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

CAS 707
Outline
2016-01-06

CAS 707 — Formal Specification Techniques

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

Calendar Description:

Topics include: Pre-/postconditions, refinement, state-based approaches, event based approaches, algebraic specifications, Petri nets, temporal logic, properties of programs, specification verification and validation.

Course Objective

The main objective is to introduce students to commonly used formal state-based, event-based, and algebraic techniques to specify and verify software systems. The course aims as well to foster understanding of the mathematical foundations of these specification techniques, and the development of the skills needed to set up and manipulate mathematical models for relevant features of software systems.

Goals:

- Understanding of the motivation of mathematical approaches to software specification
- Knowledge of typical approaches to formal software specification and verification
- Ability to produce and evaluate formal software specifications
- Experience with a selection of current software specification formalisms and verification tools
- Knowlege of different logical formalisms, of the principles of related tool support, and associated selection criteria

Learning Objectives

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

1. Students should know and understand
   a) Syntax of propositional logic
   b) Syntax of predicate logic, including variable binding issues
   c) Semantics of propositional logic
   d) Semantics of predicate logic
   e) At least one proof system for predicate logic
   f) Basics of the metatheory of predicate logic (soundness, completeness)
   g) Principles of typed expressions, and the types of the operators they are using
   h) Principles of calculational proofs
   i) Basic concepts and theorems about sets, functions and (especially binary) relations
   j) Standard kinds of algebras
   k) Imperative programming
   l) Basic datastructures and algorithms
   m) Basics of functional programming
   n) Basic concepts of decidability, computability, and complexity
   o) Common software development process models
2. Students should be able to
   a) Translate English statements of moderate complexity into predicate-logic specifications
   b) Translate moderate complex mathematical prose into predicate-logic definitions and formulae
   c) Construct derivations in some proof system for predicate logic
   d) Use structural induction to prove meta-logical properties
   e) Write and "mentally execute" simple imperative programs
   f) Define and run test cases for their programs
   g) Use debugging tools for analysing the behaviour of their programs
   h) Implement data structure definitions for linked lists, queues, trees etc.
   i) Implement 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) Big-step operational semantics of a simple imperative programming language
   b) Hoare logic proof rules for a simple imperative programming language
   c) Verification condition generation for a simple imperative programming language
   d) Scope and limitations of automated verification and program analysis tools.
   e) Theory and applications of algebraic specification
   f) The spectrum of temporal logics
   g) Patterns of temporal-logics specifications, e.g., safety and lifeness conditions
   h) The principles behind model checking of temporal-logics specifications
2. Students should be able to
   a) Translate English specifications of program fragments into formal pre- and post-condition specifications
   b) Produce counterxample traces using operational semantics
   c) Annotate their programs with appropriate specifications and assertions for mechanised analysis with at least one verification tool
   d) Use Hoare logic to prove partial/total correctness of simple imperative programs
   e) Use VCGen algorithms to extract verification conditions from programs in a simple imperative programming language
   f) Understand structured algebraic specifications
   g) Produce algebraic specifications with appropriate structure
   h) Produce state transition models and temporal-logic specifications for (components of) concurrent systems, and verify them using model checking software

Course Page: `http://www.cas.mcmaster.ca/~kahl/CAS707/2016/`

This is where you will find further information, announcements, and useful links.
It is the student's responsibility to be aware of the information on the course web page and Avenue site, and to check regularly for announcements.

Literature

"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).

This will useful for Hoare logic and verification condition generation in the first part of the course; it is also a useful reference for first-order logic for CS with applications to formal specification.

"Using Z": *Using Z: Specification, Refinement, and Proof*, by Jim Woodcock and Jim Davies, Prentice Hall, 1996 (out of print; available on-line at `http://www.usingz.com/`).

Useful also as a **gentle** introduction to typed logic via natural deduction; also **covers basic discrete math** (sets, functions, relations...) extensively.

"Huth-Ryan": *Logic in Computer Science, Modelling and Reasoning about Systems*, by Michael R. A. Huth and Mark D. Ryan, Cambridge University Press, 2nd edition 2004. `http://www.cs.bham.ac.uk/research/lics/`

Conventional untyped (single-sorted) presentation of logic via natural deduction; also includes chapters on Hoare logic and on temporal logic and model checking.

"Alagar-Periyasami": *Specification of Software Systems*, by V. S. Alagar and K. Periyasamy, Springer 2011 (available electronically via the McMaster library).

Covers all course topics to some degree; lots of explanations, but frequently lacks precision.

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

Outline (Times are rough estimates; the first three items will likely be somewhat intermingled):

- Specifying and verifying C Programs (2 weeks, Frama-C docu.)
- Review of Predicate Logic as far as necessary (<1 week, parts of RSD chapt. 4)
- Hoare logic and VCGen in depth (2 weeks, RSD chapters 5–10)
- Algebraic Specification (2 weeks)
- Specification and verification of dynamic systems: Rewriting logic, temporal logic, model-checking (2–3 weeks)
- Separation logic, fully certified verification (2 weeks, VST & CompCert documentation)
- Selection of other topics, e.g.: State-based modelling (Z, B), Specification via types, JML (1–2 weeks)

In several topics, mastering **mechanised techniques (tool support)** will be part of the course requirements.

Grading:

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

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

Grade Calculation: All exam grades will be percentage grades.

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

- All **Assignments**: 25%
- One **Project**, with deliverables typically including a handout, source files, and presentation slides publicly submitted to Avenue, and a presentation held in class: 15%
- **Attendance and participation**: 5%
- **Midterm Exam** (an **individual 5–10 minute oral examination** for each student in the week Feb. 8–12 covering the first three items in the above Outline): 15%
- **Final Exam** (an **individual 20 minute oral examination** for each student): 40%

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

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.**
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.