COMP332

Programming Languages

S2 Day 2016

Dept of Computing

Contents

General Information	2
Learning Outcomes	3
General Assessment Information	3
Assessment Tasks	3
Delivery and Resources	6
Unit Schedule	7
Policies and Procedures	8
Graduate Capabilities	9
Changes from Previous Offering	13
Assessment Standards	13

General Information

Unit convenor and teaching staff
Unit Convenor, Lecturer
Matthew Roberts
matthew.roberts@mq.edu.au
Contact via 98509564
E6A374
Fri 12-2pm

Lecturer
Kym Haines
kym.haines@mq.edu.au

Tutor
Scott Buckley

Credit points
3

Prerequisites
39cp including(COMP225 or COMP229)

Corequisites

Co-badged status

Unit description
Formal languages play a central role in modern software development. Programming languages such as Java and C++ allow developers to express their algorithms and data structures. Compilers and interpreters transform programs into running software. Data languages such as XML and JSON are widely used to transfer information between systems. This unit studies software languages by looking at how they are used in software development. Students will study how to formally understand the syntax, semantics and translation of software languages. Practical exercises involve writing software language processors of various kinds such as simple compilers or data transformation tools.

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:

Explain the role that languages play in software development and describe a spectrum of software languages that are in current use.

Express properties of software languages using formal notations.

Translate formal notations of software language properties into implementations of language processors.

Demonstrate that a language processor is operating correctly by construction and use of appropriate test cases.

General Assessment Information

ASSIGNMENTS

The unit has three assignments which together constitute the implementation of a small but non-trivial programming language using Scala and the Kiama library.

Assessment deadlines are strict, unless notification of disruption 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.

EXAMINATIONS

The unit has three examinations corresponding to weeks 1-4, 5-8, and 9-12, respectively. Each examination is offered twice: once during the mixed classes in weeks 6, 10 and 13, respectively, and once in the final examination period. The repeat offerings of the examination will not be identical examinations but will be designed to assess the same material.

If a student attempts an examination more than once then the higher of their marks for the two attempts will be used to compute the grade.

MIXED CLASSES

Each week tutorial and practical exercises will be set for the mixed classes. These exercises are not to be submitted and do not attract any marks. They are designed to prepare students for the examinations and the assignments.

Assessment Tasks

Name	Weighting	Due
Assignment 1: Syntax Analysis	10%	Week 4/5
Examination 1 (Weeks 1-4)	20%	Mixed Class Week 6

Name	Weighting	Due
Assignment 2:Semantic Analysis	15%	Week 8/9
Examination 2 (Weeks 5-8)	20%	Mixed Class Week 10
Assignment 3: Translation	15%	Week 11/12
Examination 3 (Weeks 9-12)	20%	Mixed Class Week 13
Final Examination	0%	Exam Period

Assignment 1: Syntax Analysis

Due: **Week 4/5**
Weighting: **10%**

The first assignment focuses on processing the syntax of a language to obtain a representation that the rest of the implementation can use.

On successful completion you will be able to:

- Express properties of software languages using formal notations.
- Translate formal notations of software language properties into implementations of language processors.
- Demonstrate that a language processor is operating correctly by construction and use of appropriate test cases.

Examination 1 (Weeks 1-4)

Due: **Mixed Class Week 6**
Weighting: **20%**

This examination will assess the material from Weeks 1-4 of the semester. The examination will be held in the first hour of the mixed class in this week.

On successful completion you will be able to:

- Explain the role that languages play in software development and describe a spectrum of software languages that are in current use.
- Express properties of software languages using formal notations.

Assignment 2:Semantic Analysis

Due: **Week 8/9**
Weighting: **15%**

The second assignment focuses on the semantics of a language by processing the

representation produced by syntax analysis to check language rules and provide information needed for later processing.

On successful completion you will be able to:

- Express properties of software languages using formal notations.
- Translate formal notations of software language properties into implementations of language processors.
- Demonstrate that a language processor is operating correctly by construction and use of appropriate test cases.

Examination 2 (Weeks 5-8)

Due: **Mixed Class Week 10**
Weighting: **20%**

This examination will assess the material from Weeks 5-8 of the semester. The examination will be held in the first hour of the mixed class in this week.

On successful completion you will be able to:

- Explain the role that languages play in software development and describe a spectrum of software languages that are in current use.
- Express properties of software languages using formal notations.

Assignment 3: Translation

Due: **Week 11/12**
Weighting: **15%**

The third assignment focuses on translating a language into some other form, such as another structured language (e.g., translating a programming language into a lower-level form such as bytecode or assembly language).

On successful completion you will be able to:

- Express properties of software languages using formal notations.
- Translate formal notations of software language properties into implementations of language processors.
- Demonstrate that a language processor is operating correctly by construction and use of appropriate test cases.

