CSE 5320/6392 – Computer-Aided Verification for Systems
Spring 2021, MW 1:00 pm to 2:20 pm Online Synchronous

INSTRUCTOR INFORMATION

INSTRUCTOR(S)
Dr. Allison K. Sullivan

OFFICE NUMBER
ERB 529

EMAIL ADDRESS
allison.sullivan@uta.edu

OFFICE HOURS
By appointment. Please email and we will schedule a time for a Microsoft Teams meeting. In person office hours will not be held unless necessary due to Covid-19 concerns.

COURSE INFORMATION

SECTION INFORMATION
CSE 5320 Section 002 / CSE 6392 Section 003, Track: Software Engineering

TIME AND PLACE OF CLASS MEETINGS
MW 1:00 to 2:20 pm on Microsoft Teams. This course will be Online Synchronous.

● Why synchronous? Many lectures will involve small group breakouts to work on lab assignments or there will be a focus on in class discussions.

COURSE DESCRIPTION
Mathematical logic provides the foundation for a rich set of tools for reasoning about systems and discovering whether their behavior meets our expectations. In spite of how it is often presented, logic is not only good for describing the state of Socrates and figuring out who is the imposter in “Among Us”, but also the state of buffers, caches and subtler, dangerous errors that turn out to be mistakes in the initial design. In this course, you will learn to use logic-based tools to describe and analyze program designs, algorithms, data-structures, and other system artifacts. Topics include concepts related to the theory and applications of automated reasoning techniques, such as satisfiability solving, finite model finding, model checking, and abstract interpretation. These topics will be spread out over four high-level sections: logical foundations (background material), verification of system designs, verification of system implementations, and program synthesis.

PRE-REQUISITES
● Prior experience with formal verification tools or modeling languages is not required.
● This course does assume that you know how to program and that you have taken a course in discrete mathematics.
● You can use any language you want for your course project. However, you need to be comfortable compiling, installing, and running code written in C, C++, and Java. The tools covered in the course are written in these languages, and most are released as source code.

REQUIRED TEXTBOOKS AND OTHER COURSE MATERIALS

Required Texts:
● None. We will study academic publications and utilize open-source notes and tutorials.
Suggested Readings/Texts: for further background, you are welcome to check out the following:


DESCRIPTIONS OF MAJOR ASSIGNMENTS AND EXAMINATIONS

- **Lab assignments**: Labs will contain two sections:
  - **Collaborative portions** that involve working with different computer-aided verification tools in small breakout groups.
  - **Individual portions** that involve short answer questions due by midnight on assigned lab day

- **Leading a Discussion**: Each discussion will last ~35 minutes (0.5 class):
  - 10-15 minute paper recap prepared by lead student
    - ~5 PPT slides summarizing the motivation, methods and results of the paper
    - These should be uploaded to Moodle before your presentation day
    - The student must provide critique, both positive and negative
    - What is your verdict on the conclusion in the paper? Compare to other related work
  - ~20 minute discussion by class
    - Every student in audience is expected to ask minimum 1 question
    - Recommended that all students skim through all papers beforehand

- **Class participation**: Student participation will be taken for asking at least one question during other student’s paper presentations

- **Semester project**: Due to Covid-19 and the online nature of the course, you can either do a project with a partner or by yourself.
  - The project will be a mini research project that applies computer-aided verification technique(s), such as applying known techniques or tool to new problem domains or creating a case study to evaluate existing techniques or tools. There will be a list of suggested projects, but you are free to define your own.
  - Projects will include a 5-page final report in ACM format with the following sections: overview, algorithms or logical encodings developed, evaluation and course topics applied.
  - Teams will also demo their projects during the final lecture.

Across any assignment, plagiarism and other forms of cheating and academic dishonesty will be dealt with severely.

Grading Information

**Grading**
The final grade will be the weighted sum of all work using the following weights:

<table>
<thead>
<tr>
<th>Assignment</th>
<th>Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lab Assignments</td>
<td>30</td>
</tr>
<tr>
<td>Collaborative portion</td>
<td>80</td>
</tr>
<tr>
<td>Individual portion</td>
<td>20</td>
</tr>
<tr>
<td>Discussion Lead</td>
<td>15</td>
</tr>
<tr>
<td>Class Participation</td>
<td>10</td>
</tr>
</tbody>
</table>
The grade you are given on assignment, or as your final grade, is not the starting point of a negotiation. It is your grade unless a concrete error has been made. Do not come to the instructors or TAs to ask for a better grade because you want one or you feel you deserve it. Come only if you can document a specific error in grading or in recording your scores. Errors can certainly be made in grading, especially when many students are involved. But keep in mind that errors can be made either in your favor or not. So, it is possible that if you ask to have a piece of work re-graded your grade will go down rather than up.

