Monte: A Spiritual Successor to E

Corbin Simpson

Matador Cloud LLC

October 24, 2017

Overview

- Where did we come from?
- Why are we here?
- Where are we headed?

What is E?

E is a family of lambda calculi with:

- local state
- mutable closures
- managed runtime with:
  - garbage collector, promises, safe scope
  - compiler/interpreter tools
  - communicating event loops ("vats")

Timeline

- 1993: Original-E
- 1996: Joule
- 1997: E-on-Java
- 2004: Joe-E
- 2005: E-on-CL
- 2008: Caja, Ecru
- 2012: Experiments (Greyface, Secret)
- 2013: “Original-Monte”
- 2014: Monte-on-Typhon

Whither E?

Knowledge of E is relatively rare. Why? We can only guess:

- People were not ready for E in the 90s
- Capability-security theory remains unpopular
- October 2016: Winter finally arrived

Named Arguments

Monte messages have one additional parameter.

```
[verb :Str, args :List, namedArgs :Map]
```

This enables named arguments, including:

- Optional arguments
- Miranda named arguments

Sealed Exceptions

Exceptions cannot be examined without `unsealException`!
We can deliver errors to only two places:

- The caller, via ejectors
- The debugger, via exceptions

```
try { throw(42) } catch p { traceln.exception(p) }
```

Changes to Safe Scope

safeScope Objects which have no dangerous authority, because they either can be synthesized from scratch, or come from the runtime's TCB (e.g., `eval`)

- Many gratuitous renamings
- Adjustments for changes to Kernel-Monte
- Iterators
- `Int` guards the ring of integers
- b‘‘ and `Bytes` for bytestrings
- `makeLazySlot` to work around lack of `EventuallyDeepFrozen` for the common case of wanting lazy private state
  - Perhaps memoization is the only other interesting use case?

Iterators