# When the Student becomes the Teacher

Marie Farrell[1] Hao Wu[2]

[1]Department of Computer Science, University of Liverpool, UK

[2]Department of Computer Science, Maynooth University, Ireland

FMFun 2019

# Our Contributions

- A report on our experience of teaching formal methods and an analysis of the exam results from the 2018–2019 academic year
- Observations and hypotheses to be further investigated.

# Module Overview

# Module Overview: CS357

- Compulsory for third-year Computer Science BSc students.
- Optional for General Science BSc students.
- Prerequisites: Java programming module and Discrete Structures module.
- Runs over 12 weeks with 2 lecture hours and 2 lab hours per week.

# Module Structure

- Design by Contract (1 week).
- Natural Deduction Proofs and the Coq theorem prover (3 weeks).
- Hoare Logic (2 weeks).
- Spec# (2 weeks).
- SAT/SMT (2 weeks).
- Model Checking (2 weeks).

# Learning Outcomes

Upon completion of the module, students should be able to:

- Explain the role of verification in software engineering.
- Create mathematically precise specifications.
- Prove the correctness of programs using Hoare Logic.
- Use different tools to analyse and verify properties of specifications.

# Assessment

# Assessment

- 30% continuous assessment (CA) and 70% best 3 out of 4 exam questions.
- CA: attend a 2 hour lab session each week to complete weekly assignment which is graded by the lab demonstrators.
- CA: 11 labs in total.

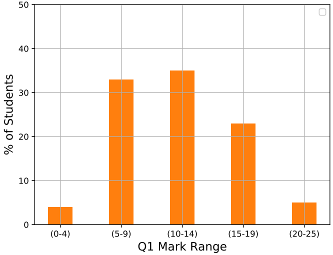
  Lab 1–3: Natural Deduction Proofs with Coq
  Lab 4–5: Hoare Logic
  Lab 6–11: examine a range of verification tools including Spec# and Z3.

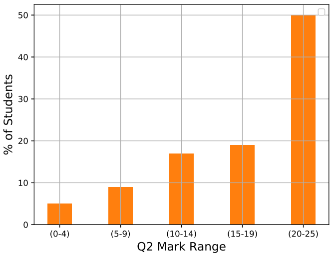
# Exam Structure

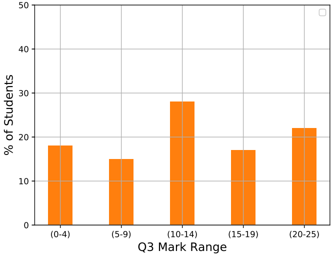
The pen and paper exam is 2 hours long and has the following structure:

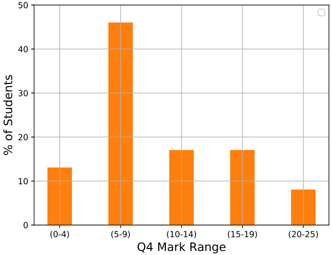
| Question | Examined Topics | Weight (marks) |
|---|---|---|
| Q1 | Design by Contract<br>Propositional and Predicate Logic<br>Natural Deduction Proofs | 25 |
| Q2 | Satisfiability<br>CNF Translation<br>DPLL (Pure literal, Unit clause and Unit propagation) | 25 |
| Q3 | Hoare Logic | 25 |
| Q4 | Basic SMT Encoding<br>Spec# Programming (Pre/Post conditions, Loop invariants)<br>Linear Temporal Logic Encoding | 25 |

