MONASH University
Information Technology

FIT3082
Programming languages and paradigms

Unit Guide

Semester 1, 2010

Last updated: 17 Feb 2010

Table of Contents

FIT3082 Programming languages and paradigms - Semester 1, 2010

Chief Examiner:

Associate Professor Bernd Meyer
Associate Professor
Phone: +61 3 990 52240
Fax: +61 3 990 55159

Lecturer(s) / Leader(s):

Clayton

Associate Professor Bernd Meyer
Associate Professor
Phone: +61 3 990 52240
Fax: +61 3 990 55159

Associate Professor Maria Garcia De La Banda
Head of School
Phone: +61 3 990 31058
Fax: +61 3 990 31077

Introduction

Welcome to FIT3082. At this point in your studies you should have achieved proficiency in at least one programming language, but you have probably also started to wonder whether there are fundamentally different ways of programming. The answer is that there are many, and for any computer scientist/software engineer knowledge of more than just a single programming paradigm is essential. This unit gives you an in-depth introduction to the two most important ones (functional programming and logic programming) and at least a glimpse of some other alternative models of computation that call for entirely different ways of programming.

Unit synopsis

This unit provides a thorough understanding of the four main programming language paradigms: imperative, functional, logic and object-oriented. In doing this, it reviews different example languages and the kinds of applications each language is intended for, focusing on how the choices made during the design of a language affect the resulting ease of programming. Particular emphasis will be made on functional and logic languages. Students will gain practical programming experience in languages that use these paradigms, such as SML and Prolog. Implications of language design on the implementation of programming languages will be highlighted.

Learning outcomes

At the completion of this unit students will have -
A knowledge and understanding of:

- the history of programming languages and the reasons for their introduction. Understanding of the fact that programming languages and programming language paradigms have evolved and will almost certainly change in the future;
- programming language paradigms including imperative, object-oriented, functional and logical. Comprehension that the choice of programming language affects the ease of programming. Ability to evaluate whether a particular paradigm or programming language is well-suited for a particular programming application;
- the importance of formal mathematical calculi as the basis for programming languages (in particular, Lambda Calculus and First Order Predicate Calculus). Understanding of how these calculi can provide the basis for programmign language semantics;
- the core aspects of modern functional programming languages Functions as first class objects, type systems, higher-order functions, eager and lazy evaluation;
- a particular functional programming language, such as ML or Scheme, which exemplifies most of these aspects and the ability to use it for implementing non-trivial algorithms;
- the core aspects of logic programming: logical inference, deduction, resolution, negation as failure, logic variables and unification;
- a particular logic language, such as Prolog or CLP, which exemplifies most of these aspects and the ability to use it for implementing non-trivial algorithms;
- the main issues in programming language design and comprehension of the various design alternatives. Examples of languages that mix influences from different programming paradigms, such as Python;
- how language design decision impact on the implementation of a programming language.

Developed attitudes that enable them to:

- value the important role that theory (in this case formal language theory and type theory) can play in the development of practical software applications (in this case compilers and other data

translation applications.);
- carefully consider the characteristics of the different programming languages available whenever developing a new application.

Developed the skills to:

- become an informed consumer of programming languages, choosing the right programming language for a particular task;
- be able to implement and debug non-trivial algorithms in both a functional and a logic language.

Contact hours

2 hrs lectures/wk, 1 hr laboratory/wk

Workload

two 1-hour lectures, weekly

one 1-hour tutorial (combined with practical work at the computer), weekly

9 hours of personal study (incl reading, lecture preparation and assignment preparation)

Unit relationships

Prerequisites

FIT2004 (or CSE2304) and FIT2014 (or CSE2303)

Prohibitions

CSE3322

Teaching and learning method

Teaching approach

Lectures will be used to present new concepts, introduce the fundamental approaches to problem solving with particular paradigms, compare different approaches, analyse their advantages and disadvantages, and propose general questions. The aim is to give students an overview of the concepts and to challenge them to think further. Tutorials and practicals will be used to link the theory with practice and deepen the students understanding and practical abilities. To this end, tutorials and practicals are tightly integrated into sessions where theoretical discussion and practical programming work complement each other.

