ECE.453/CS.447/CS.647 Syllabus

Important Websites

- Course Website: http://ece.uwaterloo.ca/~agurfink/stqam (http://ece.uwaterloo.ca/~agurfink/stqam)
- LEARN: http://learn.uwaterloo.ca (http://learn.uwaterloo.ca)
- Live stream lectures: https://twitch.tv/profarie (https://twitch.tv/profarie)

Lectures

~~2:30-3:50PM Tuesday, Wednesday, https://twitch.tv/profarie (https://twitch.tv/profarie)~~
1:00-2:20PM Tuesday, Wednesday, https://twitch.tv/profarie (https://twitch.tv/profarie)

Office Hours

By request

Instructor

Prof. Arie Gurfinkel (http://ece.uwaterloo.ca/~agurfink), Office: HOME, first . last AT uwaterloo.ca, Office Hours: by appointment online

Teaching Assistants

Nham Le

Textbook

No required textbook. Lecture slides, lecture notes, and reading material will be provided.

Course Description and Main Topics

This course will provide an introduction to software testing and quality assurance techniques. The students will learn a wide spectrum of techniques and tools that can be used to improve and evaluate software quality ranging from mature testing methodologies to cutting edge automated verification algorithms. Topics to be covered include: coverage criteria (graph, data-flow, and logic coverage), symbolic execution (static, dynamic, concolic), constraint solving (SMT), inductive invariants, automatic deductive verification, automatic invariant synthesis, and Software Model Checking.

Recommended Background

The course will include programming assignments in Java/C/Python. Background in Compilers and Logic is useful, but is not required.

Grading

Grades may be curved or adjusted at instructor's discretion.

- Assignments: 35%
  - Assignment 1: 10%
  - Fuzz Battle: %5
  - Assignment 2: 10%
  - Assignment 3: 10%
- Quizzes: 20%
  - Quiz 1: 10%
  - Quiz 2: 10%
- Final Exam: 25%
- Project: 20%

All assignments, quizzes, and tests will be returned online.

All quizzes and tests are take home, open book, closed internet. Each online test will be time limited and must be completed within 3 days. That is, you can pick the best time to take the test, but once you start, you have to finish within the given time limit.

You must pass at least the two written tests and pass the assignments to pass the course. The final grade is computed using the following formula:

```
def grade(A, P, Q1, Q2, F):
    """Final grade calculation.

    A, P, Q1, Q2, F are grades for assignments, project, Quiz 1, Quiz 2,
    and the Final, respectively. Normalized to 100%.
    """
    tests_above_fifty = len(list(filter(lambda x: x >= 50, [Q1, Q2, F])))

    normal = 0.35*A + 0.1*Q1 + 0.1*Q2 + 0.25*F + 0.2*P
    if tests_above_fifty >= 2 and A > 50:
        return normal
    else:
        return 49
```

Project

There are three choices of projects. All three are available to undergraduate students (ECE453/CS447). The last two are available to both undergraduate and graduate students (CS647). Projects can be done in groups of 2. You must declare your project by the end of Week 8. All projects must be approved by the instructor.

The exact project deliverables depend on the project chosen. All projects must include approximately a 10 page report. The length of the report depends on the amount of code involved in the project (i.e., more code means shorter report).

Project (choice 0) (U only)

In the assignments, you will be implementing a symbolic execution engine and a verification engine for a small imperative language. The project is to implement additional advanced features for the language. Details of potential features will be available on Week 6.

Implement at least two features from the list in the symbolic execution engine for `wlang`

Features

- Concolic execution in EXE-style.
- Concolic execution in DART-style.
- A different exploration strategy. Potential choices are Depth First Search, Breadth First Search, A* Search, etc.
- Symbolic state merging. For example, following this (https://www.dominic-steinhoefel.de/post/precise-symbolic-state-merging/) post. TL;DR: symbolic states can be merged together by taking disjunction of their path conditions. Merging symbolic states reduces the number of symbolic states generated, at the expense of complicated path condition.