# Exam Results (2018–2019)

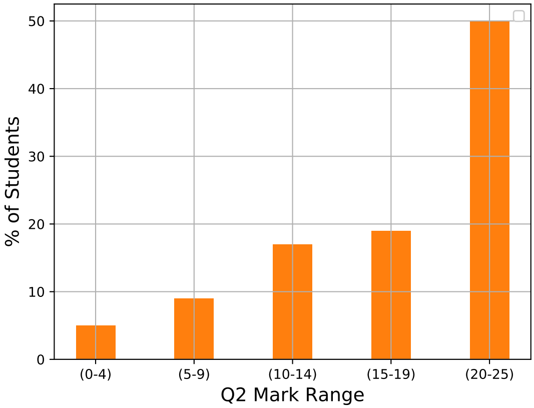


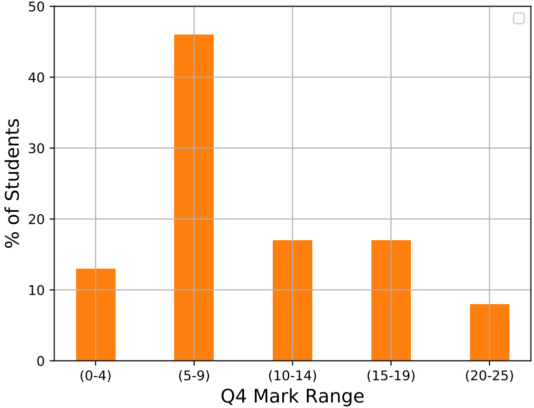



# Exam Results (2018–2019): Q2




- Students performed best on Q2 (satisfiability, CNF translation and DPLL).
- Q2 was the most popular question (88/92).
- Q2 is mechanical in nature (application of rules to formulae).

# Exam Results (2018–2019): Q4




- Students performed worst on Q4 (SMT encoding, Spec#, LTL model-checking).
- Q4 was the least popular question (24/92).
- Q4 was designed to challenge students on the topics:
  - Encode specification into SMT formulas.
  - Write Spec# program with specifications corresponding to C# sum of integer array.
  - SAT encodings for reachability, safety and liveness properties.

# Reflecting on Teaching and Learning

# Our Experience

**Learning:**

- Module is generally challenging for students:
  - identifying loop invariants
  - presenting Hoare Logic proofs
  - understanding low-level SAT/SMT encodings
- Verification tools are not very reliable and don't always give useful feedback.
- Practical applications of formal methods not clear to students.

# Our Experience

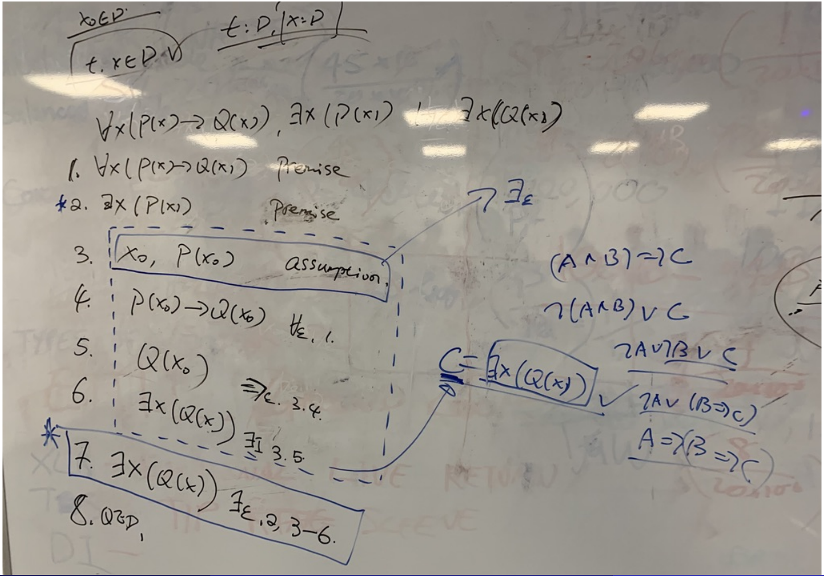
## Learning:

- Module is generally challenging for students:
  - identifying loop invariants
  - presenting Hoare Logic proofs
  - understanding low-level SAT/SMT encodings
- Verification tools are not very reliable and don't always give useful feedback.
- Practical applications of formal methods not clear to students.

## Teaching:

- Difficult to help students see the value in formal methods since they are not widely used in industry.
- Tools are normally not scalable for real-world examples making it difficult to demonstrate their usefulness to students.
- Necessary to combine slides and worked-through whiteboard examples to explain detailed computation steps.