Timetable information

For information on timetabling for on-campus classes please refer to MUTTS, http://mutts.monash.edu.au/MUTTS/

Tutorial allocation

On-campus students should register for tutorials/laboratories using the Allocate+ system: http://allocate.its.monash.edu.au/

Unit Schedule

Week	Date*	Topic	Key dates
1	01/03/10	Overview, History of Programming Languages	
2	08/03/10	ML I: Expressions, Functions, Basic Datatypes, Lists, Pattern Matching	
3	15/03/10	ML II: Type Definitions, Polymorphism	Hurdle Prac 1
4	22/03/10	ML III: Higher order functions, Structures	
5	29/03/10	ML IV: Input/Output, Destructive Update, Binding Mechanisms, Scope and Extent	Hurdle Prac 2
Mid semester break			
6	12/04/10	ML V: Type Inference, Overview of Abstraction Mechanisms	
7	19/04/10	LP I: Rule-based Programming, Deductive Databases, Basic LP Syntax, Logic Semantics	April 23, Assignment 1 due
8	26/04/10	LP II: Deduction, Search & Backtracking, Unification	
9	03/05/10	LP III: Basic Lists and Difference Lists, Cut, Negation by failure, Impure Features (I/O, Arithmetic)	Hurdle Prac 3
10	10/05/10	LP IV: Metaprogramming	
11	17/05/10	LP V: CLP basics	Hurdle Prac 4
12	24/05/10	LP VI: CLP	
13	31/05/10	Revision	May 28, Assignment 2 due

*Please note that these dates may only apply to Australian campuses of Monash University. Off-shore students need to check the dates with their unit leader.

Improvements to this unit

Some material has been removed from the syllabus (advanced material from meta programming and CLP) in favour of treating the remaining material in more depth.

Unit Resources

Prescribed text(s) and readings

Jeffrey D. Ullman, "Elements of ML Programming", Prentice Hall. Leon Sterling and Ehud Shairo: "The Art of Prolog", MIT Press.

Text books are available from the Monash University Book Shops. Availability from other suppliers cannot be assured. The Bookshop orders texts in specifically for this unit. You are advised to purchase your text book early.

Recommended text(s) and readings

Kenneth C. Louden "Programming Languages, Principles and Practice", Thomson.

David A. Watt "Programming Language Design Concepts", Wiley.

The following technical reports / journal articles may also be helpful and interesting:

(LP III) K. Knight: Unification: A Multidisciplinary Survey. ACM Computing Surveys 21(1)1989:93-124

(LP V) T. Fruwirth et al: Constraint Logic Programming - an informal introduction (ECRC Tech Report 93-5)

(LP V) P. van Hentenryck: Constraint logic programming, The Knowledge Engineering Review 6(3)1991:151-194

(Tutorial and Labs for LP) D.H.D. Warren: Logic Programming and Compiler Writing. Software - Practice and Experience 10, 1980:97-125

(Tutorials and Labs for LP) J. Cohen and T.J. Hickey: Parsing and Compiling Using Prolog. ACM Transactions on Programming Languages and Systems 9(2)1987:125-163

Required software and/or hardware

You will need access to

* SML/NJ (latest version)

* SICStus Prolog (latest version)

both are installed in the labs. For home use you can download SML/NJ from http://www.smlnj.org . Several free Prolog Systems are publicly available on the web (unfortunately, SICStus is not free).

Equipment and consumables required or provided

Students studying off-campus are required to have the minimum system configuration specified by the Faculty as a condition of accepting admission, and regular Internet access. On-campus students, and those studying at supported study locations may use the facilities available in the computing labs. Information about computer use for students is available from the ITS Student Resource Guide in the Monash University Handbook. You will need to allocate up to **n** hours per week for use of a computer, including time for newsgroups/discussion groups.

