ECE.453 / CS.447 / CS.647
STQAM

Software Testing, Quality Assurance, and Maintenance

Winter 2019

Prof. Arie Gurfinkel

Instructor and TA

Instructor

- Prof. Arie Gurfinkel

Teaching Assistants

- Jakub Kuderski
- Qian (David) Liang

Course Web Page

- https://ece.uwaterloo.ca/~agurfink/stqam
- LEARN: https://learn.uwaterloo.ca

TAs

Qian (David) Liang
@ta-qian

Jakub Kuderski
@ta-jakub

Course Time and Location

Date: Monday, Wednesday, Friday

Location: E7 5353

Time: 3:30 – 4:20 PM

Tutorials: **by announcement only**

Wednesday 7:00 – 7:50 PM

Location: E7 5353

First Tutorial: Jan 16, 2019

Office Hours

Tuesday: 4:30-5:30 PM, DC-2551 (WatForm)

Grading

Assignments: 30%

Quizzes: 20%

Final Exam: 50%

Grades may be curved or adjusted at the Instructor's discretion

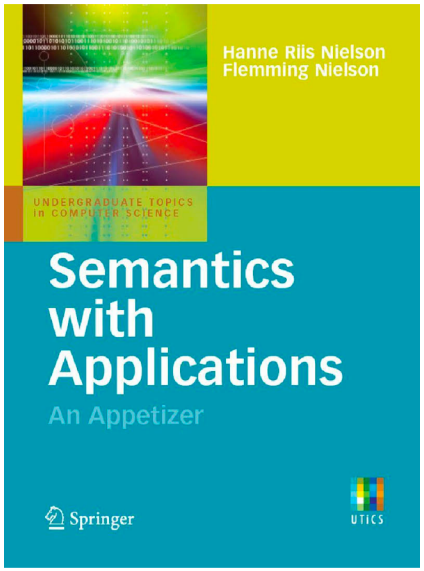
1 + 3 Assignments

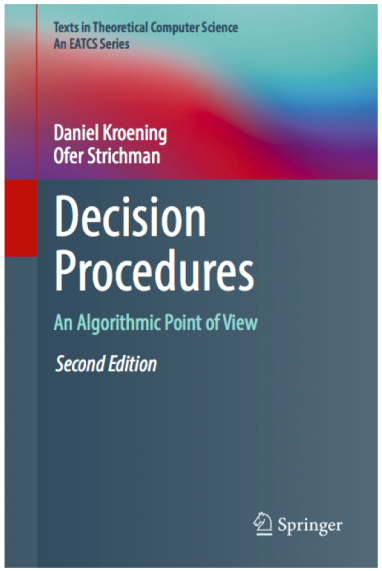
- Pen and paper exercises, and
- Programming assignments
  - mostly in Python

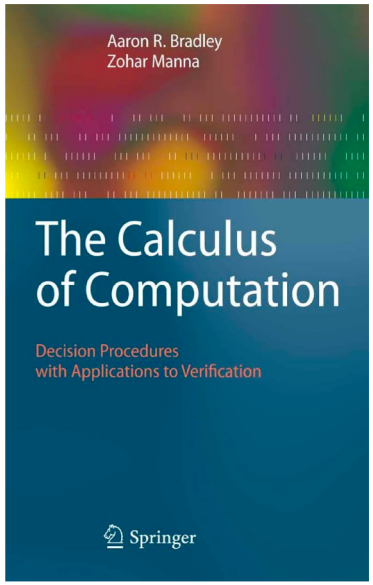
Textbook and Lecture Notes

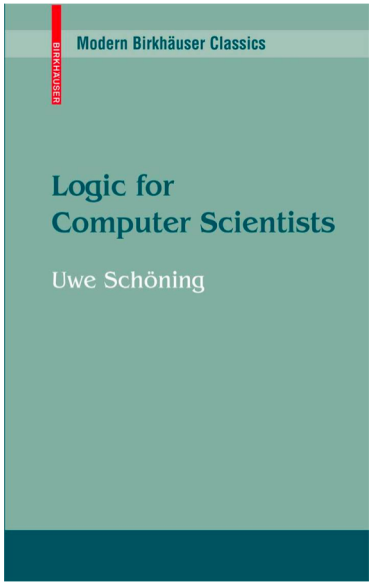
No required text book. Lecture slides and notes will be provided.

