Foundations of Program Analysis
Fall 2013

Instructors:
Adam Chlipala & Armando Solar-Lezama

Meeting Times:
Lecture: MW 1-2:30  Room: 66-156
Optional Recitation: F 1-2:30 Room: TBA

Grading:
Homework: There will be 6 homework assignments worth between 15 and 20% each (subject to minor changes). The homework assignments will consist of problem sets and programming assignments relating to the material covered in class. Homework is due no later than 5:00 pm on the day marked in the assignment handout; late homework will be accepted with a 10% penalty for each day of delay.

Useful Textbooks:
Types and Programming Languages (TAPL), Benjamin C. Pierce, MIT Press, 2003
The Formal Semantics of Programming Languages, Glynn Winskel, MIT Press, 1993

We'll also be experimenting with using the Coq proof assistant software <http://coq.inria.fr/> for some assignments. Students aren't expected to become experts on Coq, but here are some Coq resources anyway:
Certified Programming with Dependent Types: a Pragmatic Introduction to the Coq Proof Assistant, Adam Chlipala, MIT Press, 2013 (not yet available in print, but a draft is available online <http://adam.chlipala.net/cpdt/>)
Software Foundations, Benjamin C. Pierce et al., <http://www.cis.upenn.edu/~bcpierce/sf/>
### Preliminary Course Schedule:

**Unit 1: Intro to Functional Programming & Operational Semantics**

1: W 9/4  Introduction to Functional Programming and Types

**HW1 Out**  
Some interesting reading about the genesis of functional programming:

- *Can Programming be Liberated from the Von Neumann Style? A functional Style and its Algebra of Programs.* by John Backus

- *A History of Haskell: Being Lazy With Class.* by Hudak, Hughes, Peyton Jones and Wadler.

2: M 9/9  Lambda Calculus

3: W 9/11  Big-Step vs. Small-Step Semantics and the λlet calculus

4: M 9/16  Coq Crash Course (examples in operational semantics)

**Unit 2: Type Theory**

5: W 9/18  Introduction to Simple types

**HW2 Out**  
Luca Cardelli’s "Type Systems", *Types and Programming Languages* 

6: M 9/23  Hindley-Milner type inference and polymorphic types

7: W 9/25  Algebraic data types & their ingredients: product, sum, and recursive types

8: M 9/30  Subtyping

**Unit 3: Types for Imperative Programs**

9: W 10/2  Typing of Imperative Programs

**HW 3 Out**  

10: M 10/7  Verification of Complex Properties with Types: Information Flow

11: W 10/9  Verification of Complex Properties with Types: Race Detection

Unit 4: Axiomatic Semantics

12: W 10/16 Intro to Axiomatic Semantics

HW4 Out
Assigning Meanings to Programs,
Robert Floyd

13: M 10/21 Verification Condition Generation
An Axiomatic Basis for Computer Programming,
Hoare

14: W 10/23 Total Correctness and Termination

15: M 10/28 Separation Logic

16: W 10/30 Axiomatic Semantics for
Concurrency: Rely-Guarantee &
Concurrent Separation Logic

Unit 5: Abstract Interpretation

17: M 11/4 Dataflow Analysis, Lattices, Fixed
Points

HW5 Out
Gary Kildall, A Unified Approach to Global Program

18: W 11/6 Abstract Interpretation, Galois
Connections
Cousot, P. and Cousot, R. 1977. Abstract
interpretation: a unified lattice model for static
analysis of programs by construction or
approximation of fixpoints. POPL '77.

19: W 11/13 Abstract Interpretation, Galois
Connections

20: M 11/18 The Heap: Inferring Loop Invariants
about Data Structure Shape
Solving Shape-Analysis Problems in Languages with
Destructive Updating
Sagiv, Reps and Wilhelm, POPL 93
Unit 6: Model Checking

21: W 11/20    Intro to Models and Properties

22: M 11/25    Temporal Logic

23: W 11/27    Explicit State Model Checking  **HW6 Out**

24: M 12/2    Symbolic Model Checking

25: W 12/4    Software Model Checking with Abstraction Refinement

26: M 12/9    From Model Checking to Synthesis

27: W 12/11    More Synthesis