

# CS\_336/CS\_M36 (Part 2)/CS\_M46 Interactive Theorem Proving

<http://www-compsci.swan.ac.uk/~csetzer/lectures/intertheo/05/index.html>

**Dr. Anton Setzer**

<http://www-compsci.swan.ac.uk/~csetzer/index.html>

**Lent Term 2006**

36/CS.M36 (part 2)/CS.M46 Interactive Theorem Proving; Lentterm 2006, Sect. 0

0-1

## 0. Introduction

- (a) [What is interactive theorem proving?](#)
- (b) [Administrative Issues.](#)
- (c) [Plan.](#)
- (d) [Literature](#)

36/CS.M36 (part 2)/CS.M46 Interactive Theorem Proving; Lentterm 2006, Sect. 0

0-2

## (a) What is Interact. Theorem Proving

### Need for Theorem Proving

- We need to prove theorems in order to establish mathematical theorems.
  - E.g. that certain problems are decidable, undecidable, polynomial computable etc.
- We need them as well in order to establish the correctness of software and hardware.
  - Is floating point division for the Intel processor correct?
  - Is a railway control system safe?
    - When verifying the Swedish railway system, lots of bugs were found.

CS.336/CS.M36 (part 2)/CS.M46 Interactive Theorem Proving; Lentterm 2006, Sect. 0

## 4 Ways of Proving Theorems

### 1. Theorem proving by hand.

- What mathematicians do all the time.
- Will remain in the near future the main way for proving theorems.
- Problem: Errors.
  - As in programs after a certain amount of lines there is a bug, after a certain amount of lines a proof has a bug.
  - The problem can only be reduced by careful proof checking, but not eliminated completely.
- Unsuitable for verifying large software and hardware systems.
  - Data usually too large.
  - Likely that one makes the same mistakes as in the software.

CS.336/CS.M36 (part 2)/CS.M46 Interactive Theorem Proving; Lentterm 2006, Sect. 0

# 4 Ways of Proving Theorems

## 2. Theorem proving with some machine support.

- Machine checks the syntax of the statements, creates a good layout, translates it into different languages.
- Theorem proving still to be done by hand.
- **Example:** most systems for specification of software (e.g. CSP-CASL, as used by Dr. Roggenbach).
- **Advantages:**
  - Less errors.
  - User is forced to obey a certain syntax.
  - Specifications can be exchanged more easily.
- **Disadvantage:** Similar to 1.

# 4 Ways of Proving Theorems

## ● (Interactive theorem proving)

- **Disadvantages:**
  - It takes much longer than proving by hand.
    - Similar to programming:  
To say in words what a program should do, doesn't take long.  
To write the actual program, can take a long time, since much more details are involved than expected.
  - Requires experts in theorem proving.

# 4 Ways of Proving Theorems

## 3. Interactive Theorem Proving.

- Proofs are fully checked by the system.
- Proof steps have to be carried out by the user.
- **Advantages:**
  - Correctness guaranteed (provided the theorem prover is correct).
  - Everything which can be proved by hand, should be possible to be proved in such systems.

# 4 Ways of Proving Theorems

## 4. Automated Theorem Proving.

- The theorem is shown by the machine.
- It is the task of the user to
  - state the theorem,
  - bring it into a form so that it can be solved,
  - usually adapt certain parameters so that the theorem proving solves the problem within reasonable amount of time.
- Espec. Dr. Kullmann is an expert in this area.

# 4 Ways of Proving Theorems

## • (Automated theorem proving)

### • Advantages

- Less complicated to “feed the theorem into the machine” rather than actually proving it. Might be done by non-specialists.
- Sometimes faster than interactive theorem proving.

# This Lecture

## • In this lecture we will consider approach (c).

## • We will make use of the theorem prover **Agda**, based on **Martin-Löf type theory**.

- The theory was developed by **Per Martin-Löf**.
  - Per Martin-Löf is professor at Stockholm University for philosophy and mathematical logic.
  - The lecturer is and was collaborating with him, especially while working as a research associate at Uppsala University (Sweden).

# 4 Ways of Proving Theorems

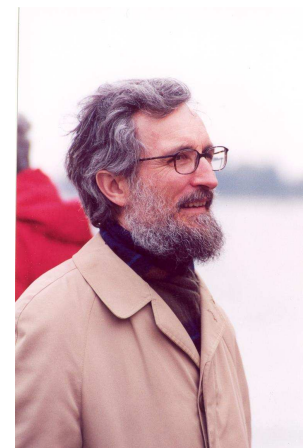
## • (Automated theorem proving)

### • Disadvantages

- Many problems cannot be proved automatically.
- Can often deal only with finite problems.
  - We can show the correctness of one particular processor.
  - But we cannot show a theorem, stating the correctness of a parametric unit (like a generic  $n$ -bit adder for arbitrary  $n$ ).
  - In some cases this can be overcome.
- Limits on what can be done (some hardware problems can be verified as 32 bit versions, but not as 64 bit versions).

