COMP782

Advanced Topics in Theory and Practice of Software

S1 Day 2017

Dept of Computing

Contents

General Information

Unit convenor and teaching staff
Convenor and lecturer
Anthony Sloane
anthony.sloane@mq.edu.au
Contact via Email
E6A315
By appointment

Mark Dras
mark.dras@mq.edu.au

Credit points
4

Prerequisites
Admission to MRes

Corequisites

Co-badged status

Unit description
This unit introduces the formal study of software systems. It is intended to provide a general basis for further study or research in software-focused areas of Computer Science such as Programming Languages and Formal Methods. The unit is organised around two main themes: a) the meaning of languages and programs, and b) techniques for verifying that languages and programs have desired properties. The practical work in the unit includes implementation of formal language semantics and development of verification proofs.

Important Academic Dates

Information about important academic dates including deadlines for withdrawing from units are available at https://www.mq.edu.au/study/calendar-of-dates

Learning Outcomes

On successful completion of this unit, you will be able to:

Understand how to use mathematical techniques to specify the meaning of programming languages and programs.

Ability to use a practical tool or system to specify and prove formal properties of programming languages and programs.

Ability to investigate a topic in advanced computer science and report findings in written and oral form.

Assessment Tasks

Name	Weighting	Hurdle	Due
Weekly Homework	20%	No	Weeks 2 to 13
Class participation	10%	No	Every week
Research Report	30%	No	Week 12
Presentation	10%	No	Week 13
Examination	30%	No	Exam period

Weekly Homework

Due: **Weeks 2 to 13**
Weighting: **20%**

Each week students will be asked to complete some Coq exercises based on the class material of that week. This mark will be allocated on the basis of the correctness and style of the submitted exercise solutions with reference to the difficulty of the questions.

On successful completion you will be able to:

- Understand how to use mathematical techniques to specify the meaning of programming languages and programs.
- Ability to use a practical tool or system to specify and prove formal properties of programming languages and programs.

Class participation

Due: **Every week**
Weighting: **10%**

Students are expected to participate in class discussions. The form that this will take can vary but will include presentation of solutions to selected weekly homework exercises plus asking or answering questions that arise during the classes. Marks will be allocated that reflect both the quantity and quality of the student's contribution to class.

On successful completion you will be able to:

- Understand how to use mathematical techniques to specify the meaning of programming languages and programs.

- Ability to use a practical tool or system to specify and prove formal properties of programming languages and programs.

Research Report

Due: **Week 12**
Weighting: **30%**

Students will be asked to investigate a theorem prover, proof assistant, software verification system or model checker other than Coq. A report must be written that describes the system that has been investigated, illustrates an example verification or proof with that system, and compares the strengths and weaknesses of the system to Coq. The report will be assessed on the basis of the understandability and correctness of the descriptions and coverage of the above points.

On successful completion you will be able to:

- Understand how to use mathematical techniques to specify the meaning of programming languages and programs.
- Ability to investigate a topic in advanced computer science and report findings in written and oral form.

Presentation

Due: **Week 13**
Weighting: **10%**

A presentation on the topic of the research report. Presentations will be thirty minutes long, including five minutes for questions, and will be held in class in Week 13. The presentation will be assessed on the basis of the form in which you present the information from your report, the clarity of your explanations, and the way in which you respond to questions from the audience.

On successful completion you will be able to:

- Understand how to use mathematical techniques to specify the meaning of programming languages and programs.
- Ability to investigate a topic in advanced computer science and report findings in written and oral form.

Examination

Due: **Exam period**
Weighting: **30%**

In the exam period, students will be given a week to undertake a non-trivial software formalisation and proof task using Coq. The mark for this assessment will be determined as for the weekly homework with the total mark for the exam determined by combining the individual

question marks according to the weights specified in the exam paper.

On successful completion you will be able to:

- Understand how to use mathematical techniques to specify the meaning of programming languages and programs.
- Ability to use a practical tool or system to specify and prove formal properties of programming languages and programs.

Delivery and Resources

CLASSES

Each week of COMP782 has three hours of face-to-face class. Classes will be a mixture of lecture-style presentation, discussion and practical demonstration.

REQUIRED AND RECOMMENDED TEXTS AND/OR MATERIALS

COMP782 will follow the book Software Foundations by Benjamin Pierce et al. (http://www.cis.upenn.edu/~bcpierce/sf/). The book is freely available from the web site and consists of annotated programs, proofs and exercises. It can be read in HTML form or the full distribution can be downloaded for formatting as PDF. This book is updated from time to time.

We will use Version 4.0 (May 2016) of the book.

Students should read the relevant sections of the book and attempt the basic exercises on their own. Class time will be devoted to the main ideas of each chapter and working through a number of exercises. Some basic and more advanced exercises will be set as homework exercises.

UNIT WEBPAGE AND TECHNOLOGY USED AND REQUIRED

COMP782 uses iLearn for delivery of class materials, discussion boards, online selftests, submission of assessment tasks and access to marks and comments. Students should check the iLearn site regularly for unit updates.

