McMaster University
Department of Computing and Software
Dr. W. Kahl

COMP SCI 3EA3
Outline
2014-09-04

COMP SCI 3EA3 — Software Specification and Correctness

Instructor: Dr. Wolfram **Kahl**, Department of Computing and Software, ITB-245
E-Mail: kahl@cas.mcmaster.ca

Calendar Description:

Formal specifications in software development; logical formalisms; functional and relational specifications; completeness and consistency of specifications; verification; validation; presentation of information; tool supported verification.

Goals:

- Understanding of the motivation of mathematical approaches to software specification
- Ability to produce and evaluate formal software specifications
- Knowledge of one typical approach to formal software specification and verification
- Knowlege of different logical formalisms, of the principles of related tool support, and associated selection criteria
- Understanding of the essence of logical formalisms
- Translation skills between natural language and logical formalisms
- Knowledge of important proof systems for propositional logic and first-order predicate logic, and skills in producing formal proofs

Learning Objectives

Precondition

Students are expected to have achieved the following learning objectives before taking this course:

1. Students should know and understand
   a) Practical syntax of propositional logic
   b) Practical syntax of predicate logic, including variable binding issues
   c) Principles of typed expressions, and the types of the operators they are using
   d) Basic semantics of expressions and formulae (including truth tables, validity)
   e) Principles of calculational proofs
   f) Basic concepts and theorems about sets, functions and (especially binary) relations
   g) Basic theory of integers and counting
   h) Imperative programming
   i) Basic datastructures and algorithms
   j) Basic concepts of decidability, computability, and complexity
   k) Common software development process models
2. Students should be able to
   a) Extract the propositional-logic structure from English sentences, and translate propositional expressions back into English
   b) Translate English statements of moderate complexity into predicate-logic specifications
   c) Translate simple mathematical prose into predicate-logic definitions and formulae
   d) Write and "mentally execute" simple imperative programs
   e) Define and run test cases for their programs
   f) Use debugging tools for analysing the behaviour of their programs
   g) Implement data structure definitions for linked lists, queues, trees etc.
   h) Implement simple algorithms on structured data, such as linked lists, queues, trees etc.

Postcondition

Students are expected to achieve the following learning objectives at the end of this course:

1. Students should know and understand
   a) Syntax of propositional logic and predicate logic
   b) Mathematical definition of semantics of propositional logic and predicate logic
   c) Proof rules of propositional logic
   d) Proof rules of predicate logic (typically natural deduction)
   e) Big-step operational semantics of a simple imperative programming language
   f) Hoare logic proof rules for a simple imperative programming language
   g) Verification condition generation for a simple imperative programming language
   h) Scope and limitations of automated verification, proof, and program analysis tools.
2. Students should be able to
   a) Translate English specifications of program fragments into formal pre- and postcondition specifications
   b) Use predicate logic for procedure and datatype specification
   c) Annotate their programs with appropriate specifications and assertions for mechanised analysis with at least one verification tool.
   d) Prove properties of syntax and semantics of propositional logic and predicate logic by induction over the structure of terms and formulae
   e) Produce semantic structures satisfying, respectively refuting, predicate logic formulae.
   f) Produce formal derivations of theorems of propositional logic and predicate logic in different proof systems
   g) Produce formal derivations of theorems of predicate logic in important theories, including Peano arithmetic and the theory of arrays.
   h) Produce counterxample traces using operational semantics
   i) Use Hoare logic to prove partial and total correctness of simple imperative programs.
   j) Use VCGen algorithms to extract verification conditions from programs in a simple imperative programming language.

Course Page: `http://www.cas.mcmaster.ca/~kahl/CS3EA3/2014/`

While most of the internal electronic information exchange for this course will be handled via **Avenue**, the course pages will contain useful links to external material. The course pages also serve as the central fallback location for making information and material available outside Avenue, in particular in the case of Avenue accessibility problems.

It is the student's responsibility to be aware of the information on the course Avenue site and on the course web page, and to check regularly for announcements.

Schedule:

Lectures:	Monday, Wednesday, Thursday 16:30–17:20, JHE-A101
Tutorial:	Wednesday, 13:30–14:20, starting 10 September, usually JHE-A102. Occasionally, special sessions in computer-equipped labs may be announced on Avenue and course pages.
Office Hours:	preliminary: Monday, 12:00–13:00, and by appointment.

Students are expected to attend all lectures and tutorials.

Textbook

"RSD": *Rigorous Software Development — An Introduction to Program Verification*, by José Bacelar Almeida, Maria João Frade, Jorge Sousa Pinto, and Simão Melo de Sousa. Springer, London, 2011. DOI: 10.1007/978-0-85729-018-2 (available electronically via the McMaster library).

Additional material will be handed out or made available electronically via the course pages.

Outline:

(With relevant textbook chapters indicated — not all textbook contents will be covered in detail. Times are rough estimates.)

- Introduction (0.5 weeks, RSD 1–2)
- C Programming, and Static Analysis of C Programs in Frama-C (2 weeks, Frama-C documentation)
- Specifying C Programs using Predicate Logic Formulae in ACSL (3 weeks, Frama-C documentation)
- Propositional Logic: Syntax, Semantics, Proof System (≈1 week, RSD chapter 3)
- Predicate Logic: Syntax, Semantics, Proof System (1 week, parts of RSD chapter 4)
- Simple Imperative Programming: Syntax, Semantics, Hoare logic proof system (2 weeks, RSD chapter 5)
- Verification Condition Generation (1 week, RSD chapter 6)
- Safety Properties (1 week, RSD chapter 7)
- Procedure Calls (1 week, RSD chapter 8)

Using tool supported provided by Frama-C and associated tools will be expected throughout the course. Familiarity with the Linux (or MacOS X) command line will be necessary for using these tools.

