Minimal Change to Nominalize the MVP

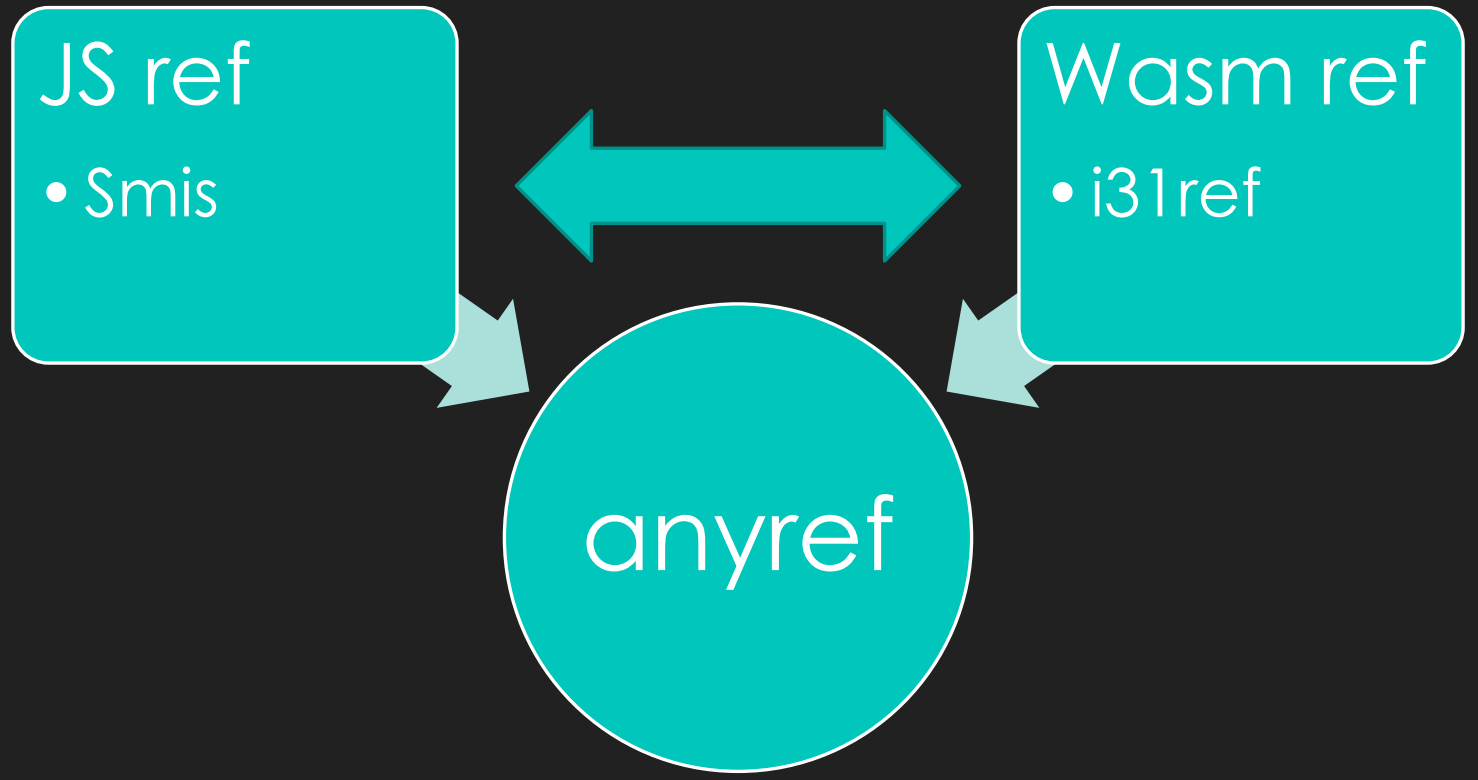
Ross Tate

Low-Level Systems

int (*)(int)

$[i32] \xrightarrow{A} [i32]$	$[i32] \xrightarrow{B} [i32]$	$[i32] \xrightarrow{C} [i32]$
input/output in register r5	input/output in register r3	input/output on stack

Low-Level Conflicts



Centralizing Coordination

- WebAssembly already requires module centralization within an application
  - All modules needs within an application need to use/import the same exception event
- Exception events are nominal for composability
  - Do not want exception events of different systems to interfere
- Nominal heap types provide composability
  - Representation/casting constraints of different systems do not interfere
- Since application-central module instance is already necessary
  - Use it to coordinate types/RTTs for primitive reference types
- High-level modules are coordinated by the language runtime
  - In WebAssembly, that language runtime is now its own module, the application-central one

Understanding rtt.canon (and rtt.sub)

- rtt.canon $type : [] -> [(rtt $type)]
  - Represent $type as an automaton
  - Minimize that automaton
  - Canonicalize that minimization
  - Hash that canonicalization
  - Look for preexisting rtt for that type in process-global hashtable
  - If $type uses an imported type, do this per instantiation
  - Post-MVP extends with parametric polymorphism
    - Rtts for type arguments are supplied at run time
    - Above process is done at run time (e.g. per call of polymorphic function constructing pairs in Post-MVP)

Concerns with Structural Typing

- Unknown how to scale to major languages (#116)
  - Whether Post-MVP has practical decidable restriction is longstanding open research problem
  - Post-MVP cast mechanisms cannot express, for example, Java/C# arrays
- Concerns about compile-time performance (#117)
  - Low-level structural types are large, and structural subtyping is quadratic
  - Parametric polymorphism would reduce applicability of memoization
- Issues with composing implementations
  - Unclear if it can be extended with pointer-tagging or unboxed nullary variants (#118)
  - i31ref conflicts with Smis in externref if both are subtypes of anyref (#76/#130)

Observations

- Bottlenecked by dynamic type system
  - Every “this” pointer must be cast
  - Elements of arrays must be cast
- Dynamic type system is nominal
  - RTTs have distinct identities and explicit hierarchy
- Low-level nominal types…
  - have been demonstrated to scale to major languages and existing compilers
  - benefit from fast compile-time and run-time subtyping
  - avoid clashing distinct low-level systems

Suggestion

- Specify hierarchy in types section
  - make heap types generative rather than applicative (i.e. two defs always makes two types)
  - make "sub" relation explicit in type declarations rather than deduced through equi-recursion
  - remove anyref
- Possibly
  - Remove rtt type
  - Separate rtt into GC descriptors (i.e. object headers) and type identifiers (i.e. magic numbers)
  - Enrich GC descriptors to store other meta-data, e.g. v-table