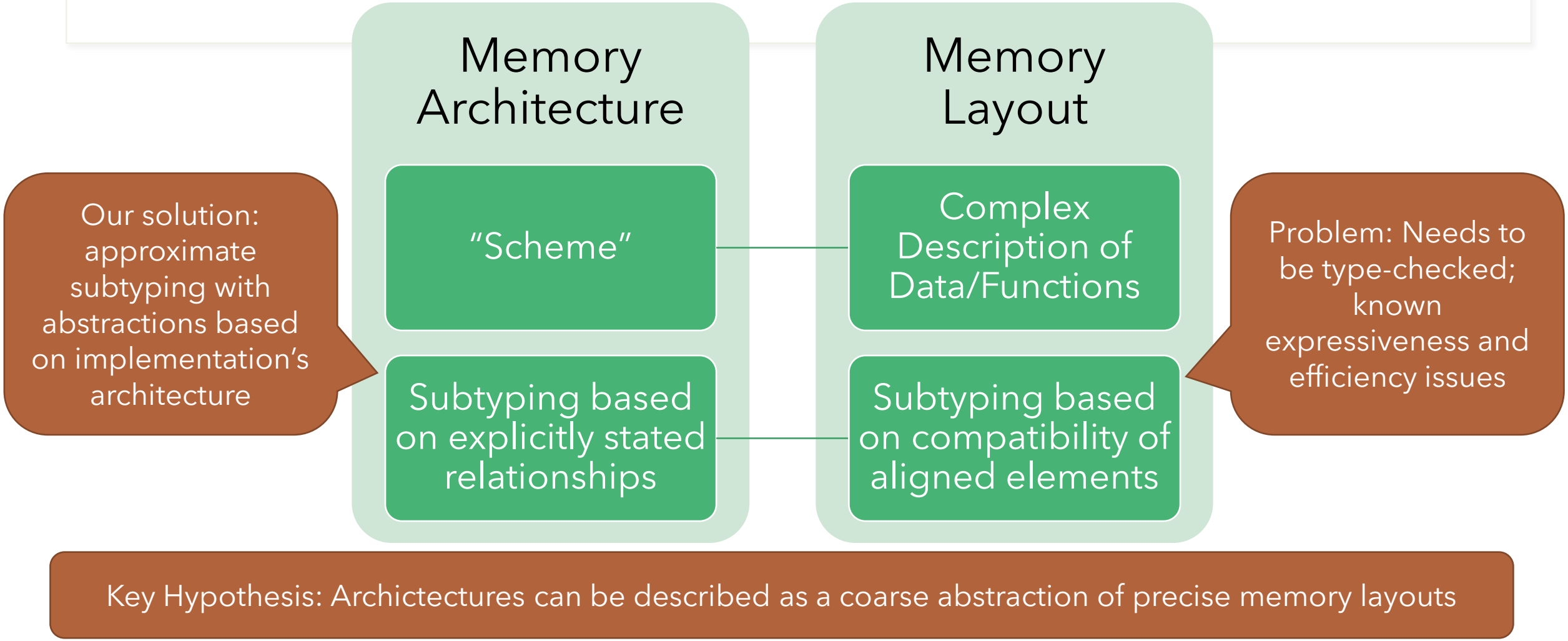
Overview of SOIL Initiative's GC Proposal

Ross Tate

High-Level Patterns



Key Hypothesis: Archictectures can be described as a coarse abstraction of precise memory layouts

Low-Level Representations

- Problem: wasm module has no control over low-level representation of pointers and casting mechanisms
  - Varying hardware architectures, e.g. 32 vs. 64 bit
  - Varying GC implementations: e.g. summaries vs. bit flags
- Our solution: provide enough information for engine to specialize low-level representation for individual modules
  - E.g. immutable i32 w/o identity can be packed on 64-bit machine
  - E.g. cast with bit flags vs. summary equality vs. array lookup

Key Hypothesis: Specialized representations mproves performance of various language runtimes

Modularity and Abstraction

- Key Hypothesis: module coordination is intentional/explicit
  - e.g. references from unrelated modules need not be castable
    - Java modules do not know how to cast Ocaml refs and vice-versa
- Goal: surface-level equivalence implies wasm equivalence
  - e.g. private Java fields should be invisible
  - e.g. public Java fields should be reorderable
  - e.g. names of Java fields should be irrelevant (ignoring reflection)
  - How to achieve this?

Ideas being explored

- Support for engines with simple memory management
  - Incorporate acyclicity for reference counting
  - Invalidatable references for resource release w/o even ref counting
- Async coop between module-manage lin. mem. and host gc
- Eliminating superfluous casts (existential/universal types)
- Interior pointers and nested structures
- Stack-allocated memory (needs to align with heap-alloc mem)
- Complex/cross-module initialization of structures