Remember that the most important characteristic of any grading scheme is that it be fair. Keep this in mind if you are thinking of asking, for example, for more partial credit points on a problem. The important thing is not the exact number of points that were taken off for each kind of mistake. The important thing is that that number was the same for everyone. So, it can't easily be changed once the grading is done and the exams or assignments have been returned.

**Final Grades**

Final grades will be assigned according to the following standard criteria:

<table>
<thead>
<tr>
<th>Grade</th>
<th>Score Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>90-100</td>
</tr>
<tr>
<td>B</td>
<td>80-89</td>
</tr>
<tr>
<td>C</td>
<td>70-79</td>
</tr>
<tr>
<td>D</td>
<td>60-69</td>
</tr>
<tr>
<td>F</td>
<td>&lt;60</td>
</tr>
</tbody>
</table>

Final class grades will be calculated to 2 decimal places and rounded to the nearest integer. 89.49 is a B. 89.50 is an A. The line has to be drawn somewhere, and no special allowances will be made for students whose final average falls near, but below the cutoff. There is a possibility of curves on the exam and quiz grades. There will not be a curve on programming assignments. Nonacademic explanations for poor class performance will have no bearing on the assignment of grades.

I reserve the right to curve the scale dependent on overall class scores at the end of the semester. Any curve will only ever make it easier to obtain a certain letter grade.
## Course Schedule

*These descriptions and timelines are subject to change at the discretion of the Instructor.*

**Legend:**
- Course Topic
- Holiday – No Class

<table>
<thead>
<tr>
<th>Week</th>
<th>Date</th>
<th>Subject</th>
<th>Assignments Due</th>
</tr>
</thead>
</table>
| 1    | 1/20   | What are formal methods and why should you care?  
            ➢ *Reading 0:* Use of Formal Methods at Amazon Web Services |                                    |
| 2    | 1/25   | Declarative thinking mindset: BABA-IS-YOU     |                                    |
|      | 1/27   | Introduction to Alloy  
            ➢ *Reading 1:* Using Lightweight Modeling to Understand Chord |                                    |
| 3    | 2/1    | Expressing OO-design structure  
            ➢ *Reading 2:* A Guide to Alloy |                                    |
|      | 2/3    | Lab 0: Signatures and Relations              |                                    |
| 4    | 2/8    | Expressing behavior of systems  
            ➢ *Reading 3:* Logic in Computer Science: Modelling and Reasoning about Systems. Chapter 2: Predicate Logic |                                    |
|      | 2/10   | Lab 1: Predicates                           |                                    |
| 5    | 2/15   | Scenario Refinement  
            ➢ *Reading 4:* Solution Enumeration Abstraction - A Modeling Idiom to Enhance a Lightweight Formal Method |                                    |
|      | 2/17   | Lab 2: Abstract Functions                   |                                    |
| 6    | 2/22   | Test concretization  
            ➢ *Reading 5:* One Evaluation of Mode Based Testing and Its Automation | Proposal Due |
|      | 2/24   | Alloy's backend and How it Works  
            ➢ *Reading 6:* Chapter 2: Satisfiability Solvers. Handbook of Knowledge Representation  
            ➢ *Further Background Reading:* Chapter 1: Propositional Logic. Calculus of Computation |                                    |
| 7    | 3/1    | SMT Solvers: Exploring Z3  
            ➢ *Reading 7:* Satisfiability Modulo Theories: Introduction and Applications |                                    |
|      | 3/3    | Lab 3: Building Encodings                    |                                    |
| 8    | 3/8    | Exploring the limits of Z3 verification  
            ➢ *Reading 8:* Applications of SMT solvers to Program Verification |                                    |
<p>|      | 3/10   | Lab 4: Verifying Programs with Z3            |                                    |
| 9    | 3/15   | <strong>SPRING BREAK</strong>                             |                                    |
|      | 3/17   | <strong>SPRING BREAK</strong>                             |                                    |
| 10   | 3/22   | <em>Lecture: Solver-Aided Verification (Beyond SAT/SMT Solving)</em> |                                    |</p>
<table>
<thead>
<tr>
<th>Week</th>
<th>Date</th>
<th>Subject</th>
<th>Assignments Due</th>
</tr>
</thead>
<tbody>
<tr>
<td>11</td>
<td>3/24</td>
<td><strong>Model-Based Development - Alloy</strong>&lt;br&gt;<strong>Discussion 0</strong>: TestEra: A Novel Framework for Automated Testing of Java Programs <em>ASE impact award</em>&lt;br&gt;<strong>Discussion 1</strong>: From UML to Alloy and Back Again</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>3/29</td>
<td><strong>Model-Based Development</strong>&lt;br&gt;<strong>Discussion 2</strong>: Using Model Checking to Generate Tests from Requirements Specifications&lt;br&gt;<strong>Discussion 3</strong>: Generating Optimized Code from SCR Specifications</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>3/31</td>
<td><strong>Symbolic Execution</strong>&lt;br&gt;<strong>Discussion 4</strong>: Test Input Generation with Java PathFinder <em>ISSTA impact award</em>&lt;br&gt;<strong>Discussion 5</strong>: Combining Unit-Level Symbolic Execution and System-Level Concrete Execution for Testing NASA Software</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>4/5</td>
<td><strong>Angelic Execution</strong>&lt;br&gt;<strong>Discussion 6</strong>: Constraints as Control&lt;br&gt;<strong>Discussion 7</strong>: Unifying Execution of Imperative and Declarative Code</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>4/7</td>
<td>PROJECT DAY</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>4/12</td>
<td><strong>Lecture: Solver-Aided Programming</strong></td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>4/14</td>
<td><strong>Symbolic Virtual Machine</strong>&lt;br&gt;<strong>Discussion 8</strong>: A Lightweight Symbolic Virtual Machine for Solver-Aided Host Languages.&lt;br&gt;<strong>Discussion 9</strong>: Finding Code That Explodes under Symbolic Evaluation</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>4/19</td>
<td><strong>Domain Specific Solver Aided Languages</strong>&lt;br&gt;<strong>Discussion 10</strong>: Synthesizing Memory Models from Framework Sketches and Litmus Tests&lt;br&gt;<strong>Discussion 11</strong>: Chlorophyll: Synthesis-Aided Compiler for Low-Power Spatial Architectures</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>4/21</td>
<td><strong>Program Synthesis</strong>&lt;br&gt;<strong>Discussion 12</strong>: From Program Verification to Program Synthesis&lt;br&gt;<strong>Discussion 13</strong>: Combinatorial Sketching for Finite Programs</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>4/26</td>
<td><strong>Program Synthesis</strong>&lt;br&gt;<strong>Discussion 14</strong>: Programming by Examples: PL meets ML&lt;br&gt;<strong>Discussion 15</strong>: SQLizer: Query Synthesis from Natural Language</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>4/28</td>
<td><strong>Course Wrap Up</strong>&lt;br&gt;<strong>Reading</strong>: On the Need for Practical Formal Methods</td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>5/3</td>
<td>Project Demonstrations&lt;br&gt;LAST CLASS DAY</td>
<td>Demo Due</td>
</tr>
</tbody>
</table>