- Use incremental solving mode of Z3. During symbolic execution many similar queries are solved over and over again. Performance of symbolic execution can be improved by using incremental solving abilities of SMT solver that allow adding (and withdrawing) constraints between multiple calls. Details on available incremental interfaces are here (https://theory.stanford.edu/~nikolaj/programmingz3.html#sec-incrementality). There are at least two different incremental solving interfaces (Scopes and Assumptions). Using each one counts for one implementation.
- Add support for bit-precise symbolic execution using the theory of bit-vectors (https://theory.stanford.edu/~nikolaj/programmingz3.html#sec-bit-vectors).
- Implement verification condition generation using Weakest Pre-Condition. We will cover this in class in Week 10. Here is a link (https://ece.uwaterloo.ca/~agurfink/stqam.w19/assets/pdf/W11-VCGen.pdf) to the slides.
- Extend `wlang` with functions. This requires adding functions to the parser. We can help with that. This is a significant feature that counts for two. Extend the language to allow for functions specifications (`requires` and `ensures` from Dafny that we will cover in class).
- Extend `wlang` with references. A reference is a variable that is allocated on the heap. This requires changing the parser. We can help with that. This is a significant feature that counts for two.

Deliverables

- **Code for the features.** If the project is done in groups, request us to create a group repository for you. All code is to be submitted into the group repository (if the project is done in a group), or in your course repository (if the project is done independently)
- **Test cases.** You have to achieve complete statement and branch coverage for all the new code that you have written.
- **Report.** An approximately 10 page report (can be 9, can be 15, cannot be 5) describing the design decisions for your implementation, any theoretical foundations of your implementation, and your testing strategy. Report is to be submitted in PDF. Suggested style is LNCS (https://www.springer.com/gp/computer-science/lncs/conference-proceedings-guidelines). An Overleaf (https://www.overleaf.com/) template is here (https://www.overleaf.com/latex/templates/springer-lecture-notes-in-computer-science/kzwwpvhwnvfj). If you insist on using MS-WORD, a template is available here (https://resource-cms.springernature.com/springer-cms/rest/v1/content/7117506/data/v1).

Project (choice 1) (U/G)

Throughout the course, we will learn of several verification techniques including fuzzing, symbolic execution, and deductive verification. In this project, you have to propose a program or an algorithm to verify with one of the techniques that we have studied in the course. The complexity of the artifact being verified will depend on the verification technique involved. The more complex the verification technique the simpler the artifact being verified can be.

Details

This project requires approval of the instructor. Talk to me. Sooner the better. Deadline is Week 10.

Deliverables

- **Code and any other artifacts you produce.** If the project is done in groups, request us to create a group repository for you. All code is to be submitted into the group repository (if the project is done in a group), or in your course repository (if the project is done independently)
- **Report.** An approximately 10 page report (can be 9, can be 15, cannot be 5) describing your experience, any theoretical foundations, any design decisions, and implementation that you had to do. Report is to be submitted in PDF. Suggested style is LNCS (https://www.springer.com/gp/computer-science/lncs/conference-proceedings-guidelines). An Overleaf (https://www.overleaf.com/) template is here (https://www.overleaf.com/latex/templates/springer-lecture-notes-in-computer-science/kzwwpvhwnvfj). If you insist on using MS-WORD, a template is available here (https://resource-cms.springernature.com/springer-cms/rest/v1/content/7117506/data/v1).

Project (choice 2) (U/G)

Quality assurance and automated verification are active areas of research. In this project, you will conduct an independent study into a topic that is closely related to the material of the course but is not explicitly covered. The study must include reading at least two scientific papers. The outcome of the project can be an implementation of new technique or a report on the topic studied.

Details

This project requires approval of the instructor. Talk to me. Sooner the better. Deadline is Week 10.

Deliverables

- **Report.** An approximately 15 page report (can be 14, can be 20, cannot be 10) critically reviewing the paper that you have selected. Your review must be critical and in-depth. Simply repeating the papers verbatim is not sufficient. You must show that you have learned and understood something new. This can be done by contrasting different techniques, creating new examples, and, potentially, implementing the techniques. If your report involves implementation, the page limit of the report can be reduced. Report is to be submitted in PDF. Suggested style is LNCS (https://www.springer.com/gp/computer-science/lncs/conference-proceedings-guidelines). An Overleaf (https://www.overleaf.com/) template is here (https://www.overleaf.com/latex/templates/springer-lecture-notes-in-computer-science/kzwwpvhwnvfj). If you insist on using MS-WORD, a template is available here (https://resource-cms.springernature.com/springer-cms/rest/v1/content/7117506/data/v1).