Examination 3 (Weeks 9-12)

Due: **Mixed Class Week 13**
Weighting: **20%**

This examination will assess the material from Weeks 9-12 of the semester. The examination will be held in the first hour of the mixed class in this week.

On successful completion you will be able to:

- Explain the role that languages play in software development and describe a spectrum of software languages that are in current use.
- Express properties of software languages using formal notations.

Final Examination

Due: **Exam Period**
Weighting: **0%**

The final examination will be a chance to sit any or all of Examinations 1, 2 and 3 again. The final examinations will not be identical to the earlier examinations but will be designed to assess the same material.

On successful completion you will be able to:

- Explain the role that languages play in software development and describe a spectrum of software languages that are in current use.
- Express properties of software languages using formal notations.

Delivery and Resources

CLASSES

Each week of COMP332 has three hours of lecture and a two-hour mixed class. The mixed classes will require a mixture of tutorial-style and practical work. Mixed classes start in Week 1.

REQUIRED AND RECOMMENDED TEXTS AND/OR MATERIALS

There is no required text. We will provide notes or references to freely available materials where relevant.

Students may find it useful to consult one of the many books that are available on the programming languages topic. The following books are among those that are available in the Macquarie University Library:

- Programming Language Pragmatics. Scott.
- Principles of programming languages: design, evaluation, and implementation. MacLennan.
- Programming languages: design and implementation. Pratt and Zelkowitz.
- Concepts of programming languages. Sebesta.
- Programming languages: concepts and constructs. Sethi.

- Introduction to compiler construction. Waite and Carter.
- Compilers: principles, techniques and tools. Aho, Sethi, and Ullman.
- Modern compiler implementation in Java. Appel.

UNIT WEBPAGE AND TECHNOLOGY USED AND REQUIRED

COMP332 Web Home Page: http://www.comp.mq.edu.au/units/comp332/

COMP332 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, its tutorials or practicals 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.

The practical work in this unit involves programming in the Scala language (http:// www.scala-lang.org) which will give students experience with modern programming language features that we expect to see in mainstream languages in the future.

We will also use the Kiama language processing library (http://kiama.googlecode.com) that is being developed by our Programming Languages Research Group. Kiama provides high-level facilities for writing processors such as compilers in Scala and makes it possible for students to implement of a language from scratch within the semester.

Instructions will be provided on how to use Scala and Kiama on the laboratory machines and how to download it for use on your own machines

Unit Schedule

Week	Topic	Assignment Due	Exam
	Introduction: Unit and Scala		
	Syntax		
	Names		
	Semantic analysis	One	
	Scala for compiler writing		
	Types		One
	Special Topic		
	Lecture Recess - Two Weeks		

Week	Topic	Assignment Due	Exam
	Transformation; compilation	Two	
	Language runtimes; interpretation		
	Functional programming		Two
	Scripting languages	Three	
	Domain-specific Languages		
	Review, Exam Discussion		Three
			One, Two, Three

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

New Assessment Policy in effect from Session 2 2016 http://mq.edu.au/policy/docs/assessment/policy_2016.html. For more information visit http://students.mq.edu.au/events/2016/07/19/new_assessment_policy_in_place_from_session_2/

Assessment Policy prior to Session 2 2016 http://mq.edu.au/policy/docs/assessment/policy.html

Grading Policy prior to Session 2 2016 http://mq.edu.au/policy/docs/grading/policy.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 http://www.mq.edu.au/policy/docs/disruption_studies/policy.html *The Disruption to Studies Policy is effective from March 3 2014 and replaces the Special Consideration Policy.*

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

Creative and Innovative

Our graduates will also be capable of creative thinking and of creating knowledge. They will be imaginative and open to experience and capable of innovation at work and in the community. We want them to be engaged in applying their critical, creative thinking.

This graduate capability is supported by:

Learning outcomes

- Express properties of software languages using formal notations.
- Translate formal notations of software language properties into implementations of language processors.
- Demonstrate that a language processor is operating correctly by construction and use of appropriate test cases.

Assessment tasks

- Assignment 1: Syntax Analysis
- Examination 1 (Weeks 1-4)
- Assignment 2:Semantic Analysis
- Examination 2 (Weeks 5-8)
- Assignment 3: Translation
- Examination 3 (Weeks 9-12)
- Final Examination

Discipline Specific Knowledge and Skills

Our graduates will take with them the intellectual development, depth and breadth of knowledge, scholarly understanding, and specific subject content in their chosen fields to make them competent and confident in their subject or profession. They will be able to demonstrate, where relevant, professional technical competence and meet professional standards. They will be able to articulate the structure of knowledge of their discipline, be able to adapt discipline-specific knowledge to novel situations, and be able to contribute from their discipline to inter-disciplinary solutions to problems.

This graduate capability is supported by:

Learning outcomes

- Explain the role that languages play in software development and describe a spectrum of software languages that are in current use.
- Express properties of software languages using formal notations.
- Translate formal notations of software language properties into implementations of language processors.
- Demonstrate that a language processor is operating correctly by construction and use of appropriate test cases.