Exercise Sheets and Tutorials:

In most weeks, an **Exercise Sheet** will be provided. These sheets will contain **Exercise Questions** and **Assignment Questions** that will usually be due about a week after they are provided.

Every week, starting September 10, there will be a one-hour tutorial session. The main purpose of the tutorials is to discuss **student work** on exercise problems. Therefore, every student is expected to complete the scheduled work, i.e., exercise problems or necessary reading, **before** the corresponding tutorial session — in particular, solutions and solution attempts to the Exercises and Assignment Questions on the weekly exercise sheets are to be brought to the tutorial.

Grading:

All examinations in this course will be **Closed Book**. That is, no written or printed material nor a calculator nor other electronic aids may be used during the examinations.

After notations, presentation rules, and basic definitions and proof rules have been introduced in class, **students are expected to know them at all times**.

Assignments: There will be graded **Assignment Questions**, and ungraded **Exercises**, essentially on a weekly basis.

Assignments may be graded only summarily; evaluation will be conducted mostly via the midterm tests and the final.

It is essential that you meet the deadlines for the Assigment Questions; there is no credit for documents handed in after the deadline.

If you cannot hand in your assignment on time **due to (e.g.) illness reasons**:

- **Hand your assignment in as soon as possible,** before the tutorial. If the assignment in question required handing in paper, hand your solution either to the instructor, or to the TA, or to Tina in the departmental office. (Last resort outside office hours: Insert into the drop box in front of the departmental office ITB-202.)

  We will make note of the time of your submission.
- **Follow the usual procedures for missed work** with your Associate Dean's office. The outcome of that process will decide whether/how the late submission can be counted.

Accomodations for missed work, including **late assignments**, require the corresponding form from the Associate Dean's office.

Where you use the electronic system, note that you are still **required** to get in touch with the instructor to actually be granted any accomodation.

Final Exam: The **final examination** will be scheduled by the Registrar's Office in the usual way. It will be a closed book examination of three hours duration and cover the material of **all** lectures, tutorials, handouts, and assignments.

Midterms: In addition, there will be **two midterm examinations**, details to be announced.

Prerequisite Knowledge Exam: During the lecture time on September 11, there will be a **Prerequisite Knowledge Exam**, mainly to help identify prerequisite topics that may need to be revisited during the course.

Grade Calculation: All exam grades will be percentage grades.

For every student, the course grade is calculated as a weighted average:

- For n being the number of **assignments** this course will have had, 25% of the weight are given to your $(n-1)$ best assignments.

  Some exercise sheets or assignments may contain **bonus questions**.

  All bonus marks will be added to the course grade *only for those who have passed the course otherwise.*

- The **prerequisite knowledge exam** counts 5%.

- Those **midterms** where your result is better than your result in the final count 20% each, and those midterms that are not better than the final count 10% each.

- The remaining weight (between 30% and 50%) is given to the **final exam**.

- As with every course at McMaster, every student will have the opportunity to evaluate the effectiveness of this course. The feedback that is received from the course evaluation is very valuable to the CS 3EA3 teaching staff, and so we are providing a **course evaluation bonus** to each student based on the level of class participation in the course evaluation according to the following table:

Class Participation	Bonus	Class Participation	Bonus
60-64%	0.25	80-84%	1.25
65-69%	0.50	85-89%	1.50
70-74%	0.75	90-94%	1.75
75-79%	1.00	95-100%	2.00

  Thus, for example, if 75% of the students enrolled in CS 3EA3 participate in the course evaluation of CS 3EA3, every student's final mark will receive a 1.00 percentage point bonus.

The final course grade will be converted from a percentage grade to a letter grade according to the scale of the Registrar's Office.

The instructor reserves the right to conduct any deferred exams orally.

Course Adaptation

The instructor and university reserve the right to modify elements of the course during the term.

The university may change the dates and deadlines for any or all courses in extreme circumstances. If either type of modification becomes necessary, reasonable notice and communication with the students will be given with explanation and the opportunity to comment on changes.

It is the responsibility of the student to check their McMaster email and course websites weekly during the term and to note any changes.

Academic Ethics

You are expected to exhibit honesty and use ethical behaviour in all aspects of the learning process. Academic credentials you earn are rooted in principles of honesty and academic integrity.

Academic dishonesty is to knowingly act or fail to act in a way that results or could result in unearned academic credit or advantage. This behaviour can result in serious consequences, e.g. the grade of zero on an assignment, loss of credit with a notation on the transcript (notation reads: "Grade of F assigned for academic dishonesty"), and/or suspension or expulsion from the university.

It is your responsibility to understand what constitutes academic dishonesty. For information on the various types of academic dishonesty please refer to the Academic Integrity Policy, located at `http://www.mcmaster.ca/academicintegrity`.

The following illustrates only four forms of academic dishonesty:

1. Plagiarism, e.g. **the submission of work that is not one's own** or for which other credit has been obtained.
2. **Collaboration where individual work is expected.**

   > **You have to produce your submissions for assignment questions yourself, and without collaboration** (except where and as far as group work is explicitly allowed or specified by the assignment statement).
   > For each assignment question there will normally be exercise questions similar to it — you **are allowed** to collaborate on these **exercise questions**. (The tutorials are typically not expected to cover all exercise questions.)

3. Improper collaboration in group work.
4. **Copying or using unauthorised aids in tests and examinations.**

Discrimination

The Faculty of Engineering is concerned with ensuring an environment that is free of all adverse discrimination. If there is a problem that cannot be resolved by discussion among the persons concerned, individuals are reminded that they should contact the Department Chair, the Sexual Harassment Office or the Human Rights Consultant, as soon as possible.