Course Policies

By registering for this class, students agree to the following class policies:

Independent work. All work turned in will be that of the individual student unless stated otherwise. Violations would result in zero credit to all students concerned. Policy 71 (https://uwaterloo.ca/secretariat-general-counsel/policies-procedures-guidelines/policy-71) will be followed for any discovered cases of plagiarism.

Lateness. You have 2 days of lateness to use on assignment submissions throughout the term. Each day you hand in an assignment late consumes one of the days of lateness. If you consume all of your late days, assignments that are still late will get 0 marks. You can only hand in an assignment up to the time all assignments are returned. Missed assignments get 0 marks. For example, you may hand in A1 two days late and A2

on time, or you can hand in A1 one day late and A2 one day late.

Missed Quizzes. If you miss a quiz, you will receive 0 marks for the quiz. If you have a legitimate reason (at the discretion of the instructor) that you cannot take quiz, and obtain permission from the instructor **a week** in advance, the percentage for the quiz may be shifted to the final. No alternative quiz time will be provided.

University Policies

Academic Integrity. In order to maintain a culture of academic integrity, members of the University of Waterloo community are expected to promote honesty, trust, fairness, respect and responsibility. [Check http://www.uwaterloo.ca/academicintegrity (http://www.uwaterloo.ca/academicintegrity) for more information.]

Grievance. A student who believes that a decision affecting some aspect of his/her university life has been unfair or unreasonable may have grounds for initiating a grievance. Read Policy 70, Student Petitions and Grievances, Section 4, http://www.adm.uwaterloo.ca/infosec/Policies/policy70.htm (http://www.adm.uwaterloo.ca/infosec/Policies/policy70.htm). When in doubt please be certain to contact the departments administrative assistant who will provide further assistance.

Discipline. A student is expected to know what constitutes academic integrity [check http://www.uwaterloo.ca/academicintegrity (http://www.uwaterloo.ca/academicintegrity)] to avoid committing an academic offense, and to take responsibility for his/her actions. A student who is unsure whether an action constitutes an offense, or who needs help in learning how to avoid offenses (e.g., plagiarism, cheating) or about rules for group work/collaboration should seek guidance from the course instructor, academic advisor, or the undergraduate Associate Dean. For information on categories of offenses and types of penalties, students should refer to Policy 71, Student Discipline, http:://www.adm.uwaterloo.ca/infosec/Policies/policy71.htm (http:://www.adm.uwaterloo.ca/infosec/Policies/policy71.htm). For typical penalties check Guidelines for the Assessment of Penalties, http://www.adm.uwaterloo.ca/infosec/guidelines/penaltyguidelines.htm (http://www.adm.uwaterloo.ca/infosec/guidelines/penaltyguidelines.htm).

Appeals. A decision made or penalty imposed under Policy 70 (Student Petitions and Grievances) (other than a petition) or Policy 71 (Student Discipline) may be appealed if there is a ground. A student who believes he/she has a ground for an appeal should refer to Policy 72 (Student Appeals) http://www.adm.uwaterloo.ca/infosec/Policies/policy72.htm (http://www.adm.uwaterloo.ca/infosec/Policies/policy72.htm).

Note for Students with Disabilities. The AccessAbility Services, located in Needles Hall, Room 1132, collaborates with all academic departments to arrange appropriate accommodations for students with disabilities without compromising the academic integrity of the curriculum. If you require academic accommodations to lessen the impact of your disability, please register with the AccessAbility Services at the beginning of each academic term.

Territorial Acknowledgement. The University of Waterloo acknowledges that much of our work takes place on the traditional territory of the Neutral, Anishinaabeg and Haudenosaunee peoples. Our main campus is situated on the Haldimand Tract, the land promised to the Six Nations that includes six miles on each side of the Grand River.