**MAY 7: Report and Project Artifacts Due 11:59 pm**

**Institution Information**

UTA students are encouraged to review the below institutional policies and informational sections and reach out to the specific office with any questions. To view this institutional information, please visit the
Institutional Information page (https://resources.uta.edu/provost/course-related-info/institutional-policies.php) which includes the following policies among others:

- Drop Policy
- Disability Accommodations
- Title IX Policy
- Academic Integrity
- Student Feedback Survey
- Final Exam Schedule

Additional Information

Attendance
At the University of Texas at Arlington, taking attendance is not required but attendance is a critical indicator of student success. Each faculty member is free to develop his or her own methods of evaluating students’ academic performance, which includes establishing course-specific policies on attendance. As the instructor of this section, attendance is not mandatory but highly encouraged. However, while UT Arlington does not require instructors to take attendance in their courses, the U.S. Department of Education requires that the University have a mechanism in place to mark when Federal Student Aid recipients “begin attendance in a course.” UT Arlington instructors will report when students begin attendance in a course as part of the final grading process. Specifically, when assigning a student a grade of F, faculty report must the last date a student attended their class based on evidence such as a test, participation in a class project or presentation, or an engagement online via Canvas. This date is reported to the Department of Education for federal financial aid recipients.

Emergency Exit Procedures
Should we experience an emergency event that requires evacuation of the building, students should exit the room and move toward the nearest exit. When exiting the building during an emergency, do not take an elevator but use the stairwells instead. Faculty members and instructional staff will assist students in selecting the safest route for evacuation and will make arrangements to assist individuals with disabilities.

Students should also be encouraged to subscribe to the MavAlert system that will send information in case of an emergency to their cell phones or email accounts. Anyone can subscribe at Emergency Communication System.

Student Success Programs
For the design document and reflection document, you may utilize the below resource to help prepare your written content:

The English Writing Center (411LIBR)
The Writing Center offers FREE tutoring in 15-, 30-, 45-, and 60-minute face-to-face and online sessions to all UTA students on any phase of their UTA coursework. Register and make appointments online at the Writing Center (https://uta.mywconline.com). Classroom visits, workshops, and specialized services for graduate students and faculty are also available. Please see Writing Center: OWL for detailed information on all our programs and services.

Emergency Phone Numbers
In case of an on-campus emergency, call the UT Arlington Police Department at 817-272-3003 (non-campus phone), 2-3003 (campus phone). You may also dial 911. Non-emergency number 817-272-3381.