COMP7010

Advanced Topics in Theory and Practice of Software

Session 1, Special circumstances 2021

School of Computing

Contents

General Information 2

Learning Outcomes 2

General Assessment Information 2

Assessment Tasks 4

Delivery and Resources 6

Unit Schedule 7

Policies and Procedures 8

Changes from Previous Offering 10

Disclaimer

Macquarie University has taken all reasonable measures to ensure the information in this publication is accurate and up-to-date. However, the information may change or become out-dated as a result of change in University policies, procedures or rules. The University reserves the right to make changes to any information in this publication without notice. Users of this publication are advised to check the website version of this publication [or the relevant faculty or department] before acting on any information in this publication.

Notice

As part of Phase 3 of our return to campus plan, most units will now run tutorials, seminars and other small group activities on campus, and most will keep an online version available to those students unable to return or those who choose to continue their studies online.

To check the availability of face-to-face activities for your unit, please go to timetable viewer. To check detailed information on unit assessments visit your unit's iLearn space or consult your unit convenor.

General Information

Unit convenor and teaching staff Lecturer, Convener Matthew Roberts matthew.roberts@mq.edu.au
Credit points 10
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:

ULO1: Understand, specify, analyse and evaluate the meaning of programming languages and programs using mathematical techniques.

ULO2: Apply a practical tool or system to specify and prove formal properties of programming languages and programs.

ULO3: Research a topic in advanced computer science and report findings in written and oral form.

General Assessment Information

COMP7010 will be assessed and graded according to the University assessment and grading

policies.

Late Submission

No extensions will be granted without an approved application for Special Consideration. There will be a deduction of 10% of the total available marks made from the total awarded mark for each 24 hour period or part thereof that the submission is late. For example, 25 hours late in submission for an assignment worth 10 marks – 20% penalty or 2 marks deducted from the total. No submission will be accepted after solutions have been posted.

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.

Special Consideration

If you receive special consideration for the final exam, a supplementary exam will be scheduled in the interval between the regular exam period and the start of the next session. By making a special consideration application for the final exam you are declaring yourself available for a resit during the supplementary examination period and will not be eligible for a second special consideration approval based on pre-existing commitments.Please ensure you are familiar with the policy prior to submitting an application. Approved applicants will receive an individual notification at least one week prior to the exam with the exact date and time of their supplementary examination.

Assessment Tasks

Name	Weighting	Hurdle	Due
Weekly Homework	20%	No	weekly
Research Presentation	20%	No	week 13
Final Examination	30%	No	exam period
Research Report	30%	No	week 13

Weekly Homework

Assessment Type [1]: Programming Task
Indicative Time on Task [2]: 48 hours
Due: **weekly**
Weighting: **20%**

Each week students will be asked to complete some practical 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, specify, analyse and evaluate the meaning of programming languages and programs using mathematical techniques.
- Apply a practical tool or system to specify and prove formal properties of programming languages and programs.

Research Presentation

Assessment Type [1]: Presentation
Indicative Time on Task [2]: 5 hours
Due: **week 13**
Weighting: **20%**

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, specify, analyse and evaluate the meaning of programming languages and programs using mathematical techniques.
- Apply a practical tool or system to specify and prove formal properties of programming languages and programs.
- Research a topic in advanced computer science and report findings in written and oral form.

Final Examination

Assessment Type [1]: Examination
Indicative Time on Task [2]: 7 hours
Due: **exam period**
Weighting: **30%**

In the final examination period, students will be given a week to undertake a non-trivial software formalisation and proof task using the tool(s) studied in this unit. 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, specify, analyse and evaluate the meaning of programming languages and programs using mathematical techniques.
- Apply a practical tool or system to specify and prove formal properties of programming languages and programs.

Research Report

Assessment Type [1]: Report
Indicative Time on Task [2]: 30 hours
Due: **week 13**
Weighting: **30%**

Students will be asked to investigate a theorem prover, proof assistant, software verification system or model checker other than the one being used in the practical work of this unit. 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 the system used in this unit. 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, specify, analyse and evaluate the meaning of programming languages and programs using mathematical techniques.
- Apply a practical tool or system to specify and prove formal properties of programming languages and programs.
- Research a topic in advanced computer science and report findings in written and oral form.

---

[1] If you need help with your assignment, please contact:

- the academic teaching staff in your unit for guidance in understanding or completing this type of assessment
- the Writing Centre for academic skills support.

[2] Indicative time-on-task is an estimate of the time required for completion of the assessment task and is subject to individual variation

Delivery and Resources

CLASSES

Each week of COMP7010 has two to 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

COMP7010 will follow the book "Software Foundations" by Benjamin Pierce et al. The book 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 host a custom version of the text at softwaretechnology.mattr.net.au

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

COMP7010 uses iLearn for delivery of class materials, discussion boards, online self-tests, 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 8.13

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. Another option is VsCoq which is a Coq extension for the Visual Studio Code editor.

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 COMP2000 Object-Oriented Programming Practices, COMP2010 Algorithms and Data Structures, and any 3000-level unit that applies programming skills to particular problem domains (e.g., COMP3170 Computer Graphics or COMP3010 Algorithm Theory and Design). MATH1007 Discrete Mathematics I and MATH2907 Discrete Mathematics II provide good

mathematical background. Previous courses in areas such as programming language concepts or implementation, such as COMP3000 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.

Week	Topic	Chapter(s)
	Introduction, Functional Programming	Preface, Basics
	Proof by Induction, Structured Data	Induction, Lists
	Polymorphism and Higher-Order Functions	Poly
	More Basic Tactics	Tactics
	Logical Reasoning	Logic
	Inductive Propositions	IndProp
	Maps, Imperative Programming Language	Maps, Imp
	Program Equivalence	Equiv
	Hoare Logic for Imperative Programs	Hoare
	Smallstep Operational Semantics	Smallstep
	Simple-typed Lambda Calculus	Stlc, StlcProp
	Student Presentations	

Policies and Procedures

Macquarie University policies and procedures are accessible from Policy Central (https://policies.mq.edu.au). Students should be aware of the following policies in particular with regard to Learning and Teaching:

- Academic Appeals Policy
- Academic Integrity Policy
- Academic Progression Policy
- Assessment Policy
- Fitness to Practice Procedure
- Grade Appeal Policy
- Complaint Management Procedure for Students and Members of the Public
- Special Consideration Policy

Students seeking more policy resources can visit Student Policies (https://students.mq.edu.au/support/study/policies). It is your one-stop-shop for the key policies you need to know about throughout your undergraduate student journey.

To find other policies relating to Teaching and Learning, visit Policy Central (https://policies.mq.edu.au) and use the search tool.

Student Code of Conduct

Macquarie University students have a responsibility to be familiar with the Student Code of Conduct: https://students.mq.edu.au/admin/other-resources/student-conduct

Results

Results published on platform other than eStudent, (eg. iLearn, Coursera etc.) 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 or if you are a Global MBA student contact globalmba.support@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 help you improve your marks and take control of your study.

- Getting help with your assignment
- Workshops
- StudyWise
- Academic Integrity Module

The Library provides online and face to face support to help you find and use relevant information resources.

- Subject and Research Guides
- Ask a Librarian

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

If you are a Global MBA student contact globalmba.support@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.

Changes from Previous Offering

none