Questions regarding the content of this unit should be posted to the appropriate discussion board on iLearn. In particular, any questions which are of interest to all students in this unit should be posted to one of these discussion boards, so that everyone can benefit from the answers.

Coq Proof Assistant

The practical work in this unit involves functional programming and proof construction using the Coq proof assistant (http://coq.inria.fr). Ideally, students should install Coq on a laptop and bring it to class so they can follow along with in-class exercises.

We will use Coq version 8.6.

Coq is usually used via a simple IDE platform of which the easiest one is the CoqIDE that is

available as part of the Coq distribution. Students who are familiar with Emacs may want to look at the Proof General interface which is similar to CoqIDE but embedded in Emacs.

Unit Schedule

The unit introduces the formal study of software systems. It is intended to provide a general basis for further study or research in software-focused areas of Computer Science such as Programming Languages and Formal Methods.

The unit is organised around two main themes:

a) The meaning of programs. To study software systems it is necessary to have a proper understanding of the programming languages in which those systems are written. Formal semantic descriptions of languages assign mathematical meanings to programs. Typical kinds of meaning that will be studied include types that specify the operations that programs will perform, and operational aspects that capture how a program behaves as it executes.

b) Techniques for verifying that languages and programs have desired properties. We will see how to analyse a language semantics to prove that all programs written in that language have desirable properties. E.g., a desirable property of a type system is type safety: that an executing program cannot execute an illegal operation. We will also study the properties of particular programs. E.g., it is often desirable to be able to prove that a program produces a desired result.

The practical work in the unit will include implementation of formal language semantics and development of verification proofs. We emphasise the use of frameworks and tools to assist with both of these activities. Examples include the use of software language engineering tools and libraries to assist with language implementation, and the use of proof assistants, program verification systems or model checkers to help us specify properties and to find proofs.

Students entering this unit should have reasonable programming experience and should have studied discrete mathematics. Relevant Macquarie University units for programming maturity are COMP225 Algorithms and Data Structures, COMP229 Object-Oriented Programming Practices and any 300-level unit that applies programming skills to particular problem domains (e.g., COMP330 Computer Graphics, COMP333 Algorithm Theory and Design, and ISYS302 Advanced Application Development). DMTH137 Discrete Mathematics I and DMTH237 Discrete Mathematics II provide good mathematical background. Previous courses in areas such as programming language concepts or implementation, such as COMP332 Programming Languages, will be helpful, but are not required.

The class material will be structured according to the following schedule. The rightmost column refers to the relevant chapters of the "Software Foundations" required text. Note that there will be no class in Week 5 and Week 7 due to the Easter Monday and Anzac Day holidays, respectively.

Week	Topic	Chapter of Text
1	Introduction, Functional Programming	Preface, Basics
2	Proof by Induction, Structured Data	Induction, Lists

3	Polymorphism and Higher-Order Functions	Poly
4	More basic tactics	Tactics
5	Logical Reasoning	Logic
6	Inductive Propositions	IndProp
7-8	Maps, Imperative Programming Language	Maps, Imp
9	Program Equivalence	Equiv
10	Hoare Logic for Imperative Programs	Hoare
11	Smallstep Operational Semantics	Smallstep
12	Simply-Typed Lambda Calculus	Stlc, StlcProp (overview)
13	Presentations, Review	

Policies and Procedures

Macquarie University policies and procedures are accessible from Policy Central. Students should be aware of the following policies in particular with regard to Learning and Teaching:

Academic Honesty Policy http://mq.edu.au/policy/docs/academic_honesty/policy.html

Assessment Policy http://mq.edu.au/policy/docs/assessment/policy_2016.html

Grade Appeal Policy http://mq.edu.au/policy/docs/gradeappeal/policy.html

Complaint Management Procedure for Students and Members of the Public http://www.mq.edu.au/policy/docs/complaint_management/procedure.html

Disruption to Studies Policy (in effect until Dec 4th, 2017): http://www.mq.edu.au/policy/docs/disruption_studies/policy.html

Special Consideration Policy (in effect from Dec 4th, 2017): https://staff.mq.edu.au/work/strategy-planning-and-governance/university-policies-and-procedures/policies/special-consideration

In addition, a number of other policies can be found in the Learning and Teaching Category of Policy Central.

Student Code of Conduct

Macquarie University students have a responsibility to be familiar with the Student Code of Conduct: https://students.mq.edu.au/support/student_conduct/

Results

Results shown in *iLearn*, or released directly by your Unit Convenor, are not confirmed as they are subject to final approval by the University. Once approved, final results will be sent to your student email address and will be made available in eStudent. For more information visit ask.mq.edu.au.

Student Support

Macquarie University provides a range of support services for students. For details, visit http://students.mq.edu.au/support/

Learning Skills

Learning Skills (mq.edu.au/learningskills) provides academic writing resources and study strategies to improve your marks and take control of your study.