# Per Martin-Löf

The Father of Martin-Löf Type Theory, the variant of dependent type theory used in this module.



# This Lecture

- The students will learn (in practical lab sessions) how to actually carry out proofs in Agda.
  - **Proving theorems** will not be much different from **programming**.
  - In Martin-Löf type theory “proving” and “programming” is the same.
- With a slight shift of emphasis, this module could as well be called “**programming with dependent types**”.

## (b) Administrative Issues

### Address:

Dr. A. Setzer  
Dept. of Computer Science  
University of Wales Swansea  
Singleton Park  
SA2 8PP  
UK

**Room** Room 211, Faraday Building

**Tel.** (01792) 513368

**Fax.** (01792) 295651

**Email** a.g.setzer@swansea.ac.uk

# Assessment

- 80% Exam.
- 20% coursework (for **all 3 modules** involved):
  - 4 **small** assignments. Each counts 5% (Plan, might be changed).
    - Handed out approx. every 2nd week.
    - Due two weeks later.

## Course home page

- Located at  
<http://www.cs.swan.ac.uk/~csetzer/lectures/intertheo/05/index.html>
- There is an open version,
- and a password protected version.
- The password is \_\_\_\_\_.
- Errors in the notes will be corrected on the slides and noted on the list of errata.
- In order to reduce plagiarism, coursework and solutions to coursework will **not** be made available in electronic form (e.g. on this web site).

# Timetable, Course Material

---

- Two lectures per week.
  - Tuesday, 12:00, Faraday-B
  - Thursday, 13:00, Faraday-J
- Web page contains overhead slides from the lectures. Course material will be continually updated.

## (c) Plan

---

0. [Introduction](#).
1. From simple to dependent types.
2. Reduction systems and term rewriting.
3. The  $\lambda$ -calculus and implication.
4. The  $\lambda$ -calculus with products and conjunction.
5. The logical framework.
6. Data types.

# (d) Literature

---

- In general, the module is self-contained.

## Main Course Literature

---

- B. Nordström, K. Peterson, J. M. Smith: *Programming in Martin-Löf's type theory*. Available via <http://www.cs.chalmers.se/Cs/Research/Logic/book/>. **Course book**, although a little bit too high level.
- B. Nordström, K. Peterson, J. M. Smith: *Martin-Löf's type theory*. Handbook of Computer Science, Vol 5, 1-37. Oxford Univ. Press, 2000. Available via <ftp://ftp.cs.chalmers.se/pub/cs-reports/papers/smith/hlcs.ps.gz>. Similar as the previous book, but shorter.

## Other Introductory Books

---

- Martin-Löf, Per: *Intuitionistic Type theory*. Bibliopolis, Naples, 1984.  
Relatively easy short book, from the father of the type theory we are using. Intended for philosophers.
- Aarne Ranta: *Type-theoretic grammar*. Clarendon Press, 1995.  
Use of type theory in linguistics and for translation between languages. Has a good and simple introduction into type theory.

## Books on Term Rewriting

---

- Terese: *Term Rewriting Systems*. Cambridge Tracts in Theoretical Computer Science 55, 2003.  
Very thick and detailed study of term rewriting written by some of the most important people in term rewriting.
- Baader, Franz; Nipkow, Tobias: *Term rewriting and all that*. Cambridge University Press, 1999.  
Shorter than the book by Terese.

## More advanced Books

---

- Troelstra, A. S. and van Dalen, D.: *Constructivism in mathematics. Vol. I*. North Holland, 1988.  
Contains some material of interest (e.g. BHK interpretation of logical connective). Postgraduate level.
- Chapter 11 of Troelstra, A. S. and van Dalen, D.: *Constructivism in mathematics. Vol. II*. North Holland, 1988.  
Book on postgraduate level. Deviates from “official Martin Löf type theory”.

## Books on the $\lambda$ -Calculus

---

- J. R. Hindley, B. Lercher and J.P. Seldin: *Introduction to combinatory logic*. Cambridge University Press, 1972.  
Mainly chapter 1 relevant for this module.
- J. Roger Hindley and Jonathan P. Seldin: *Introduction to Combinators and  $\lambda$ -Calculus*. Cambridge University Press, 1986.  
Mainly chapter 1 relevant for this module.
- J. Roger Hindley: *Basic simple type theory*. Cambridge University Press, 1997.  
Introduces a slightly more powerful type theory than used in this module.

# Books on the $\lambda$ -Calculus

---

- H.P. Barendregt:  
*The Lambda Calculus. It's syntax and semantics.*  
Revised Edition. Elsevier, 1984.  
Thorough monograph, the bible of the  $\lambda$ -calculus. Level too high for this module.