- https://ece.uwaterloo.ca/~agurfink/stqam
- LEARN: https://learn.uwaterloo.ca









Course Website & LEARN

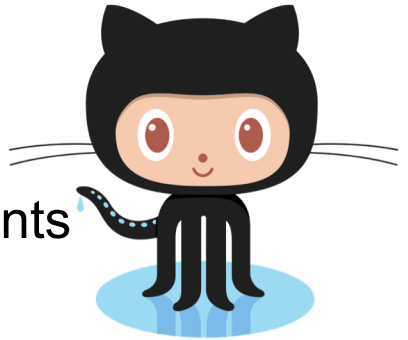
The course website is the definitive source

- When in doubt, consult the web page
- Check syllabus for final grade computation

YOUR responsibility to check for updates!

- Course website
- LEARN (http://learn.uwaterloo.ca)

GitHub and Slack



We will use **GitHub** for managing and submitting assignments

- This requires a free GitHub account
- Follow the link in Assignment 0 to get started
- Let me know if there are any problems!!!
- GitHub Tutorial: https://try.github.io

We will use **Slack** for communication (don't use EMAIL if you can!)

- https://uw-stqam-w19.slack.com/
- there are slack apps for Win/Mac/Linux/iPhone/Android – use them!
- Signup link is available on LEARN (or use your @uwaterloo.ca email)
- Create #channels for group discussion and DM for private conversations
- monitor #general for course announcements
- ask questions about assignments in #aN
- send DM to me and TA for a private question
- Share cool slack features that you find helpful with the rest of the class

Independent Work

All work turned in must be of that individual student unless stated otherwise.

Violations will result in zero credit to all students concerned. University of Waterloo Policy 71 will be followed for any discovered cases of plagiarism.

Policy on Late Assignments

You have 2 days of lateness for assignments that you can use throughout the term

- These are TWO days for the term. Not for each assignment!

Each day the assignment is late consumes one day of lateness

For example,

- You can be 2 days late on assignment A1, or
- One day late on A1, and one day late on A3, or
- You can hand all of the assignments on time ☺

Missed Quiz

If you miss a quiz, you will receive 0 for it. If you have a legitimate reason (at the discretion of the instructor) for not taking the midterm and obtain a permission from the instructor **a week** in advance, the percentage for the quiz will be shifted to the final.

See syllabus for more detail

https://ece.uwaterloo.ca/~agurfink/stqam/syllabus.html

Is this course for me?

Not a TESTING course!

- Foundations of Testing / Coverage
- Foundations of Symbolic Execution and Symbolic Reasoning
- Foundations of Deductive Program Verification
- (Possibly) Foundations of Automated Verification

Enough background?

- Can you code? (Python?) https://docs.python.org/2.7/tutorial/
- Have you used a Unix/Linux machine before?
  - command line, shell, editor…
- Do you know Logic / Automated Reasoning?
  - Propositional logic: AND, OR, NOT, Boolean SATisfiability
- Do you have basic understanding of Compilers?
  - Control Flow Graphs, Operational Semantics, Intermediate Representation
- Have you used a SAT / Theorem Prover / Constriant Solver / SMT ?

Contact

Office Hours (with Me)

- by appointment
- best time is before/after lectures

Email (email address on the course web page)

- https://ece.uwaterloo.ca/~agurfink/stqam
- Identify yourself
  - Originated from your uwaterloo email address, or
  - Signed with your full name and student ID
- Start **Subject** of email with **[STQAM]**

Use SLACK

A little about me

2007, PhD University of Toronto

2006-2016, Principle Researcher at Software Engineering Institute, Carnegie Mellon University

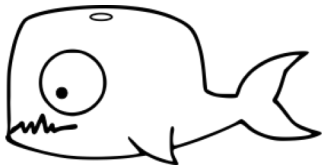
Sep 2016, Associate Professor, University of Waterloo

