Computer Supported Modeling and Reasoning

David Basin, Achim D. Brucker, Jan-Georg Smaus, and Burkhart Wolff

April 2005

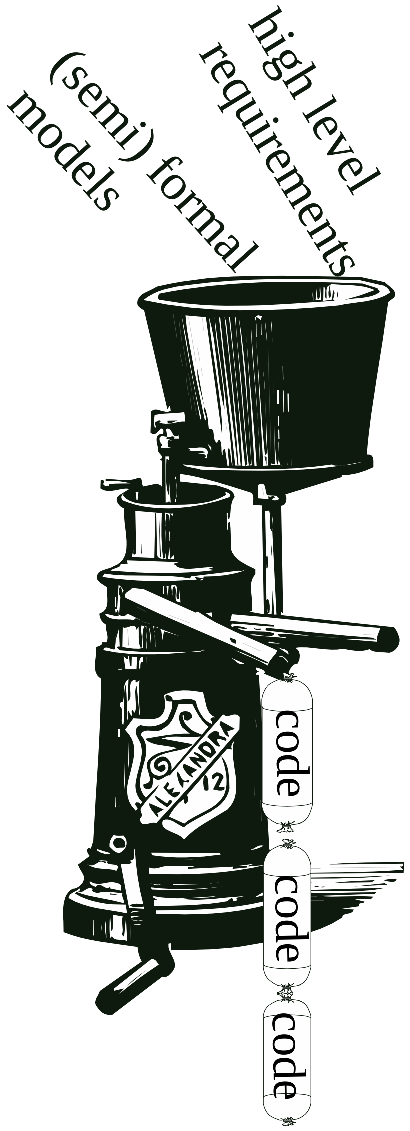
http://www.infsec.ethz.ch/education/permanent/csmr/

Introduction

David Basin and Burkhart Wolff

What this Course is about

- Modeling in Mathematics and Computer Science
  - logics and proofs:
    input: logics, deductive systems
    output: (meta)-theorems
  - program programs and verifications:
    input: specifications, programs, properties
    output: correctness guarantees
- Technically: mechanization and application
- Generally: making logic come to life and useful as a general tool.



Why this Course Matters

Academic motivation: deepen knowledge of logic and formal reasoning. Gain insights into philosophical foundations such as Reductionism and Hilbert's Program

Practical motivation: verification and formal methods

- The last decade has seen spectacular hardware and software failures and the birth of a new discipline: the verification engineer
- Exciting positions at companies like Intel, Gemplus, . . .

Why this Course Matters (2)

In general:

- Understanding formal reasoning improves understanding of how to build correct systems
- Mechanization provides formal guarantees

Relationship to Other Courses

Logic: deduction, foundations, and applications

Software engineering: specification, refinement, verification

Hardware: formalizing and reasoning about circuit models

Artificial Intelligence: knowledge representation, reasoning, deduction

In general, you will develop a deeper understanding of mathematical and logical reasoning, which is central to computer science.

Overview: Five Parts

1. Logics (propositional, first-order, higher-order)
2. The "Metalogical Approach":
   Representing logics in logics
3. Theorem Proving with Isabelle:
   proof strategies, machine supported theory development
4. The "Semantic Approach":
   Representing syntax and semantics of logics in logics
5. Applications:
   Case studies in formalizing theories of computer science.

Our presentation roughly follows this conceptual structure.

