COMP4011/8011
Advanced Topics in
Formal Methods and Programming Languages

– Software Verification with Isabelle/HOL –

Peter Höfner

July 21, 2024

Section 1

Introduction

Binary Search (`java.util.Arrays`)

```
1:     public static int binarySearch(int[] a, int key) {
2:         int low = 0;
3:         int high = a.length - 1;
4:
5:         while (low <= high) {
6:             int mid = (low + high) / 2;
7:             int midVal = a[mid];
8:
9:             if (midVal < key)
10:                 low = mid + 1
11:              else if (midVal > key)
12:                  high = mid - 1;
13:              else
14:                  return mid; // key found
15:         }
16:         return -(low + 1);  // key not found.
17:    }
```

```
6:             int mid = (low + high) / 2;
```

http://googleresearch.blogspot.com/2006/06/extra-extra-read-all-about-it-nearly.html

What you will learn

- how to use a theorem prover
- background, how it works
- how to prove and specify
- how to reason about programs

Health Warning

Theorem Proving may be addictive

Prerequisites

This is an advanced course. It assumes knowledge in

- Functional programming
- First-order formal logic

The following program should make sense to you:

```
map f []       =   []
map f (x:xs)   =   f x : map f xs
```

You should be able to read and understand this formula:

$$\exists x.\ (P(x) \longrightarrow \forall x.\ P(x))$$

Increase chance to succeed

you should:

- attend lectures
- try Isabelle early
- redo all the demos alone
- try the exercises/homework we give, when we do give some
- **DO NOT CHEAT**
  - assignments and exams are take-home. This does NOT mean you can work in groups. Each submission is personal.
  - for more info, see Plagiarism Policy

What is a formal proof?

A derivation in a formal calculus

Example: $A \wedge B \longrightarrow B \wedge A$ derivable in the following system

Rules:

$$\frac{X \in S}{S \vdash X} \text{ (assumption)} \qquad \frac{S \cup \{X\} \vdash Y}{S \vdash X \longrightarrow Y} \text{ (impI)}$$

$$\frac{S \vdash X \quad S \vdash Y}{S \vdash X \wedge Y} \text{ (conjI)} \qquad \frac{S \cup \{X, Y\} \vdash Z}{S \cup \{X \wedge Y\} \vdash Z} \text{ (conjE)}$$

Proof:

1.	$\{A, B\} \vdash B$	(by assumption)
2.	$\{A, B\} \vdash A$	(by assumption)
3.	$\{A, B\} \vdash B \wedge A$	(by conjI with 1 and 2)
4.	$\{A \wedge B\} \vdash B \wedge A$	(by conjE with 3)
5.	$\{\} \vdash A \wedge B \longrightarrow B \wedge A$	(by impI with 4)

What is a theorem prover?

Implementation of a formal logic on a computer.

- fully automated (propositional logic)
- automated, but not necessarily terminating (first order logic)
- with automation, but mainly interactive (higher order logic)

There are other (algorithmic) verification tools:

- model checking, static analysis, ...
- See COMP3710: Algorithmic Verification (S2 2022) or COMP4130

Why theorem proving?

- Analyse systems/programs thoroughly
- Find design and specification errors early
- High assurance: mathematical, machine checked proofs
- It's not always easy
- It's fun!

Main theorem proving system for this course

Isabelle

What is Isabelle?

A generic interactive proof assistant

- **generic**
  not specialised to one particular logic
  (two large developments: HOL and ZF, will mainly use HOL)
- **interactive**
  more than just yes/no, you can interactively guide the system
- **proof assistant**
  helps to explore, find, and maintain proofs

Correctness

If I prove it on the computer, it is correct, right?

No, because:

1. hardware could be faulty
2. operating system could be faulty
3. implementation runtime system could be faulty
4. compiler could be faulty
5. implementation could be
6. logic could be inconsistent
7. theorem could mean something else

Correctness

If I prove it on the computer, it is correct, right?

No, but: probability for

- OS and H/W issues reduced by using different systems
- runtime/compiler bugs reduced by using different compilers
- faulty implementation reduced by having the right prover architecture
- inconsistent logic reduced by implementing and analysing it
- wrong theorem reduced by expressive/intuitive logics

No guarantees, but assurance immensely higher than manual proof

Meta Logic

Meta language:
The language used to talk about another language.

Examples:
English in a Spanish class, English in an English class

Meta logic:
The logic used to formalise another logic

Example:
Mathematics used to formalise derivations in formal logic

Meta Logic – Example

Syntax:

Formulae: $F ::= V \mid F \longrightarrow F \mid F \wedge F \mid False$

$V ::= [A - Z]$

Judgement: $S \vdash X$ X a formula, S a set of formulae

logic / meta logic

$$\frac{X \in S}{S \vdash X} \qquad \frac{S \cup \{X\} \vdash Y}{S \vdash X \longrightarrow Y}$$

$$\frac{S \vdash X \quad S \vdash Y}{S \vdash X \wedge Y} \qquad \frac{S \cup \{X, Y\} \vdash Z}{S \cup \{X \wedge Y\} \vdash Z}$$

Isabelle's Meta Logic

$$\bigwedge \qquad \Longrightarrow \qquad \lambda$$

$\bigwedge$

Syntax: $\bigwedge x.\ F$ (F another meta logic formula)
in ASCII: `!!x. F`

- this is the meta-logic universal quantifier
- example and more later

$\Longrightarrow$

Syntax: $A \Longrightarrow B$ (A, B other meta logic formulae)
in ASCII: `A ==> B`

Binds to the right:

$$A \Longrightarrow B \Longrightarrow C \quad = \quad A \Longrightarrow (B \Longrightarrow C)$$

Abbreviation:

$$[\![A; B]\!] \Longrightarrow C \quad = \quad A \Longrightarrow B \Longrightarrow C$$

- read: A and B implies C
- used to write down rules, theorems, and proof states

Example: a theorem

mathematics: if $x < 0$ and $y < 0$, then $x + y < 0$

formal logic: $\vdash\ x < 0 \wedge y < 0 \longrightarrow x + y < 0$

variation: $x < 0; y < 0 \ \vdash\ x + y < 0$

Isabelle: **lemma** "$x < 0 \wedge y < 0 \longrightarrow x + y < 0$"

variation: **lemma** "$[\![x < 0; y < 0]\!] \Longrightarrow x + y < 0$"

variation: **lemma**

assumes "$x < 0$" and "$y < 0$" shows "$x + y < 0$"

Example: a rule

logic: $$\frac{X \quad Y}{X \wedge Y}$$

variation: $$\frac{S \vdash X \quad S \vdash Y}{S \vdash X \wedge Y}$$

Isabelle: $[\![X; Y]\!] \Longrightarrow X \wedge Y$

Example: a rule with nested implication

logic:

$$\frac{X \vee Y \quad \begin{matrix} X \\ \vdots \\ Z \end{matrix} \quad \begin{matrix} Y \\ \vdots \\ Z \end{matrix}}{Z}$$

variation:

$$\frac{S \cup \{X\} \vdash Z \quad S \cup \{Y\} \vdash Z}{S \cup \{X \vee Y\} \vdash Z}$$

Isabelle:

$$[\![X \vee Y; X \Longrightarrow Z; Y \Longrightarrow Z]\!] \Longrightarrow Z$$

λ

Syntax: $\lambda x.\ F$ (F another meta logic formula)
in ASCII: `%x. F`

- lambda abstraction
- used to represent functions
- used to encode bound variables
- more about this soon