- Workshops
- StudyWise
- Academic Integrity Module for Students
- Ask a Learning Adviser

Student Services and Support

Students with a disability are encouraged to contact the Disability Service who can provide appropriate help with any issues that arise during their studies.

Student Enquiries

For all student enquiries, visit Student Connect at ask.mq.edu.au

IT Help

For help with University computer systems and technology, visit http://www.mq.edu.au/about_us/offices_and_units/information_technology/help/.

When using the University's IT, you must adhere to the Acceptable Use of IT Resources Policy. The policy applies to all who connect to the MQ network including students.

Graduate Capabilities

PG - Discipline Knowledge and Skills

Our postgraduates will be able to demonstrate a significantly enhanced depth and breadth of knowledge, scholarly understanding, and specific subject content knowledge in their chosen fields.

This graduate capability is supported by:

Learning outcomes

- Understand how to use mathematical techniques to specify the meaning of programming languages and programs.
- Ability to use a practical tool or system to specify and prove formal properties of programming languages and programs.
- Ability to investigate a topic in advanced computer science and report findings in written

and oral form.

Assessment tasks

- Weekly Homework
- Class participation
- Research Report
- Presentation
- Examination

PG - Critical, Analytical and Integrative Thinking

Our postgraduates will be capable of utilising and reflecting on prior knowledge and experience, of applying higher level critical thinking skills, and of integrating and synthesising learning and knowledge from a range of sources and environments. A characteristic of this form of thinking is the generation of new, professionally oriented knowledge through personal or group-based critique of practice and theory.

This graduate capability is supported by:

Learning outcomes

- Understand how to use mathematical techniques to specify the meaning of programming languages and programs.
- Ability to use a practical tool or system to specify and prove formal properties of programming languages and programs.
- Ability to investigate a topic in advanced computer science and report findings in written and oral form.

Assessment tasks

- Weekly Homework
- Class participation
- Research Report
- Presentation
- Examination

PG - Research and Problem Solving Capability

Our postgraduates will be capable of systematic enquiry; able to use research skills to create new knowledge that can be applied to real world issues, or contribute to a field of study or practice to enhance society. They will be capable of creative questioning, problem finding and problem solving.

This graduate capability is supported by:

Learning outcomes

- Understand how to use mathematical techniques to specify the meaning of programming languages and programs.
- Ability to use a practical tool or system to specify and prove formal properties of programming languages and programs.
- Ability to investigate a topic in advanced computer science and report findings in written and oral form.

Assessment tasks

- Weekly Homework
- Research Report
- Presentation
- Examination

PG - Effective Communication

Our postgraduates will be able to communicate effectively and convey their views to different social, cultural, and professional audiences. They will be able to use a variety of technologically supported media to communicate with empathy using a range of written, spoken or visual formats.

This graduate capability is supported by:

Learning outcomes

- Understand how to use mathematical techniques to specify the meaning of programming languages and programs.
- Ability to investigate a topic in advanced computer science and report findings in written and oral form.

Assessment tasks

- Weekly Homework
- Class participation
- Research Report
- Presentation
- Examination

Changes from Previous Offering

There are some topic changes and rearrangements due to changes in the Software Foundations text since the last offering, but broadly speaking the same ground is covered. The assessments have been re-weighted to reflect concerns from the 2016 unit review. In particular, the homework

weight has been reduced and class participation including presentation of homework solutions has been added to ensure that students understand the work that they submit. The research report weight has been increased to more accurately reflect the significant amount of work that we expect.

Assessment Standards

COMP782 will be assessed and graded according to the University assessment and grading policies.

Submission Deadlines

Assessment deadlines are strict, unless an application for special consideration is received (preferably in advance) accompanied by appropriate documentary evidence. Late submissions will be penalised at the rate of 20% of the full marks for the assessment per day or part thereof late.

Standards

The following general standards of achievement will be used to assess each of the assessment tasks with respect to the letter grades.

Pass: Has a basic undestanding of language and program semantics as discussed in class. Can use a verification tool or proof system to specify and prove simple language and program properties similar to those discussed in class. Can perform a basic research investigation in the area and present the results of that research in rudimentary written and oral forms.

Credit: As for Pass plus: Is able to apply the techniques we have discussed to specify and prove new properties that are not direct analogues of ones discussed in class. Shows more than basic insights into the results of a research investgiation and is able to communicate those insights.

Distinction/High Distinction: As for Credit plus: Is able to generalise from the language and program properties discussed explicity to new ones for problem domains not explicitly discussed in class and can apply verification tools and systems to proofs about them. Can critically evaluate the limits of the techniques and tools we have discussed.

Assessment Process

These assessment standards will be used to give a numeric mark out of 100 to each assessment submission during marking. The mark will correspond to a letter grade for that task according to the University guidelines. The final mark for the unit will be calculated by combining the marks for all assessment tasks according to the percentage weightings shown in the assessment summary.