Automatic Verification of Programs
IDC Herzliya
Fall 2013

Time: Tuesday, 15:30-18:00

Lecturer: Dr. Udi Boker, udiboker@idc.ac.il
Office hour: Tuesday, 18:15-19:15

Grade
Exercises: 20%
Exam: 80%
For passing the course, one should both pass the exam and have a final passing grade.

Books
The course will not follow a specific textbook.
Some of the course subjects appear in the following books:

- *Model Checking*, by E. M. Clarke, O. Grumberg, and D. A. Peled

Course Overview
It is a graduate-level course on formal verification.
We will address theoretical and practical aspects of hardware and software verification, discussing the question of how and whether one can automatically prove that a given system satisfies all the properties it is required to. The course will briefly speak of *deductive verification*, a method for verifying correctness by syntactically proving theorems on the program’s text, and will concentrate on *model checking*, a method for verifying correctness by analyzing the state space of the program’s runs. The framework that we will mainly look at is of verifying whether a *reactive system*, modeled by a *Kripke structure*, satisfies an *ongoing behavior*, modeled by a *temporal logic* formula. The main practical limitation of model checking is due to the *state-explosion problem*, namely the possibly huge size of the graph that corresponds to the program’s runs. We will study the basic algorithms for model checking, based on the *automata-theoretic approach*, as well as methods for coping with the state-explosion problem, among which are *symbolic representation, partial-order reduction, bounded model checking*, and *abstraction*. We will also look on verification tools, such as NuSMV and SPIN, understanding how they work and experiencing in using them.
Topics

- The core undecidability underlying formal verification.
- The advantages and disadvantages of model checking and of deductive verification.
- Modeling reactive systems by means of Kripke structures.
- Formalizing specifications by means of temporal logic: branching time vs. linear time temporal logic; the logics CTL, LTL, CTL*, and PSL; the logic expressiveness; property types such as safety and liveness.
- CTL model checking.
- Model checking LTL using the automata theoretic approach: Büchi automata over infinite words; the product of an automaton and a Kripke structure; translating a temporal logic formula to a Büchi automaton, checking whether an automaton has an empty language.
- The state explosion problem of model checking.
- Symbolic representation of systems and specifications using binary decision diagrams (BDDs), and symbolic model checking.
- Bounded model checking.
- Partial order reduction.
- Abstraction: simulation and bisimulation, over approximation and under approximation, counter example guided abstraction refinement (CEGAR).
- Deductive verification: The Floyd-Hoare logic, preconditions and postconditions, Hoare’s inference rules, loop invariants, Dijkstra’s weakest precondition technique.
- Combining model checking and deductive verification.
- Software vs. hardware verification.
- Verification tools, such as NuSMV and SPIN – looking at the algorithms they use, and studying how to work with them.