There is a growing need for effective methods of designing correct computer hardware and software, particularly for safety critical systems such as automotive embedded systems software or for systems that are costly to recall such as microprocessors (e.g. The $500M loss by Intel on the Pentium division error.) To cope with these problems, Formal Methods have been developed based on the use of mathematical logic precision tools for specifying and reasoning about program correctness. Hardware, software, and design automation companies use formal methods to make their products more reliable and less costly to develop. This course will survey the basic concepts of formal methods. The emphasis will be on using and applying mathematical logic plus finite state systems theory to program verification and debugging.

Topics

1 Preliminaries
Discrete Math, Logic, Automata, Transition Systems

2 Verification of Sequential and Non-Deterministic Programs
Flowchart Programs Invariants, Well-founded sets
Assertional Reasoning Partial and total correctness, compositionality
Predicate transformers Weakest precondition calculus(wp, wlp)

3 Verification of Concurrent and Reactive Programs
Linear Temporal Logic (LTL) G(always), F(sometime), X(next time) ...
Branching time Temporal Logics CTL, CTL*
Model Checking Tarski-Knaster Theorem for Branching time
Language containment for Linear time
State Explosion Problem and Solutions
As time permits advanced topics from ...

4 Advanced
Linear time vs Branching time temporal logics
Tableaux for testing satisfiability and automata construction
Binary decision diagrams for symbolic model checking
Abstraction techniques: Homomorphisms, Bisimulation, Symmetry etc

5 Texts
Huth and Ryan : Logic for Computer Science

6 Grading Policy
Participation: 50%
Research \ Expository Project with 15 page report: 50%