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

CAS 763
Outline
2020-01-08

Certified Programming with Dependent Types
CAS 763

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

Quick Introduction

Dependent types are everywhere in mathematics — you probably don't even notice them. But can you implement a matrix multiplication function in C, C++, Java, Rust, Go, Standard ML, OCaml, or Haskell 2010 such that the *type system* will catch attempts to multiply a 3-by-4 matrix with a 2-by-5 matrix? Dependent types can encode such constraints, and much more: Via the Curry-Howard correspondence, specifications expressed as logical formulae can be turned into types, so that the type checker can be used to guarantee data-type invariants and program correctness: Programs that are not proven correct then won't even compile!

The course CAS 763 will help you to learn the dependently-typed programming language Agda, practice formalising the mathematics needed for expressing datatype invariants and program specifications, and also acquire knowledge and understanding of the relevant foundations.

Previous experience with Haskell wold be useful, but is not strictly necessary. Familiarity with basic propositional and predicate logic is essentially assumed.

Calendar Description

Type systems featuring types depending on values empower not only logics that can capture common mathematical formalisations more naturally than conventional first-order or higher-order logics; they also empower programming languages where specifications may be incorporated into the type of programs, and well-typed programs are thus guaranteed to satisfy these specifications.

Students will learn at least one dependently-typed programming language in depth. The course will also cover associated foundations, including relevant type theories and the Curry-Howard correspondence, as well as useful patterns of formalising, programming, and proving in dependently-typed programming languages.

Topics

The main language for this course will be Agda2 (usually simply referred to as "Agda").

- Basic data type definitions; functional programming with pattern matching
- The Curry-Howard correspondence: Formulae as types, proofs as data
- Proving properties of functional programs
- Indexed datatypes
- Simple certified programs: Incorporating specifications into types
- Invariant-carrying datatypes
- Representing typed data: Expression languages, logics and programming languages
- Dependently-typed programming patterns: Universes, Views
- Category-theoretic abstractions as typed interfaces
- Other dependently-typed languages, e.g., Coq and Idris.

Learning Objectives

Precondition

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

1. Students should know and understand
   a) Propositional logic and predicate logic
   b) Principles of typed expressions and statically typed programming languages
   c) Basic concepts and theorems about sets, functions and relations
   d) Standard kinds of algebras
   e) Basic datastructures and algorithms
   f) Basics of functional programming
2. Students should be able to
   a) Translate moderate complex mathematical prose into predicate-logic definitions and formulae
   b) Construct derivations in some proof system for predicate logic
   c) Routinely use structural induction for proofs where appropriate
   d) Quickly acquire a new programming paradigm

Postcondition

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

1. Students should know and understand
    a) The difference in expressiveness dependent types bring to logic
    b) The difference in expressiveness dependent types bring to programming languages
    c) The Curry-Howard correspondence
    d) The syntax and tpying principles of the Agda2 programming language
    e) The reduction semantics of the Agda2 programming language
    f) Commonly used programming patterns in dependently-typed languages, including universes, views
    g) An overview of dependently-typed languages currently available
2. Students should be able to
    a) Design and implement dependently-typed data structures
    b) Express specifications as types in a dependently-typed language
    c) Implement functions that "prove their own correctness"
    d) Design module interfaces that guarantee correct implementations
    e) Fluently use the Agda2 programming language to formalise specifications and produce certified implementations.

Course Page:

`http://www.cas.mcmaster.ca/~kahl/DepTyp/2020/`

This and the Avenue course page are 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.

Grading:

Assignments: In the first part of the course, there will be weekly **Assignments** to support the learning of the Agda programming language. Solutions will need to be submitted electronically.

Paper and Project Presentations: In the later part of the course, you will be assigned topics for presentation that will start with seminar-style paper presentations, and may also involve exploring other languages and systems, or completing a larger, individual programming assignment.

Evaluation will be based both on the project work (to be submitted electronically) including documentation quality, and on the presentation; details TBA.

Grade Calculation: All exam grades will be percentage grades.

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

- All **Assignments**: 35%
- Three **Paper and Project Presentations**: 15% + 20% + 30%

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 Integrity

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.