Assessment tasks

- Assignment 1: Syntax Analysis
- Examination 1 (Weeks 1-4)

- Assignment 2:Semantic Analysis
- Examination 2 (Weeks 5-8)
- Assignment 3: Translation
- Examination 3 (Weeks 9-12)
- Final Examination

Critical, Analytical and Integrative Thinking

We want our graduates to be capable of reasoning, questioning and analysing, and to integrate and synthesise learning and knowledge from a range of sources and environments; to be able to critique constraints, assumptions and limitations; to be able to think independently and systemically in relation to scholarly activity, in the workplace, and in the world. We want them to have a level of scientific and information technology literacy.

This graduate capability is supported by:

Learning outcomes

- Explain the role that languages play in software development and describe a spectrum of software languages that are in current use.
- Express properties of software languages using formal notations.
- Translate formal notations of software language properties into implementations of language processors.
- Demonstrate that a language processor is operating correctly by construction and use of appropriate test cases.

Assessment tasks

- Assignment 1: Syntax Analysis
- Examination 1 (Weeks 1-4)
- Assignment 2:Semantic Analysis
- Examination 2 (Weeks 5-8)
- Assignment 3: Translation
- Examination 3 (Weeks 9-12)
- Final Examination

Problem Solving and Research Capability

Our graduates should be capable of researching; of analysing, and interpreting and assessing data and information in various forms; of drawing connections across fields of knowledge; and they should be able to relate their knowledge to complex situations at work or in the world, in order to diagnose and solve problems. We want them to have the confidence to take the initiative in doing so, within an awareness of their own limitations.

This graduate capability is supported by:

Learning outcomes

- Express properties of software languages using formal notations.
- Translate formal notations of software language properties into implementations of language processors.
- Demonstrate that a language processor is operating correctly by construction and use of appropriate test cases.

Assessment tasks

- Assignment 1: Syntax Analysis
- Examination 1 (Weeks 1-4)
- Assignment 2:Semantic Analysis
- Examination 2 (Weeks 5-8)
- Assignment 3: Translation
- Examination 3 (Weeks 9-12)
- Final Examination

Effective Communication

We want to develop in our students the ability to communicate and convey their views in forms effective with different audiences. We want our graduates to take with them the capability to read, listen, question, gather and evaluate information resources in a variety of formats, assess, write clearly, speak effectively, and to use visual communication and communication technologies as appropriate.

This graduate capability is supported by:

Learning outcomes

- Explain the role that languages play in software development and describe a spectrum of software languages that are in current use.
- Express properties of software languages using formal notations.
- Translate formal notations of software language properties into implementations of language processors.
- Demonstrate that a language processor is operating correctly by construction and use of appropriate test cases.

Assessment tasks

- Assignment 1: Syntax Analysis
- Examination 1 (Weeks 1-4)
- Assignment 2:Semantic Analysis
- Examination 2 (Weeks 5-8)

- Assignment 3: Translation
- Examination 3 (Weeks 9-12)
- Final Examination

Changes from Previous Offering

No changes from 2015

Assessment Standards

COMP332 will be assessed and graded according to the University assessment and grading policies.

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

Pass: Can correctly reproduce facts and definitions across a breadth of concepts, but lacks depth of understanding. Can use notations to specify familiar language concepts in ways that are close to those discussed in lectures or notes. Can implement and test the basic features of a programming language similar to examples provided. Uses basic standards for code comprehension such as variable naming or documentation. Adheres to basic standards for presentation of written work. Can produce a basic description of the main aspects of a software system. Can describe basic test cases for software under study.

Credit/Distinction: As for Pass plus: Exhibits breadth and depth of understanding of concepts. Can use terminology accurately in new contexts. Can express ideas in their own words and has an understanding of the limits of their understanding. Can apply formal notations to describe language concepts that have not previously been seen. Can use provided general techniques to implement language concepts whose detailed implementation in code has not previously been discussed. Has well-developed skills for writing comprehensible, modular and well-documented code. Able to describe all or relevant aspects of a software system to an appropriate level of detail. Can articulate the principles behind the design of a suite of test cases.

High Distinction: As for Credit/Distinction plus: Is aware of the context in which the concepts are developed and their limitations. Able to generate and justify principles and hypotheses for existing or new concepts. Can recognise the limitations of formal notations for specifying some language concepts and is able to propose alternatives. Can develop new techniques to implement language concepts beyond those provided. Can critically evaluate aspects of the software system under study and the appropriateness of testing as a method for demonstrating software correctness.

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.

We will look at your overall performance on all assessments when determining your final grade. A total mark of at least 50% and a mark of at least 50% on the examinations taken together will be sufficient to pass the unit. Students who do not meet this cut-off will be examined on a case-by-case basis.