Lecture Notes 13

Polymorphic Type Checking

- Hindley-Milner Type Checking – Infer types by analyzing parse tree
  - Type Variables – Variable assigned to all types of unknown identifiers
  - Instantiation – Assignment of type for all instances of type variable
  - Unification – Replacement of one type variable with another
  - Let-bound Polymorphism – Requires separation of types into (at checking) and (at instantiation) categories
- Overloaded Functions – Names have several different types simultaneously
- Parametric Polymorphism
  - Implicit – Parameters of polymorphism are implicitly introduced
  - Explicit – Parameters of polymorphism are explicitly stated
  - Ad-hoc – Type of polymorphism used with overloading
- Pure or Subtype Polymorphism – Polymorphism used with objects with common ancestor
- Monomorphic – No polymorphism
- Machine Code Generation Issues
  - Expansion – Make copy of each polymorphic function (code bloat)
  - Boxing and Tagging – Single function that takes in a tag to determine type

Explicit Polymorphism

- User-defined Type Constructors – Explicit polymorphism is just another way to create user-defined types
- template (C++)
- Constrained Parametric Polymorphism – Puts a requirement on type used in polymorphism
  - Implicit – The requirement is implied not explicit parameter (i.e. comparison operator for C++)
  - Explicit – The requirement is an explicit parameter

Coroutines

- Coroutine – Special subprogram that does not have the traditional caller callee relationship. Often called symmetric units control model.
- Resume – Invocation of a coroutine (coroutines start from where left off)
- Quasi-Concurrency – Provides appearance of coroutines running concurrently
- Python Generators – A form of coroutine in Python