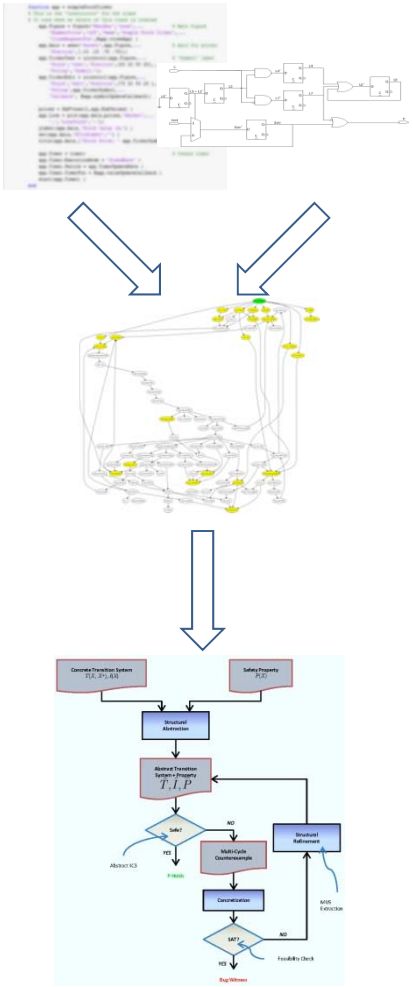
Formal Verification of Hardware and Software Systems

EECS 598-006 Winter 2017
TuTh 9:00-10:30
1690 BBB
Instructor: Karem Sakallah

Overview:



This course explores the latest advances in automated proof methods for checking whether or not certain properties hold under all possible executions of a complex hardware or software system. Specifically, we focus on the class of "control-centric" properties, namely those properties that are *weakly-dependent* on the data state of the system. Examples of such properties include, among others, the equivalence between different implementations of an instruction set architecture (hardware) and the correct usage of an Application Programming Interface (software).

The key to the scalable verification of such properties is a closed-loop CounterExample-Guided Abstraction Refinement (CEGAR) framework that involves:

a. A suitable state transition system encoding of the software or hardware being checked and the property they are expected to satisfy;
b. Structural abstraction of irrelevant data state that has nothing or little to do with the property;
c. Full unbounded reachability analysis of the abstract state space using efficient incremental induction algorithms;
d. Concretization of any resulting abstract counterexamples to determine their feasibility;
e. Automatic refinement of any spurious counterexamples that bring back only those relevant data constraints needed to provably establish that the property holds or to demonstrate a true violation.

The automated reasoning engines that make this possible are the modern Conflict-Driven Clause-Learning (CDCL) Boolean Satisfiability (SAT) solvers, the SAT modulo Theory (SMT) solvers, and the Minimal Unsatisfiable Subset (MUS) extractors.

Logistics:

This is a graduate special topics course. The only prerequisites are graduate standing in CSE and good programming skills. The course will be based on reading classic papers on software and hardware verification as well as more recent papers that describe advances in automated reasoning algorithms and their applications to verification. A theme of the course will be to find common threads that tie together the seemingly-disparate methods used in HW and SW verification. Students can expect to learn how to encode software programs and hardware circuits as transition systems, and how to develop suitable abstractions for checking control-centric properties. Additionally, they will have hands-on experience with a variety of existing reasoning and verification tools and the option of developing extensions to these tools. The course work will consist of a few short assignments to help the students master the main technical issues and semester-long individual or group projects selected by the students. The course is expected to satisfy the CSE PhD depth requirement.