# Whiteboard Example



# Observations

## Observation 1:

- Automated verification tools are not appealing to students.
- Spec# and Z3 (online versions) for lab work.
- Ambiguous feedback makes these tools difficult to use.

# Observations

**Observation 1:**

- Automated verification tools are not appealing to students.
- Spec# and Z3 (online versions) for lab work.
- Ambiguous feedback makes these tools difficult to use.

**Observation 2:**

- Students good at natural deduction but not using Coq.
- Difficult to connect pen and paper proof with Coq proofs.

# Observations

**Observation 3:**

- We did not integrate our own research tools into this module.
- Our work uses SAT/SMT techniques to solve software engineering problems.

# Observations

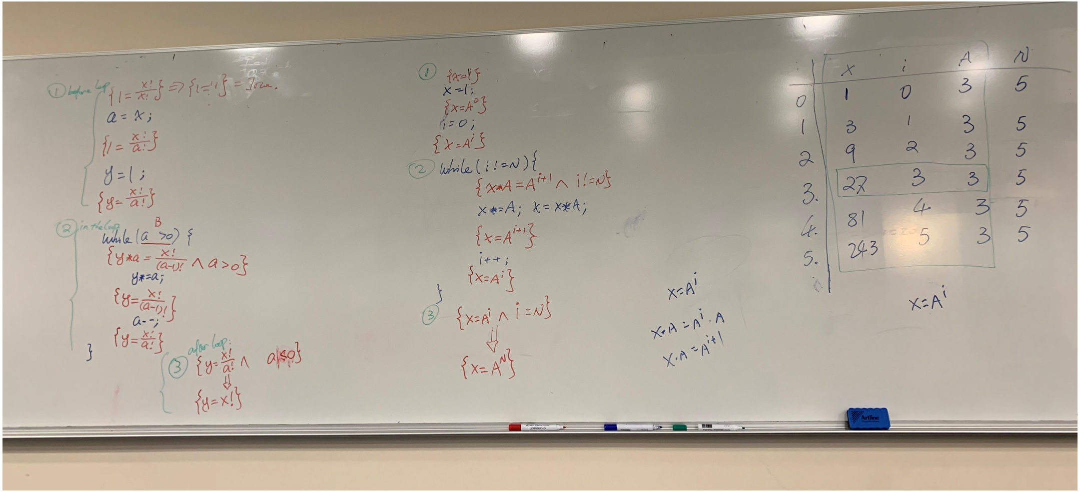
## Observation 3:

- We did not integrate our own research tools into this module.
- Our work uses SAT/SMT techniques to solve software engineering problems.

## Observation 4:

- Mixed reactions to Hoare Logic.
- Used whiteboard examples.
- Students with strong mathematical background performed better on this topic.

# Whiteboard Example



# Observations

- Observations 1 and 2 point to a lack of tool usability. This can also hinder the uptake of formal methods in industry.
- Observation 3 has caused us to integrate our research tool, MaxUSE, into teaching. So far, students seem interested in this.
- Observation 4 noted that students found Hoare Logic difficult but this may have been exacerbated by their difficulty with some of the tools e.g. Spec#.

# Hypotheses

We intend to investigate the following hypotheses in the current (2019–2020) academic year:

## Hypothesis 1:

- The development of an online repository of real-world examples would be useful for both teaching and illustrating industrial uses of formal methods.
- Action: design and distribute a survey among past students to identify the most interesting and educational examples to be used in class.

## Hypothesis 2:

- A platform that turns logical reasoning proofs into games would increase the interactions between students and lecturers.
- Action: a live coding session using SMT solvers to solve a Sudoku.

# Conclusions and Future Work

- We discussed our own experience of both studying and teaching the same software verification module at Maynooth University.
- Based on the 2018–2019 results, we made several observations from which we have constructed two hypotheses that we intend to investigate in the current (2019–2020) academic year.
- We plan to work with the education research group at Maynooth to develop interesting experiments in order to improve our teaching of this course.
- Maynooth offers a similar module at Master's level and we plan to investigate it and compare the results to this paper.

# Questions?