# Mathematical and Logical Foundations of Computer Science

## Theorem proving

Vincent Rahli

(some slides were adapted from Benedikt Ahrens' slides)

University of Birmingham

# Where are we?

- Symbolic logic
- Propositional logic
- Predicate logic
- **Intuitionistic vs. Classical logic**
- **Type theory**

# Today

Theorem proving using Lean

- Propositional Logic in Lean
- Proofs as programs in Lean

**Useful links**:

- online version: https://leanprover.github.io/live/master/
- reference manual: https://leanprover.github.io/reference/index.html
- theorem proving in Lean: https://leanprover.github.io/theorem_proving_in_lean/index.html

**Further reading**:

- Chapter 4 of http://leanprover.github.io/logic_and_proof/

# Theorem Proving - What and Why?

**Theorem provers** (or **proof assistants**) are formal verification tools that help establish claims

Of both

- **mathematical results**
- **hardware & software verification**

**Why use formal verification tools?**

- computer checked proofs are **less error prone** than pen-and-paper proofs
  - those tools can check that we do not make mistakes
  - they can handle low level details "fairly" automatically
  - they can easily check and re-check proofs
- To **rely on safety-critical systems**, they need to be checked/verified to the highest standards possible
  - Such as power plants, airplanes, autonomous vehicles, etc.
  - failures can be catastrophic (loss of lives, money, data, etc.)!

# Theorem Proving - What and Why?

**Example of a mistake in Mathematical proofs:** Voevodsky (Fields medalist) discovered a mistake in one of his paper. *"A technical argument by a trusted author, which is hard to check and looks similar to arguments known to be correct, is hardly ever checked in detail"* - Vladimir Voevodsky, 2014, `https://www.ias.edu/ideas/2014/voevodsky-origins`

**Many examples of costly hardware & software failures:**

- 1994: Intel Pentium bug in the floating point unit
- 1996: ESA (European Space Agency) Ariane 5 launcher self-destructs
- 2010/2015: Toyota Prius software "glitches"
- 2012/2014: Heartbleed security bug in the OpenSSL crypto library
- 2018: TSB online banking "glitches"

**Could we avoid these issues using formal verification?**

# Theorem Proving - Automated vs. Interactive

Several kinds of formal verification tools:

- **automated**
  - e.g., SAT/SMT solvers such as Z3
  - e.g., Model checkers such as Spin, Prism
  - automatically **search** solutions to problems
- **interactive**
  - e.g., Agda, Coq, Lean, etc.
  - provide ways of **building** and **verifying** proofs
  - every single proof step is checked to be correct w.r.t. some logical foundations
- many systems, such as Lean, support both automated and interactive theorem proving

# Theorem Proving - Formal Verification

Formal verification tools such as theorem provers are especially useful in computer science to

- prove **mathematical** results, e.g.,
  - four-color theorem
  - odd-order theorem
- prove the correctness of **hardware & software systems**, e.g.,
  - CompCert: a C compiler verified in Coq
  - CakeML: a verified ML language in HOL4
  - sel4: a microkernel in Isabelle

Many companies nowadays use formal verification tools to ensure the correctness of their hardware & software systems, such as (see `https://github.com/ligurio/practical-fm`):

| | | |
|---|---|---|
| Airbus | Amazon | ARM |
| BAE Systems | Data61 | Ethereum |
| Facebook | Galois | Google |
| Microsoft | NASA | ... |

# Theorem Proving - Formal Verification

In hardware & software system verification, formal verification tools are used to **mathematically prove** that **systems** (programs or models) satisfy formal **specifications**

**Example:** For a sorting algorithm, we want to prove that the result is sorted

**Example:** For a “remove” algorithm that removes an element from a collection, we want to prove that the resulting collection does not contain the element

# Theorem Proving - Formal Verification

Benefits and drawbacks of formal verification by theorem proving:

- exhaustive (cover all cases)
- rigorous (includes all required logical steps)
- difficult and time consuming
- lead to large proofs
- relies on the correctness of the prover

# Theorem Proving - Formal Verification

**We will now show how to prove mechanically-checked theorems using the Lean theorem prover**

- developed by Microsoft Research
- it runs in the browser
- you can install it on your machines

**Lean implements a logical system:**

- it is based on a **dependent type theory**, which includes predicate and propositional logic
- it supports **constructive reasoning** and has a **computational interpretation** that allows viewing **proofs as programs** and **propositions as types**
- it implements a sequent calculus
- it relies on **tactics** to apply rules (we will see a few today)

# Let us switch to Lean now

https://leanprover.github.io/live/master/

**see the file called prop.lean**

# Conclusion

**What did we cover today?**

- Theorem proving
- Propositional Logic in Lean
- Proofs as programs in Lean

**Further reading**:

- Chapter 4 of http://leanprover.github.io/logic_and_proof/

**Next time?**

- Theorem proving