UFO

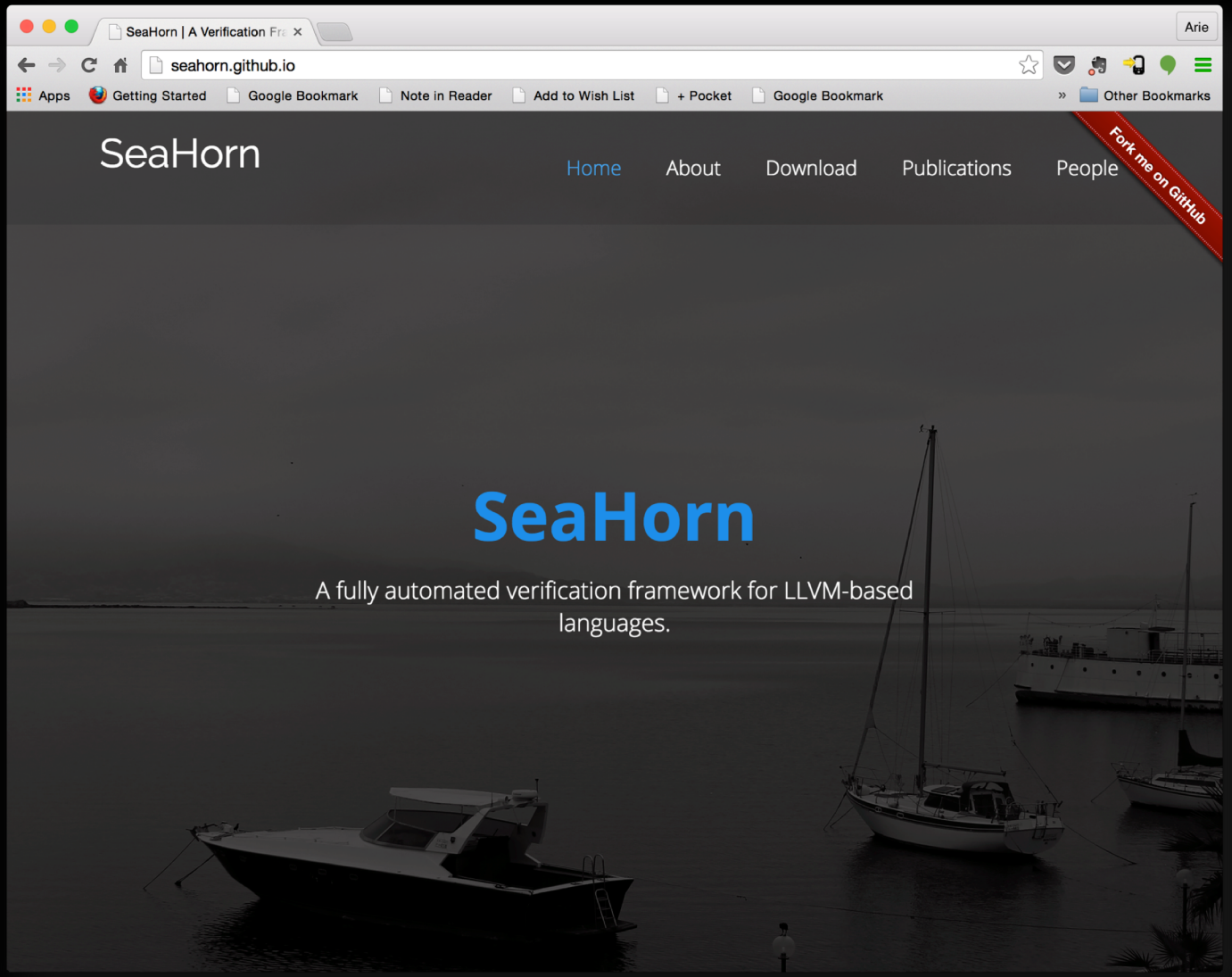
FrankenBit

SPACER

Avy



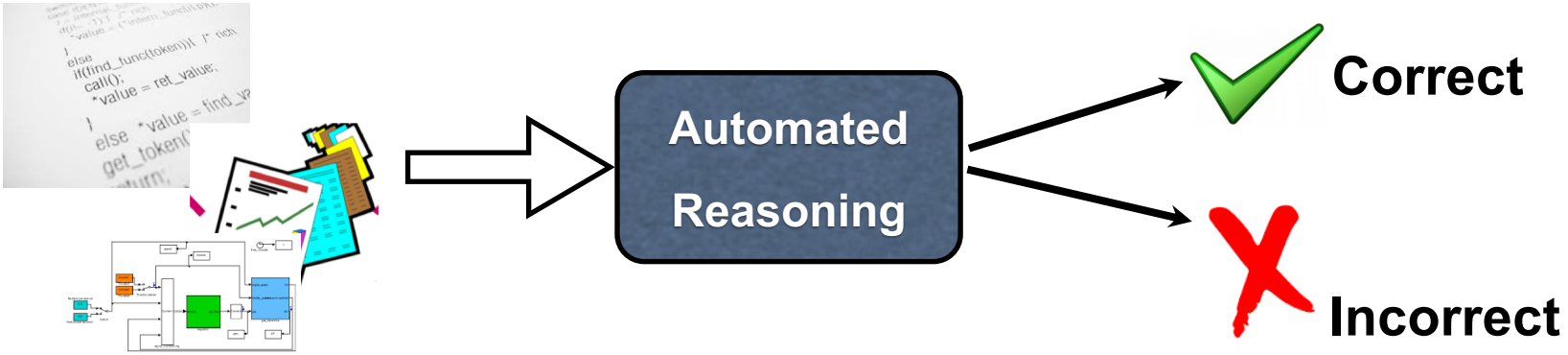
SeaHorn



http://seahorn.github.io

Automated (Software) Verification

Program and/or model



Alan M. Turing. 1936: “Undecidable”

Alan M. Turing. ”Checking a large routine” 1949

How can one check a routine in the sense of making sure that it is right?

programmer should make a number of definite assertions which can be checked individually, and from which the correctness of the whole programme easily follows.

