CS 550: Semantical Foundations of Programming Languages

Spring, 2018

Instructor: Deepak Kapur, kapur@cs.unm.edu, x7-1581, Farris 3140.
Office (Farris 3140) Hours: Tuesdays: 4:45PM-6PM;Thursdays: 2:15PM-3:15PM

Course Contents

This course focuses on semantics of imperative programming languages, program analysis and verification. The course will primarily cover axiomatic and denotational approaches to semantics of programming languages. This will be followed by an overview of the abstract interpretation framework and its application to program analysis. The course may also briefly review lambda calculus and logic programming. Besides lectures, the course work will include an extensive set of home work problems, one or two midterms (on Tuesday or Thursday following the spring break depending upon my travel schedule), a term paper/project leading to a class presentation (the written report due by the last day of the class) and a final exam.

Here is a list of possible topics covered in the course during its previous offerings.

1. Overview of Logic: A bit about propositional connectives and quantifiers. Characterizing Program States using Formulas/Properties.
2. Programs as Predicate Transforms.
3. Hoare Triples. Axiomatic semantics of a toy language. Weakest precondition and strongest postcondition.
4. Backward vs Forward Semantics: The case of assignment statement.
5. Importance/Significance of Loop Invariants.
6. Generating Loop Invariants Automatically/Semi-Automatically (this will be a key theme of the course from a research perspective).
7. Well-founded relations, ranking functions and methods for showing termination of programs. Total correctness.
8. Logic Programming: Horn Clauses, Relational Programming; Reversible Computation.
9. Review of Unification. Unification as Constraint Solving.

10. Semantics of Logic Programs: Herbrand Universe, Herbrand Interpretation, Herbrand Models.

11. Overview of Denotational Semantics.

12. Fix-Point Theory; examples of fixed points; Domain Theory. Complete Partially Orderest Sets. Lattices. Tarski-Knaster Theorem.

13. Information Theoretic Orderings; Least Fixed Point. Proving whether a given function is a fixed point (least) of a functional.

14. Fixed Points of functionals and recursive definitions

15. Denotational Semantics of a Toy Language using recursive functions.

16. Relationship between Denotational Semantics and Axiomatic Semantics.

17. Semantics of Logic Programming revisited using Fixed Points.

18. Abstract Interpretation Framework: Galois Connection. Widening.

19. Examples of Program analysis using Abstract Interpretation Framework.

20. Polyhedral Domain, Octagonal Domain and Zonal Domain.

21. Student presentation on their projects/papers.

Some Helpful Hints and Course Policies

1. Every attempt should be made to participate in the class. If you are unable to attend a class for some emergency, inform me and before coming to the next class, please consult notes of one of your colleagues to find out what happened in the class.

2. The best way to learn the course material is to do home work problems. It is better to start working on home works early enough so that you can utilize office hours to clarify doubts and/or fix any gaps in your understanding of the material.

   Try to work on the problems by yourself. I am not fond of homework problems being solved collaboratively. If you end up working with others, state on your homework solutions what problems you collaborated on and with whom, and the nature of collaboration. Please make sure that you write the answers to the homework problems by yourself even if you have consulted others.

   Cheating will not be tolerated. Anyone found cheating will be asked to drop the class with a **WF** at the very least. The department chair, the department faculty, and even the Dean may be informed about any cheating case.

3. Homeworks will typically be due a week after they are handed out in the class (usually on Tuesdays).

4. The course grade will be determined as follows:

   - Homeworks: 30%
   - Midterm: 15%.
   - Project/Paper including Presentation: 25%.
   - Final: 25%.
   - Class Attendance and Participation: 5%.

Recommended Books:

You can pick any book on formal semantics of programming languages. The following books look pretty good.

B. Pierce, *Software Foundations.* It is free. A good aspect of this book is that it uses Coq theorem prover for presenting key ideas about software verification and program analysis.

G. Winskel, *The Formal Semantics of Programming Languages.* MIT Press.

This nice comprehensive book is a theoretical approach to formal semantics of programming languages. It also provides the necessary background in set theory, logic and domain theory. It perhaps is good for discussion of denotational semantics and structural operation semantics. Later chapters on information systems, nondeterminism and concurrency make excellent advanced reading material.

J.B. Almeida, M.J. Frade, J.S. Pinto, and S.M. de Sousa, *Rigorous Software Development: An Introduction to Program Verification.* Springer.

This book, which is freely available from any of the UNM IP addresses, focuses on the axiomatic approach to programming language semantics. It also provides a reasonable background of propositional logic and first-order predicate logic. This book will be excellent for 1/3 of the topics in the course focusing on Hoare-Floyd style of axiomatic semantics.

H.R. Nielson & F. Nielson, *Semantics with Applications: An Appetizer.* http://www.springerlink.com/c

This book, which is also freely available, is a good introductory undergraduate text on operational semantics (small step as well as big step), structural operational semantics, as well as denotational semantics. However, the students did not find it easy when I used it for the course several years ago.

A. Bradley & Z. Manna, *Calculus of Computation.* http://www.springerlink.com/content/wv0127/

This book, which is also freely available, focuses mostly on the axiomatic method and automated program verification. It has a nice coverage of logic and decision procedures for theories.

J. Loeckx & K. Sieber, *The foundations of program verification.* Wiley-Teubner, 1987.

This very expensive book (because it is out of print) focuses on Floyd-Hoare method of axiomatic semantics and program verification based on that approach. It is very rigorous and has lots of examples and exercises. It also has a few sections on denotational semantics. Overall the book is very good. I used it about several years ago. Students interested in theory liked it, but others did not.

D. Schmidt, *Denotational Semantics Methodology.*

This out of print book focuses primarily on denotational semantics. Its coverage is quite extensive. Every year I have used some chapters from the book when I cover denotational semantics. A revised edition (but without figures etc) is available freely on the web.

J. Mitchell, *Foundations for Programming Languages.*

C. Gunter, *Semantics of Programming Languages.*