Overview: Part 1: Logics

1.1 This Introduction

1.2 Propositional Logic (PL)

1.3 Natural Deduction in PL

1.4 First-Order Logic (FOL)

1.5 Natural Deduction in FOL

1.6 Theories in FOL

1.7 Naive Set-Theory in FOL

Overview: Part 2: Metalogical Approach

2.1 Foundation: λ-calculi

2.2 Encoding Syntax in LF

2.3 Encoding Deduction in LF

Overview: Part 3: Theorem Proving with Isabelle

3.1 Basic Deduction: Resolution

3.2 Automated Deduction: Classical Reasoner

3.3 Automated Deduction: Term Rewriting

3.4 The Isabelle Metalogic

3.5 Proof Pragmatics

Overview: Part 4: Semantic Approach

4.1 Foundations of Higher-order Logic (HOL)

4.2 Derived Rules of HOL

4.3 Conservative Extensions in HOL

4.4 Basic Library of HOL

4.5 Fixpoints and Inductive Sets

4.6 Wellfounded Orderings and Recursion

4.7 Arithmetic in HOL

4.8 Datatypes in HOL

Overview: Part 5: Applications

5.1 Encoding Imperative Languages

5.2 Encoding Functional Languages

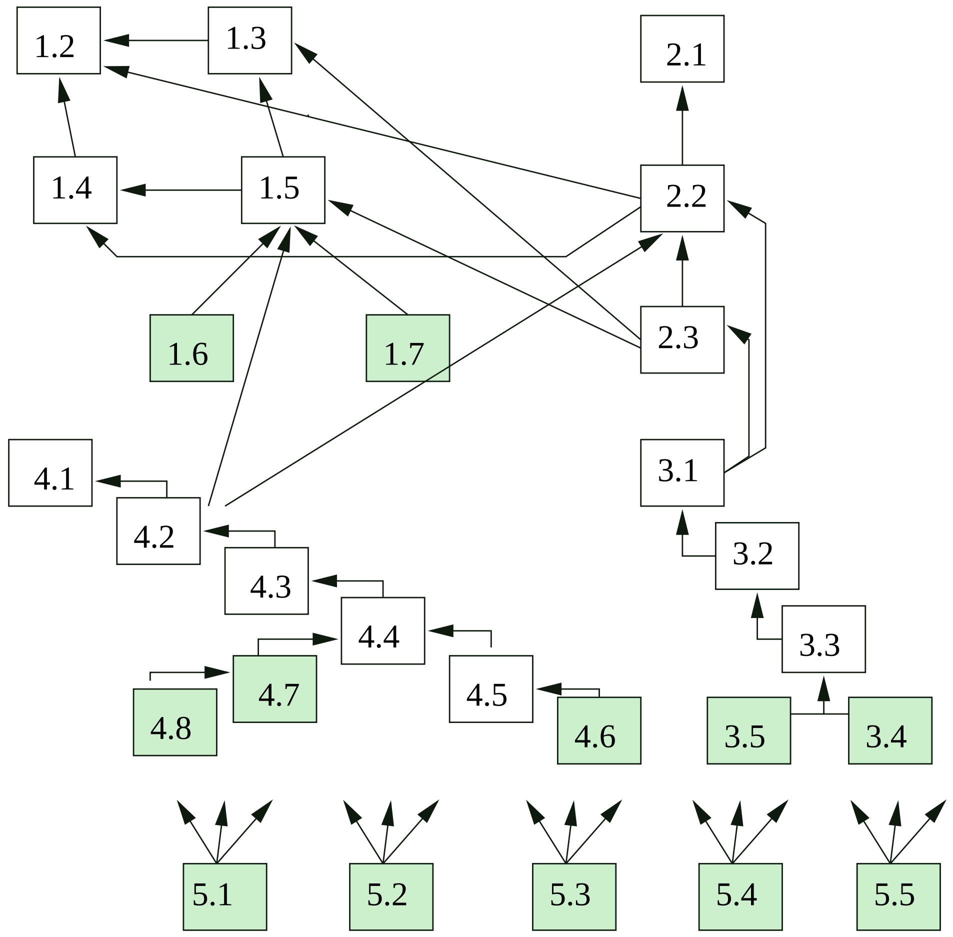
5.3 Refinement

5.4 HOL-OCL

5.5 Other Applications

Overview: Part 6

- Dependencies of the chapters:
- (Green boxes may be skipped at first reading)



Requirements

Some knowledge of logic is a useful prerequisite for this course.

We will try to accommodate different backgrounds, e.g. with pointers to additional material. Your feedback is essential!

Requirements (2)

You must be willing to participate in the labs and get your hands dirty using a proof development system:

- further develop course material
- present orthogonal material on pragmatics of mechanized reasoning
- hands-on experience.

Experience shows students end up in two equivalence classes.

It is hard in the beginning but the rewards are large.

More Detailed Explanations

What is Verification?

Verification is the process of formally proving that a program has the desired properties. To this end, it is necessary to define a language in which the desired properties can be formulated, i.e. specified. One must define a semantics for this language as well as for the program. These semantics must be linked in such a way that it is meaningful to say: “Program X makes formula Φ true”.

What is Reductionism?

The philosophical position of reductionism, carried to the extreme, says that anything in the world can be reduced to physics and mathematical modeling, which in itself can be reduced to a small number of axioms, and which can be finally reduced to one formula.

While we do not claim such a strong position ourselves, part of the material we present here (in particular HOL and the development of its library) roughly follow the Russels and Whiteheads Principia Mathematica which was deeply influenced by reductionistic ideas.

