Embracing Heterogeneity

ROSS TATE

64-bit machines

- 48-bit address space (256 terabytes)
  - varies by system, but using as concrete example
- Leaves 16 bits free!
  - 3 more with standard 8-byte alignment
- With NaN boxing
  - 53 bits free
- Use 1 bit to distinguish refs from non-refs
  - 52 bits free for non-address values
  - 6 bits free for address values
- Can still GC without interpreting these bits
- How to use them?

Kotlin

- Non-References
  - Type Index (Integer, Float, Character, Short, Byte, Boolean) + 32-bit value
    - Type index is used to index into special primitive v-table array
- References
  - Small Enums (5 bits for value + reference to v-table)
  - Hard-Coded Reference Types (no v-table?): String, Double, Long, arrays
  - Other

OCaml

- Non-References
  - Algebraic Data Type Tag (19 bits) + 32-bits of primitive values
    - Primitive Types: bool, char, int, Int32
    - nil, none
- References
  - float
  - tuple, record/ref, Array
  - Closure
    - Use spare bits to specify arity
  - Algebraic Data Type
    - Use spare bits for common cases: cons, some

Scheme

- Non-References
  - boolean, character, float32, i32, void, undefined
  - keyword, symbol
- References
  - Numeric: float64, i64
  - String, Byte String, Regular Expression
  - Pair, List, Mutable Pair, Mutable List, Vector, Box, Hash Table, Sequence, Stream, Dictionary, Set
  - Interface, Class, Object, …
  - Procedure, Continuation, Thread, Channel, Semaphore …
  - Impersonator, Security Guard, …

Different Languages, Different Needs

- Kotlin packed primitives need v-table index
- Kotlin enums need reference to v-table
- OCaml ADTs need type tag
- Ocaml needs walkable values for structural equality
- Scheme needs “numeric” bit tag for fast eqv?
- Scheme needs “impersonator” bit tag for fast common case

Same bits within pointer

~~anyref as top~~

Different meaning for bits