CS 745/845: Formal Specification and Verification of Software Systems

Catalog description

Focuses on the formal specification and verification of reactive systems, most notably concurrent and distributed systems. Topics relevant to these systems, such as nondeterminism, safety and liveness properties, asynchronous communication or compositional reasoning, are discussed. We rely on a notation (TLA$^+$, the Temporal Logic of Actions) and a support tool (TLC, the TLA$^+$ Model Checker). Prereqs: CS520 & CS659.

Overview

Formal methods can be used to complement other validation techniques (like testing) by providing a more rigorous, mathematically grounded view of software and hardware systems. Specifications (intended behavior) and implementations (actual systems) can be modeled with various degrees of abstraction. Such models can then be used to better track bugs or to verify correctness. Formal methods techniques are particularly beneficial to *safety-critical* systems, for which faults can have dramatic consequences (e.g., automated systems in transportation, medical equipment, industrial processes or infrastructure).

This course is an introduction to two formal verification techniques. First, we discuss *model checking*, a technique that checks that a model satisfies its specification by basically enumerating all its possible behaviors. Model checking has been quite popular in industry and applied to both hardware (e.g., Intel) and software (e.g., Amazon) systems. It is attractive because it can be mostly automated once a system and its specification have been modeled. Model checking has proven to be very valuable in finding tricky bugs and the behaviors that trigger them, especially in (nondeterministic) concurrent and distributed systems.

A limitation of model checking is that it does not scale well as a verification tool, due to the massive growth in the number of possible behaviors as systems become more complex (a phenomenon know as *state* (or *state space*) *explosion*). Alternatively, correctness of a model can be shown by formally proving that the model satisfies its specification. This course introduces the concept of formal correctness proofs, using two classic techniques: *inductive invariants* (for safety proofs) and *well-founded sets* (for liveness proofs).

The class spends about three quarters of the semester on modeling and model checking and one quarter on proofs. The focus is on reactive (concurrent) sysyems. The same formalism, TLA$^+$, is used throughout. TLC is used as the model checker for TLA$^+$; TLAPS, the TLA$^+$ proof assistant currently being developed by Microsoft and INRIA, is not used. Coursework is identical for graduate (CS-845) and undergraduate (CS-745) students, but graduate students need to achieve at least a B$^-$ grade to pass the course.

Attributes

- This course is one of the CS electives designated as *theory*.

Evaluation

Six homework assignments (20%), one project (30%) and two exams (50%).

Minimum score for each grade: A: 90, A$^-$: 87, B$^+$: 83, B: 80, B$^-$: 77, C$^+$: 73, C: 70, C$^-$: 67, D$^+$: 63, D: 60, D$^-$: 57.

ABET Outcomes

- **Outcome 1:** students are required to consider software systems at a higher level of abstraction than programming language code; they are made to express implementation and specification aspects of these systems logically and rigorously.

ABET Curriculum

- **Curriculum 1:** model checking.
- **Curriculum 4:** b) set theory, symbolic logic, proofs via inductive invariants and well-founded sets.
- **Curriculum 6:** e) modeling, debugging and verification of reactive and concurrent systems.
- **Curriculum 7:** state transition systems, temporal logic.

Topics

- **Formal specification and verification:**
  - implementation vs specification
  - functional correctness, typical properties (precondition, postcondition, invariants, termination)
- **Reactive systems as state transition systems:**
  - reactive systems vs transformational systems
  - system states, initial states, state transitions, behaviors
  - linear-time temporal logic (LTL), safety and liveness
  - formal definition of correctness
- **Modeling of reactive systems:**
  - state predicates
  - state modeling using sets and functions
  - sequentiality, parallelism, nondeterminism, atomicity
  - weak and strong fairness
- **Reactive systems in TLA$^+$:**
  - state transitions as binary predicates
  - stuttering and termination
  - "next-state" predicates as disjunctive formulas
- **TLA$^+$ syntax and semantics:**
  - sets and functions in TLA$^+$
  - tuples, sequences and records as functions
  - quantifiers and set builder notation
  - finite sets and cardinality
  - $\wedge$ and $\vee$ in bulleted lists form
  - `IF-THEN-ELSE`, `EXCEPT`, `UNION` and `CHOOSE` operators
- **System properties in TLA$^+$:**
  - $\Box$, $\Diamond$ and $\rightsquigarrow$ properties
  - state-based and action-based properties
  - correctness as logical implication
- **Model checking with TLC:**
  - Explicit-state model checking
  - using TLC, model configuration
  - limitations of model checking
- **Proving properties in TLA$^+$:**
  - inductive invariants, `INV1` rule
  - proving action properties
  - well-founded sets (variants), `WF1` rule, *Lattice Rule*

Textbooks

Reference:

- Leslie Lamport. *Specifying Systems: The TLA$^+$ Language and Tools for Hardware and Software Engineers*, Addison-Wesley, 2003. ISBN: 0-321-14306-X.
- Leslie Lamport. *The Temporal Logic of Actions*, ACM Trans. Program. Lang. Syst., 16(3):872-923, 1994.

Additional:

- Leslie Lamport. *The TLA$^+$ Video Course*. https://lamport.azurewebsites.net/video/videos.html
- James L. Hein. *Discrete Mathematics*, Jones and Bartlett Computer Science, 2003, second edition. ISBN: 0-7637-2210-3. *(for a discrete math refresher)*