Study resources

Study resources we will provide for your study are:

- Weekly detailed lecture notes outlining the learning objectives, discussion of the content, required readings and exercises;
- Fortnighlty tutorial and laboratory tasks and exercises;
- Assignment specifications and sample solutions;
- A sample examination and suggested solution
- Access to past examination papers when available
- Discussion groups;
- This Unit Guide outlining the administrative information for the unit;
- The unit web site on MUSO (Moodle), where the resources outlined above will be made available.

Assessment

Overview

Examination (3 hours): 70%; In-semester assessment: 30%

Faculty assessment policy

To pass a unit which includes an examination as part of the assessment a student must obtain:

- 40% or more in the unit's examination, and
- 40% or more in the unit's total non-examination assessment, and
- an overall unit mark of 50% or more.

If a student does not achieve 40% or more in the unit examination or the unit non-examination total assessment, and the total mark for the unit is greater than 50% then a mark of no greater than 49-N will be recorded for the unit.

The unit is assessed with two assignments (assessed in the pracs) and a three hour closed book examination. To pass the unit you must:

- attempt at least one assignment and the examination
- achieve no less than 40% of the possible marks in the total non-examination assessment
- achieve no less than 50% of the possible marks in the examination
- achieve no less than 50% of possible marks overall

Note that there are also four "Hurdle Pracs" assessed as pass/fail. You will only be eligible to submit Assignment 1 if you pass at least one of the Hurdle Pracs 1, 2 and you will only be eligible to submit Assignment 2 if you pass at least one of the Hurdle Pracs 3, 4.

Assignment tasks

Assignment coversheets

Assignment coversheets are available via "Student Forms" on the Faculty website:
http://www.infotech.monash.edu.au/resources/student/forms/
You MUST submit a completed coversheet with all assignments, ensuring that the plagiarism declaration section is signed.

Assignment submission and return procedures, and assessment criteria will be specified with each assignment.

- **Assignment task 1**

  Title:
  Functional Programming
  Description:
  This will consist of programming tasks in ML including advanced functional concepts, such as higher order functions, list comprehensions, simulated laziness etc.
  Weighting:
  15%
  Due date:

April 23

- **Assignment task 2**

Title:
Logic Programming
Description:
This will consist of programming tasks in Prolog/CLP ranging from introductory concepts, such as lists, terms, and unification, to advanced concepts, such as meta-programming, DCGs, constraint solving, extra-logical features etc.
Weighting:
15%
Due date:
May 28

Examination

- **Weighting:** 70%
  Length: 3 hours
  Type (open/closed book): Closed book

See Appendix for End of semester special consideration / deferred exams process.

Due dates and extensions

Please make every effort to submit work by the due dates. It is your responsibility to structure your study program around assignment deadlines, family, work and other commitments. Factors such as normal work pressures, vacations, etc. are not regarded as appropriate reasons for granting extensions. Students are advised to NOT assume that granting of an extension is a matter of course.

Students requesting an extension for any assessment during semester (eg. Assignments, tests or presentations) are required to submit a Special Consideration application form (in-semester exam/assessment task), along with original copies of supporting documentation, directly to their lecturer within two working days before the assessment submission deadline. Lecturers will provide specific outcomes directly to students via email within 2 working days. The lecturer reserves the right to refuse late applications.

A copy of the email or other written communication of an extension must be attached to the assignment submission.

Refer to the Faculty Special consideration webpage or further details and to access application forms: http://www.infotech.monash.edu.au/resources/student/equity/special-consideration.html

Late assignment

Assignments received after the due date will be subject to a penalty of 10% per day of late submission. No submissions will be accepted later than 1 week after the due date.

Return dates

Students can expect assignments to be returned within two weeks of the submission date or after receipt, whichever is later.

Appendix

Please visit the following URL: http://www.infotech.monash.edu.au/units/appendix.html for further information about:

- Continuous improvement
- Unit evaluations
- Communication, participation and feedback
- Library access
- Monash University Studies Online (MUSO)
- Plagiarism, cheating and collusion
- Register of counselling about plagiarism
- Non-discriminatory language
- Students with disability
- End of semester special consideration / deferred exams