

DTU



Proof Assistants and Related Tools

Frederik Krogsdal Jacobsen Jørgen Villadsen

Technical University of Denmark

Introduction

Technical University of Denmark – undergraduate software *engineering* students

Undergraduate logic course: Logical Systems and Logic Programming (80+ students)

Graduate logic course: Automated Reasoning (40+ students)

Undergraduate course leads into many other courses and potentially a thesis project

50% lectures and 50% supervised exercise sessions, plus homework assignments

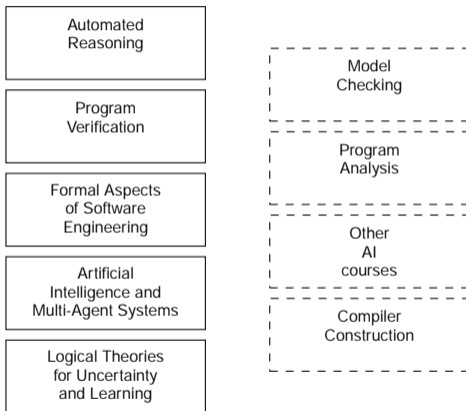
Danish custom: almost all students pursue a MSc degree

The surrounding curriculum

Year				
1	2	3	4	...
BSc			MSc	
Discrete Mathematics (mandatory)	Functional Programming (mandatory)	Logical Systems and Logic Programming	Automated Reasoning	
Introductory Programming (mandatory)	Computer Science Modelling (mandatory)		Program Verification	
Algorithms and Data Structures 1 (mandatory)	Algorithms and Data Structures 2		Formal Aspects of Software Engineering	
	Introduction to Artificial Intelligence		Artificial Intelligence and Multi-Agent Systems	
	Introduction to Machine Learning and Data Mining		Logical Theories for Uncertainty and Learning	

Axioms and inference rules

Why pick one over the other?
Notation is widely used
Special case: type systems



Extensions to a logic

In practice, first-order logic is not enough

Examples:

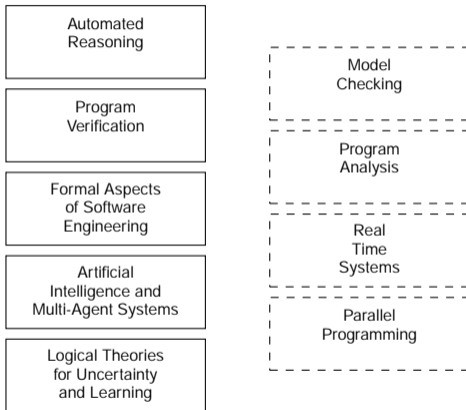
Modal Logic

Separation Logic

Epistemic Logic

Hoare Logic

Higher-order logic



Soundness and completeness

Basic understanding is enough
Sound approximations
Proof systems

Automated
Reasoning

Program
Verification

Formal Aspects
of Software
Engineering

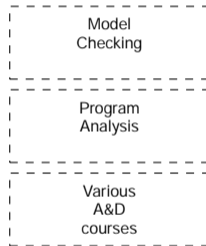
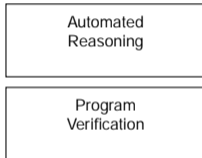
Artificial
Intelligence and
Multi-Agent Systems

Model
Checking

Program
Analysis

Decidability

Basic understanding is enough
Techniques for proving termination
Only practical examples are useful!

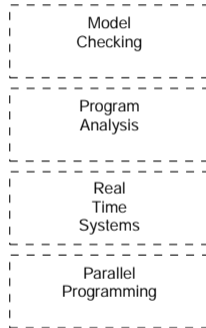
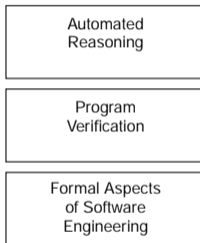


Writing formal specifications

Verification tools need formal specifications

Connection to the real world

Approximations and models

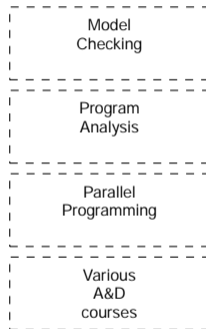
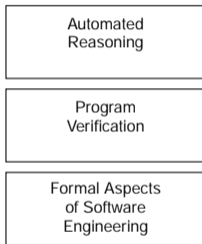


Writing formal proofs

Needed for all proofs when they become difficult

Many techniques to explore and practice

Focus should be on strategies, not particular proofs!

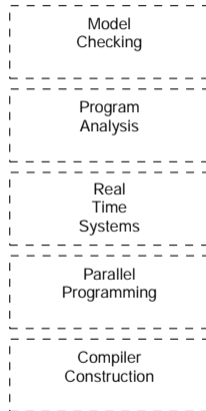
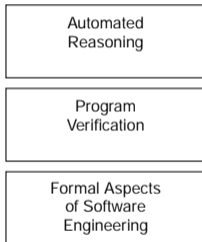


Proof algorithms

How do automated theorem provers work?

The foundation of formal analysis tools

Special case: type checkers



Representing knowledge

Foundation of artificial intelligence

Logic programming

Databases

Artificial
Intelligence and
Multi-Agent Systems

Logical Theories
for Uncertainty
and Learning

Other
AI
courses

Implementations

Theory is not enough

Complexity of binders, substitution, etc.
is often swept under the rug

Often much harder than expected – see
e.g. the POPLmark challenge

Automated
Reasoning

Program
Verification

Artificial
Intelligence and
Multi-Agent Systems

Model
Checking

Program
Analysis

Compiler
Construction

Topics I did not mention

“Advanced” set theory

Belongs in a math course

Recursion theory

Non-computable functions are not useful for computer science undergraduates

Complexity theory belongs in courses on algorithms and data structures

Non-finite model theory

Quantifier elimination and decidability belongs in a course on automated reasoning

Non-structural proof theory

Proof-theoretic semantics are interesting, but belongs in a specialized course

Summary of topics

- Axioms and inference rules
- Extensions to a logic
- Soundness and completeness
- Decidability
- Writing formal specifications
- Writing formal proofs
- Proof algorithms
- Representing knowledge
- Implementations

Summary of topics

Axioms and inference rules

Extensions to a logic

Soundness and completeness

Decidability

Writing formal specifications

Writing formal proofs

Proof algorithms

Representing knowledge

Implementations

Difficult to practice with pen and paper



Proof assistants

What are they good for?

Proof assistants

What are they good for?

All the things!

- Axioms and inference rules

- Extensions to a logic

- Soundness and completeness

- Decidability

- Writing formal specifications

- Writing formal proofs

- Proof algorithms

- Representing knowledge

- Implementations

Proof assistants

What are they good for?

All the things!

- Axioms and inference rules

- Extensions to a logic

- Soundness and completeness

- Decidability

- Writing formal specifications

- Writing formal proofs

- Proof algorithms

- Representing knowledge

- Implementations

And more:

- Confidence when experimenting

- Instant feedback on definitions and proofs

- Implemented examples of proof algorithms and formal proofs