What is Hilbert’s Program?

In the 1920’s, David Hilbert attempted a single rigorous formalization of all of mathematics, named Hilbert’s program. He was concerned with the following three questions:

1. Is mathematics complete in the sense that every statement could be proved or disproved?
2. Is mathematics consistent in the sense that no statement could be proved both true and false?
3. Is mathematics decidable in the sense that there existed a definite method to determine the truth or falsity of any mathematical statement?

Hilbert believed that the answer to all three questions was ‘yes’.

Thanks to the the incompleteness theorem of Gödel (1931) and the undecidability of first-order logic shown by Church and Turing (1936–37) we know now that his dream will never be realized completely. This makes it a never-ending task to find partial answers to Hilbert's questions.

For more details:

- Panel talk by Moshe Vardi
- Lecture by Michael J. O'Donnell
- Article by Stephen G. Simpson
- Original works Über das Unendliche and Die Grundlagen der Mathematik [vH67]
- Some quotations shedding light on Gödel's incompleteness theorem
- Eric Weisstein's world of mathematics explaining Gödel's incompleteness theorem. Gödel's incompleteness theorem

Mechanizing Logic

Meanwhile, a number of theorem proving systems mechanizing logical deduction is available (c.f. The Sixteen Provers of the World; compiled by Freek Wiedijk). The number of Formal Methods systems (i.e. (automated) theorem provers geared towards system and program verification) is even larger.

We will learn to make logic run on a computer by using the Isabelle system.

Isabelle has been used for very substantial proofs in pure mathematics (such as the Prime Number Theorem) or computer science (such as the Java Virtual Machine).

What is (a) Logic?

The word logic is used in a wider and a narrower sense.

In a wider sense, logic is the science of reasoning. In fact, it is the science that reasons about reasoning itself.

In a narrower sense, a logic is just a precisely defined language allowing to write down statements (i.e. some of the syntactic entities of this language), together with a predefined inference or deduction mechanism allowing for deducing new statements from established ones. The deduction mechanism is usually represented by logical rules. In this course, we consider Propositional logic, first-order logic, and higher-order logic are three different logics, but also applied logics (called formal specification languages) such as Z or Hoare Logic.

What is a Metalogic?

A metalogic is a logic that is used to formalize syntax, deduction system, semantics and possibly meta-properties (such as correctness, completeness, adequacy, etc) of another logic — called the object logic.

What is a Semantics?

A semantics of a formal language (i.e. a logic, a programming or specification language) is a function that assigns to each element of its syntax a denotation or value.

Given a semantics, the question can be settled when a rule — allowing derivations or deductions or inferences of syntactic elements such as logical statements from other syntactic elements — is correct, namely that it never changes the value of a derived statement.

With the semantic approach we refer to the method consiting in defining an (explicit or implicit) semantic function, usually in terms of a set theory or constructs of similar expressive power, and to derive the logical rules of the object logic from this definition.

What is a Formal Language?

A formal language must have a syntax, i.e. a formally defined set of sequences of elementary symbols and a semantics that give it a formally defined meaning.

The term “formal language” ma be referred to logics, programming or specification languages.

What is a Theory?

A theory is a collection of logical statements in a logic and the set of all logical statements that can be derived from them via the inference of the logic.

Quite often, we will implicitly identify the former (usually finite) set with the (usually infinite) latter one.

A theory is used to model a tiny portion of the “world”, let it be gravitation and quantums in physics, or prime numbers in mathematics, or the Java Virtual Machine in computer science.

More information later.

What we Neglect

We will introduce different logics and formal systems (so-called calculi) used to deduce formulas in a logic. We will neglect other aspects that are usually treated in classes or textbooks on logic, e.g.:

- Gödels completeness and . . .
- . . . incompleteness theorems.

As an introduction we recommend [vD80] or [And86].

Equivalence Classes

In this course it makes no sense to follow just a little bit. Our experience is that the committed students learn the material, whereas the others go away empty-handed.

References

[And86] Peter B. Andrews. *An Introduction to Mathematical Logic and Type Theory: To Truth Through Proofs*. Academic Press, 1986.

[vD80] Dirk van Dalen. *Logic and Structure*. Springer-Verlag, 1980. An introductory textbook on logic.

[vH67] Jean van Heijenoort, editor. *From Frege to Gödel: A Source Book in Mathematical Logic, 1879-193*. Harvard University Press, 1967. Contains translations of original works by David Hilbert.