CS131: Programming Languages

Lun Liu

06.09.17

Final Review

- OCaml
  - recursion, higher-order functions, datatypes, pattern matching, exceptions
  - static vs. dynamic typechecking
    - compile time vs execution time
  - static vs. dynamic scoping
    - lexical context vs calling context
  - parametric polymorphism vs static overloading
    - one function vs many functions with same name (different type signature)
  - strong vs. weak typechecking
    - variable type cannot change vs loopholes in type system to work around

Final Review

- **Java**
  - subtyping, inheritance
    - inheritance: code reuse
  - parametric polymorphism (generics)
    - stronger type checks at compile time
  - memory, aliasing
    - parameter passing: still by value, but the value is a “pointer” (“reference”)
  - static overloading vs dynamic dispatch
    - compile time vs runtime
    - **type signature**
  - exceptions
    - **throws** annotation
  - parallelism: fork/join, streams

Final Review

- Prolog
  - unification
    - takes two terms t1 and t2 and returns either NO or an environment (mapping variables to terms) that makes t1 and t2 syntactically identical
  - proof trees
    - try build proof trees for problems yourself!

Thanks!

Don’t forget the course evaluation! Good luck with your exam! ☺

Midterm Question 4

- For each property of OCaml below, say whether it is a consequence of OCaml being statically typed (write “static”), strongly typed (write “strong”), both (write “both”), or neither (write “neither”).
- (a) OCaml treats functions as first-class values. ANSWER: neither
- (b) OCaml does not support static overloading. ANSWER: neither
- (c) OCaml never allows a primitive operation to be invoked with arguments of the wrong types. ANSWER: strong
- (d) OCaml never allows a user-defined function to be invoked with arguments of the wrong types. ANSWER: both
- (e) OCaml rejects some programs at compile time that would not incur any errors if allowed to execute. ANSWER: static