SeaHorn Usage

Example: in test.c, check that x is always greater than or equal to y

test.c

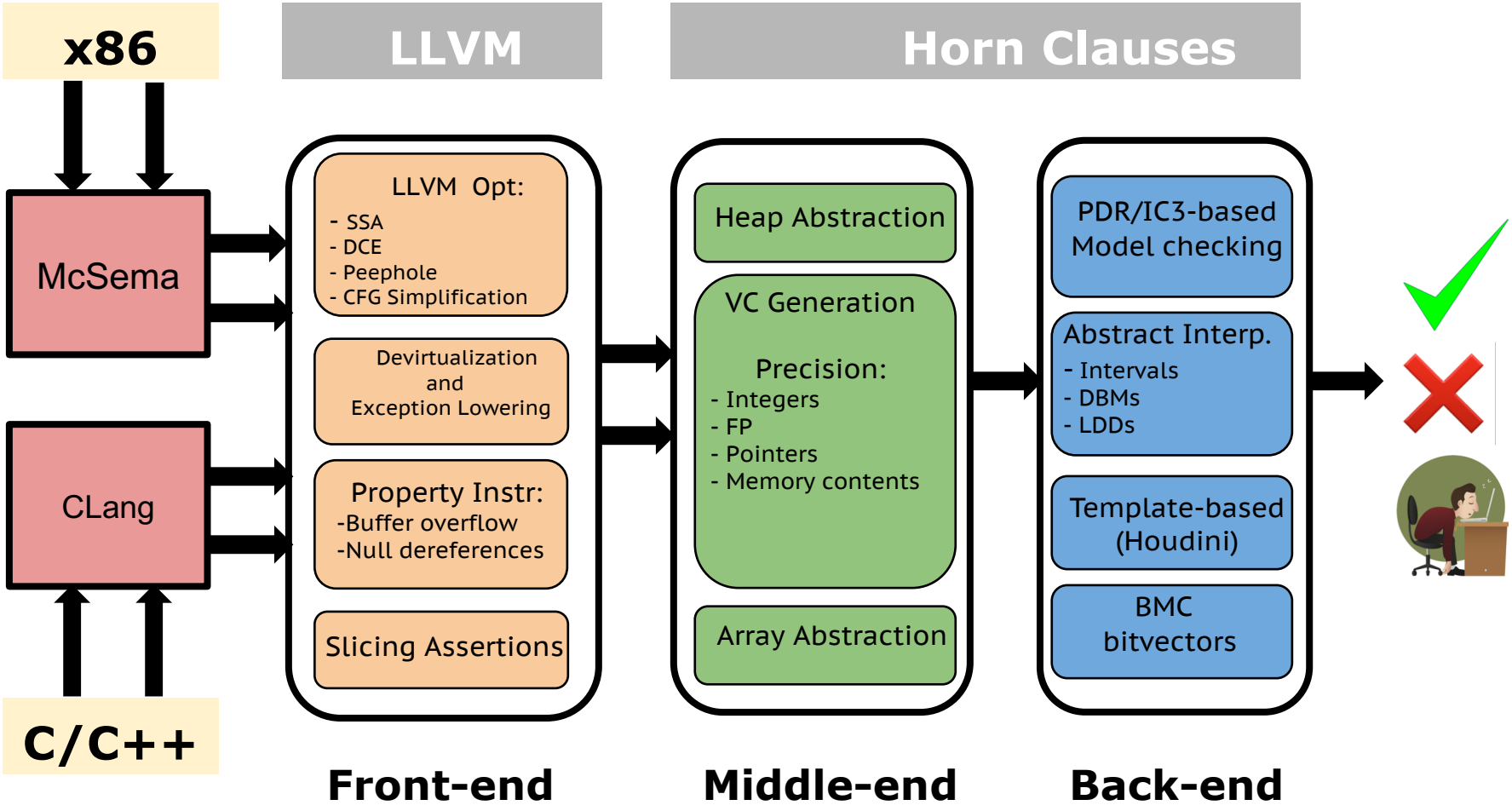
```
extern int nd();
extern void __VERIFIER_error() __attribute__((noreturn));
void assert (int cond) { if (!cond) __VERIFIER_error (); }
int main(){
  int x,y;
  x=1; y=0;
  while (nd ())
  {
    x=x+y;
    y++;
  }
  assert (x>=y);
 return 0;
}
```

SeaHorn command:

```
-> sea pf test.c
```

SeaHorn result:

```
            SEAHORN
-----------------------------
PROPERTY (line 12) | TRUE
-----------------------------
TIME(ms)           |  0.06
```


x86
LLVM
Horn Clauses
McSema
CLang
C/C++
LLVM Opt:
- SSA
- DCE
- Peephole
- CFG Simplification
Devirtualization
and
Exception Lowering
Property Instr:
-Buffer overflow
-Null dereferences
Slicing Assertions
Heap Abstraction
VC Generation
Precision:
- Integers
- FP
- Pointers
- Memory contents
Array Abstraction
PDR/IC3-based
Model checking
Abstract Interp.
- Intervals
- DBMs
- LDDs
Template-based
(Houdini)
BMC
bitvectors
Front-end
Middle-end
Back-end

URA Opportunities in Automated Verification

Our group is working on scalable automated verification that is usable by software engineers and is integrated into the software development life-cycle. Verification ensures that software satisfies given properties for all possible inputs, and can be used to prove absence of bugs.

You will be able to use your programming skills and contribute to the development of the automated verification tool SeaHorn and will learn about Compilers, Constraint Solvers (SAT/SMT), and Automated Verification.

Depending on your interests and experience, you will be able to work on various parts of the framework, and focus on Logic-based Reasoning, Compiler Optimizations, Program Analysis, Pointer Analysis, Testing Infrastructure, Smart Contracts, or C++ Standard Library implementations. The project uses technologies like: C++, Python, LLVM, Z3, Docker, Travis.