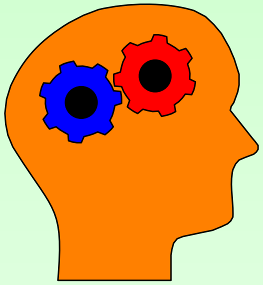
CS6202: Advanced Topics in Programming Languages and Systems

Lecture 0 : **Overview**



"Advanced Language Features and Foundations"

Lecturer : Chin Wei Ngan

Email : chinwn@comp.nus.edu.sg

Office : S15 06-01

Administrative Matters

- mainly via Web-page + IVLE

- Reading Materials :

  various papers/books
  Robert Harper : Foundations of Practical Programming Languages.
  Free PL books : http://www.cs.uu.nl/~franka/ref

- Lectures + Term Paper (100% CA)
  - Assignment (30%)
  - Take-Home Tests (20%)
  - Term Paper and Miniproject (50%)

Course Objectives

- graduate-level course with research focus

- languages as tool for programming/research

- foundations for reasoning about programs

- explore research frontiers

Course Outline

- Lecture Topics (10 weeks)
  - Advanced Language (Standard ML)
    http://www.cs.cmu.edu/~rwh/smlbook/online.pdf
  - Type System for Lightweight Analysis
    http://www.cs.cmu.edu/~rwh/plbook/book.pdf
  - Genericity for OO (Java 5)
    http://java.sun.com/j2se/1.5/pdf/generics-tutorial.pdf
    https://java-generics-book.dev.java.net/
  - Formal Reasoning – Separation Logic + Theorem Provers

- Term Paper Project (7 weeks)
  - Read, Present, Research, Critique, Evaluate

Possible Term Paper Topics

Dependent types and sized analysis .
Types for security.
Language support for XML processing.
Security Vulnerability analysis.
Automatic ProgramVerification.
Domain-specific languages (e.g. sensor programming).
Real-time Languages
Resource Analysis for Embedded Devices
Reasoning about Program Concurrency.
OO Genericity.
Others : …(you propose and let me know)

Advanced Language - ML

- Strongly-typed with polymorphism
- Higher-order functions
- Mostly pure except for mutable references.
- Algebraic data types + records
- Exceptions
- Strong module system - components
- Advantages : concise, abstract, reuse

- Why use ML ?

productivity

Example - ML Program

- Apply a function to every element of a list.

a type variable

```
datatype ‘a list = Nil | Cons of ‘a * (‘a list)
```

type is : (‘a → ‘b) * (‘a list) → (‘b list)

```
fun map (f, Nil)         = Nil
  | map (f, Cons(x,xs)) = Cons (f(x), map(f,xs))


map(inc,Cons(1,Cons(2,Cons(3,Nil))))
             ==> Cons(2,Cons(3,Cons(4,Nil))))
```

Type System – Lightweight Analysis

- Abstract description of code + genericity
- Compile-time analysis that is tractable
- Guarantees absence of some bad behaviors
- Issues – expressivity, soundness, completeness, inference?
- How to use, design and prove type system.
- Why?

detect bugs

Java 5

- mainstream language with generic types
- sophisticated subtyping mechanism
- F-bounds polymorphism with use-site variance
- Why?

generic code + type safety

Example – Java 4

- Inclusion polymorphism – safe during upcast but may fail during downcast.

```
class Cell {
   Object val;
   Object get()        { return val; }
   void set(Object x ) { val = x; }
}


   Cell c;

   c.set(new Integer(3)};

   Integer y = (Integer) c.get();
```

generic container

safe upcast

downcast may fail

Background to OO Genericity

- Why not adopt FL’s type polymorphism?
- Covariance for container
  `List<Int> <: List<Num>`
  but requires immutability while OO has mutable objects

Solutions

- GJ,Pizza : Parametric type
- Eiffel, Scala, C# : Declaration-site variance
- Java 5 : Use-site variance

Example – Java 5

- Bounded parametric polymorphism with variance

```
class Cell<T> {
   T val;
   <T> Cell<? extends T> | T get() { return val; }
   <T> Cell<? super T> |  void set(T x ) { val = x; }
}

   Cell<Integer> c;
   <Integer> c.set(new Integer(3)};
   Cell<? extends Number> d;
   d = c;

   (?) c.get();
   (?) d.get();
    d.set(new Float(1.0));
    d.set(null);
```

type parameter

reading/writing

for reading mainly

illegal due to writing

Separation Logic and Theorem Proving

- Is sorting algorithm correct?
- Any memory leaks?
- Any null pointer dereference?
- Any array bound violation?

- What is the your specification/contract?

- How to verify program correctness?

- Issues – mutation and aliasing

- Why?

sw reliability