Topics Laundry List

CMSC22620/23620, Spring 2007

March 23, 2007

• type checking, recursive definitions, symbol tables
  – review of Hindley-Milner type inference (algorithm $W$ and its variations)
  – other type systems
    * subtyping
    * co-/contra-variance
    * runtime safety issues

• representation of values
  – scalars
  – records and tuples
  – sum types (enumerations, lists, trees, ...)
  – function closures
  – Pascal records or C structs
  – C unions
  – objects

• stack vs. heap vs. static allocation

• intermediate language(s): lambda, ANF, trees, etc.
  – high-level optimizations
  – tail recursion and loops
- making things explicit (closure conversion, etc.)
- canonicalization (on trees)

• runtime environment
  - calling conventions
    * parameter-passing conventions
    * result-passing conventions
    * activation records and stack layout
  - more stack organization
    * frames
    * without nested functions
    * with nested functions
    * with local variables that “outlive” their function invocation (e.g., because of an address-of operator or first-class functions)
    * with objects and methods

• machine instructions
  - CISC vs. RISC
  - registers
  - addressing modes

• instruction selection
  - “maximal munch”
  - dynamic programming

• basic blocks and traces

• simple local optimizations (e.g., intra-block value numbering)

• flow analysis
• global (flow-based) optimizations
• loops and dominators, loop-based optimizations
• other intermediate languages
– static single assignment (SSA) form
– continuation-passing style (CPS)
– inter-procedural optimizations

• pipelining and scheduling
• branch prediction
• liveness analysis
• (graph-coloring) register allocation
  – with coalescing
• generating assembly code
• garbage collection