reset etc.

Location of methods - where does #select: run?

Scheduling done via a tree. of locations
– fair roundrobin at each node?


$\frac{1}{6}$ $\frac{1}{6}$ $\frac{1}{3}$ $\frac{1}{6}$ $\frac{1}{6}$
etc.

I/O on state ports, lifts for reply
– both must be customised for location-based RCHAM. + for transactional logging.

and method invocation (= λ application.)!

~~νk.(update! k(6 [illegible]~~

$\nu k.(\, update[\langle k,6\rangle] \mid k[(old).P]\,)$
$\mid update[!(k,new).\, state[(old).(\langle new\rangle \mid k[\langle old\rangle])]]$
$\mid fetch[!(k).\, state[(old).(\langle old\rangle \mid k[\langle old\rangle])]]$

} Stateful invocation
update: 6.

Stateless : $1 + 2 \rightarrow 1$ plus: 2 .
$\rightarrow$ {~~#~~plus: . 1. 2}

$\nu k.\ plus{:}[\langle k,1,2\rangle] \mid k[(sum).P]$
$\mid plus{:}[!(k,a,b).\ primIntAdd[\langle k,a,b\rangle]]$

Locations – have state? if paused/lifted, then processes arriving at the location are lifted and delivered to the lift receiver? metaprotocol?

better example 1 plusTwo.

$\nu k.(\ plusTwo{:}[\langle k,1\rangle] \mid k[(sum).P])$
$\mid plusTwo{:}[!(k,a).\ plus{:}[\langle k,a,2\rangle]]$

Mon Nov 29 2004

$\boxed{2}$

Want some way of making the λ 'unpeel' a banged copy into the calling location.

1 times4plus2.

similarity to state binder. state output similar to 'reply'

$\nu k_1 . (\text{times4plus2}[<k_1, 1>] \mid k_1[(\text{sum}).P])$
$\mid \text{times4plus2}[!(k, x). \nu k_2 . \text{times}[<k_2, x, 4>] \mid k_2[(v). \text{plus}[<k, v, 2>]]]$

$\ldots \mid \text{times4plus2}[!(k,x).k[\nu k_2 . *[<k_2, x, 4>] \mid k_2[(v). +[<k, v, 2>]]]]$

$\text{alwaysFour}[!(k).k[<4>]]$

$[\![\lambda(\tilde{y}).P]\!]_k \Rightarrow \nu m . m[!(n, \tilde{y}). n[[\![P]\!]_n]] \mid k[<m>]$

$[\![\lambda(\varsigma = E \mid \tilde{y}).P]\!]_k \Rightarrow \nu\sigma . [\![E]\!]_\sigma \mid$ $\nu m . m[!(n,\tilde{y}). \sigma[(\varsigma). n[[\![P]\!]_n]]]$
$k[<m>]$

$[\![\varsigma \leftarrow E]\!]_k \Rightarrow [\![E]\!]_\sigma$

$[\![\tilde{V}]\!]_k \Rightarrow k[<\tilde{V}>]$

$[\![\cdot]\!]_N^{\{N \times N\}} : E \to P$

What is ~~the~~ or a π encoding of delimited continuations?

The Next Big Thing

Tu 30 Nov 04

Reflection vs Metaprogramming
Transactions (Henry Baker)
Concurrency
Functional (Mostly)
Object-oriented (PMD)
Distributed
Malenfant's reflection — no — infinite push-down-list!

Self — speed, simplicity, flexibility
- uniform approach: everything is an object (including activation contexts, in which local variables are slots)
- concurrency (apparently) poorly integrated
- FAST fib benchmark
- may be a semantics: haven't seen it if so? → research topic!

Slate — Almost ideal
- The more I learn about it the closer to ideal it seems. eg transactions a la Baker.
- PMD
- Futures for concurrency lifted from E
  - more like promises, in a way
  - resolution needed for dispatch
- no formal model as yet, no semantics

Ambients — close to ideal for Txns, Concur, Distributed
- routing is an issue
- silent on Functional, OO, Reflection
- Adding lift & drop allows definition of ambient operators (conjecture)

Reflection – for debugging
– for exceptions
– for transaction rollbacks
– for distributed programming
– requires a semantics ?

Efficiency – poor to begin with, fully interpreted
– method caching needs adapting to PMD –
(ask on #slate about numbers from their expts.)
– other self compilation techniques applicable
– cartesian product algorithm etc
– immutability of objects a challenge for representing metadata

Should support #become: etc?

~~[illegible]~~ "A synthesis of dynamic language ideas"
– Thesis topic