SE 5302: Formal Methods

Course Instructor: Parasara Sridhar Duggirala, Ph.D.

Catalog Description. 3 credits. This course is designed to provide students with an introduction to formal methods as a framework for the specification, design, and verification of software-intensive embedded systems. Topics include automata theory, model checking, theorem proving, and system specification. Examples are driven by cyber-physical systems. The course is addressed to students in engineering who have had at least a year of software or embedded systems design experience.

Pre-Requisites: SE 5100 or SE 5101 or SE 5102 and at least one year of software or embedded systems design experience.

Course Delivery Method. The course will be offered online, asynchronously, in small recorded modules according to the course schedule and syllabus. Direct and live communication with the instructor will be available each week, according to the class schedule, for discussion, questions, examples, and quizzes. Attendance at live sessions is required, and you must notify the instructor in advance if you cannot attend. A social networking tool called Slack will be used to communicate with students and the instructor between live sessions.

Course Objective. This course is designed to provide students with an introduction to formal methods as a framework for the specification, design, and verification of software-intensive embedded systems. Topics include automata theory, model checking, theorem proving, and system specification. Examples are driven by control systems and software systems.

Anticipated Student Outcomes. By the end of the course, a student will be able to

(1) Gain familiarity with current system design flows in industry used for embedded system design, implementation and verification.
(2) Learn what formal methods are and how they are used in embedded systems design.
(3) Learn how to translate informal requirements to formal specifications.
(4) Learn languages for formal specifications and the applicability and appropriateness of various language choices for expressivity and efficiency.

(5) Learn how formal specifications and formal methods can be used in verification.

(6) Learn what model checking is and how it can be used in embedded systems verification.

(7) Learn the theory behind SAT solvers, SMT solvers, and bounded model checking.

(8) Learn how model checking can be used for real timed, continuous, and hybrid systems.

(9) Learn about program analysis – both static and dynamic.

(10) Learn about theorem proving and its use in embedded systems verification.

(11) The course is intended to serve as a key component to achieve standard work proficiency levels L2-L3 in embedded system design.

Course Organization

The course is organized into three learning modules: (1) Theorem Proving, (2) Temporal Logics, (3) Model Checking.

Learning Module I: Temporal Logics (2-6)

Learning Module II: Model Checking (7-11)

Learning Module III: Theorem Proving (12-14)

Structuring of these 3 learning modules into 14 lectures of a one semester course, along with the topics and references, is included below.

Course Outline

Lecture 1: Introduction to product development processes for embedded systems

- Current UTC product development process
- Current process enhanced by platform-based design principles
- Review of Embedded Systems Abstractions and Models
- Introduction to formal methods
  - What can we do?
  - What can we not do?
• Where does this course fit in: Requirements, Requirements Compliance
• Comparisons to other verification methods – simulations, review, etc.
• Regulations in the aerospace
• Domain of applications

-------------------------------Module 1: Temporal Logics-------------------------------

Lecture 2-6: Temporal logics and formal specifications

• Labeled transition systems (for discrete specifications)
• Executions, invariants, traces, composition
• Abstract and concrete specifications, and the notion of implementation
• Trace inclusion: abstraction (simulation, implementation) functions/relations
• Propositional calculus
• Predicate calculus
• Set theory
• LTL, CTL, PSL
  o How do you select which one?
  o Tradeoffs with respect to model checking
  o Tie specification languages to capabilities of algorithms
• Complexity, fairness
• Formal semantics
• Model checking examples
• Case studies on real systems
  o Composition of a real system
• Tools.
• Formal specifications
  o Applicable problems
  o Requirements modeling of hardware and software systems for formal methods
  o State-based and transition-based specifications (redux)
  o Functional specifications
  o Languages: Z-notation, Statechart, ASL

-------------------------------Module 2: Model Checking and Formal Verification-------------------------------

Lectures 7-9: Model checking

• Fundamental concepts
  o What can model checking do and what it cannot do?
• State space explosion
• Symbolic model checking
  o Binary decision diagrams
  o SAT solvers
  o SMT solvers
• Bounded model checking
• Invariant checking
• Tools

Lecture 10-11: Formal Verification
• Abstract interpretation
• Functional correctness
• Meaning of correctness
• Verified Design by Contract

-----------------------------Module 3: Theorem Proving-----------------------------

Lecture 12-14: Theorem Proving
• Meaning of proofs
• Automated and interactive theorem proving
• Tools – Isabelle, Coq, TLA+, PVS

Texts.

Copyright. Copyrighted materials within the course are only for the use of students enrolled in the course for purposes associated with this course and may not be retained or further disseminated.

Due Dates and Late Policy. All course due dates are identified in the Course Schedule. Deadlines are based on Eastern Standard Time; if you are in a different time zone, please adjust your submittal times accordingly. The instructor reserves the right to change dates accordingly as the semester progresses. All changes will be communicated in an appropriate manner.

Student Conduct. http://www.dosa.uconn.edu/student_code.html. Students are responsible for adherence to the University of Connecticut student code of conduct. Pay attention to the section on Student Academic Misconduct, “Academic misconduct is dishonest or unethical academic behavior that includes, but is not limited, to misrepresenting mastery in an academic area (e.g., cheating), intentionally or knowingly failing to properly credit information, research or ideas to their rightful
originators or representing such information, research or ideas as your own (e.g., plagiarism).” Examples of academic misconduct in this class include, but are not limited to: copying solutions from the solutions manual, using solutions from students who have taken this course in previous years, copying your friend’s homework, looking at another student’s paper during an exam, lying to the professor or TA and incorrectly filling out the student workbook.

**Absences.** Make-up of missed exams requires permission from the Dean of Students, see “Academic Regulations.” Midterm-exams are treated the same as Final Examinations. Students involved in official University activities that conflict with class time must inform the instructor in writing prior to the anticipated absence and take the initiative to make up missed work in a timely fashion. In addition, students who will miss class for a religious observance must “inform their instructor in writing within the first three weeks of the semester, and prior to the anticipated absence, and should take the initiative to work out with the instructor a schedule for making up missed work.”

**Adding or Dropping a Course.** If you should decide to add or drop a course, there are official procedures to follow:

- Matriculated students should add or drop a course through the Student Administration System.
- Non-degree students should refer to Non-Degree Add/Drop Information located on the registrar’s website.

You must officially drop a course to avoid receiving an "F" on your permanent transcript. Simply discontinuing class or informing the instructor you want to drop does not constitute an official drop of the course. For more information, refer to the online Graduate Catalog.

**Grading.** As this is a graduate course, we will adopt relative grading for this course. The distribution of points will be fitted to a normal distribution curve and the grades will be awarded according to the performance of the student.

**Academic Calendar.** The University’s Academic Calendar contains important semester dates.

**Students with Disabilities.** Students needing special accommodations should work with the University’s Center for Students with Disabilities (CSD). You may contact CSD by calling (860) 486-2020 or by emailing csd@uconn.edu. If your request for accommodation is approved, CSD will send an accommodation letter directly to your
instructor(s) so that special arrangements can be made. (Note: Student requests for accommodation must be filed each semester.)

**Instructors’ Contact Information.** Sridhar Duggirala, psd@uconn.edu

**Helpful Links:**
- Virtual Computer Lab at UConn: [http://skybox.uconn.edu/](http://skybox.uconn.edu/)
- Course Material: [https://lms.uconn.edu](https://lms.uconn.edu)
- Institute for Advanced Systems Engineering: [http://www.utc-iae.uconn.edu/](http://www.utc-iae.uconn.edu/)