**NICTA** 

#### Organisatorials



NICTA

 When
 Wed
 9:00 - 10:30

 Fri
 9:00 - 10:30

 Where
 Mon:
 MatSc G10

 Fri:
 MatSc G11

http://www.cse.unsw.edu.au/~cs4161/

Slide 3

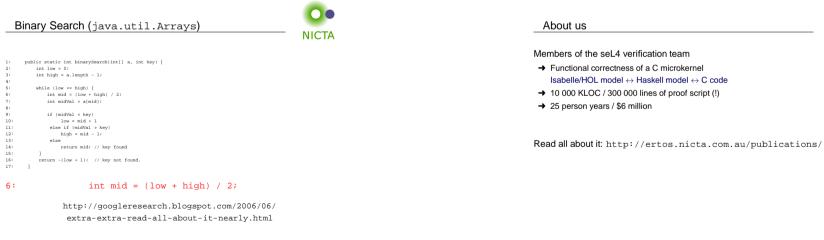
Slide 1

COMP 4161

NICTA Advanced Course

Advanced Topics in Software Verification

Simon Winwood, Toby Murray, June Andronick, Gerwin Klein





#### 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** 

NICTA

NICTA

#### **Theorem Proving is addictive**

Slide 5

Content — Using Theorem Provers

- → Intro & motivation, getting started (today)
- ➔ Foundations & Principles
  - Lambda Calculus
  - Higher Order Logic, natural deduction
  - Term rewriting
- → Proof & Specification Techniques
  - Datatypes, recursion, induction
  - Inductively defined sets, rule induction
  - Calculational reasoning, mathematics style proofs
  - Hoare logic, proofs about programs

Credits



some material (in using-theorem-provers part) shamelessly stolen from



Tobias Nipkow, Larry Paulson, Markus Wenzel



Don't blame them, errors are mine

Slide 7

#### What is a proof?



- → from Latin probare (test, approve, prove)
- → to learn or find out by experience (archaic)
- → to establish the existence, truth, or validity of (by evidence or logic)
  - prove a theorem, the charges were never proved in court

#### pops up everywhere

- → politics (weapons of mass destruction)
- → courts (beyond reasonable doubt)
- → religion (god exists)
- → science (cold fusion works)

Slide 6

Slide 8



(Marriam-Webster)



#### What is a mathematical proof?

NICTA

#### In mathematics, a proof is a demonstration that, given certain axioms, some statement of interest is necessarily true. (Wikipedia)

#### **Example:** $\sqrt{2}$ is not rational.

Proof: assume there is  $r \in \mathbb{Q}$  such that  $r^2 = 2$ .

Hence there are mutually prime p and q with  $r = \frac{p}{q}$ .

Thus  $2q^2 = p^2$ , i.e.  $p^2$  is divisible by 2.

2 is prime, hence it also divides p, i.e. p = 2s.

Substituting this into  $2q^2 = p^2$  and dividing by 2 gives  $q^2 = 2s^2$ . Hence, q is also divisible by 2. Contradiction. Qed.

#### Slide 9



- → still not rigorous enough for some
  - what are the rules?
  - · what are the axioms?
  - how big can the steps be?
  - what is obvious or trivial?
- → informal language, easy to get wrong
- → easy to miss something, easy to cheat

#### Theorem. A cat has nine tails.

**Proof.** No cat has eight tails. Since one cat has one more tail than no cat, it must have nine tails.

#### What is a formal proof?



NICTA

#### A derivation in a formal calculus

Example: $A \land B$ —	$ ightarrow B \wedge A$ derivable in the following system
	$C \mapsto (\mathbf{V}) \vdash \mathbf{V}$

Rules:	$rac{X \in S}{S \vdash X}$ (assumption)	$\frac{S \cup \{X\} \vdash Y}{S \vdash X \longrightarrow Y} \text{ (impl)}$
	$\frac{S \vdash X  S \vdash Y}{S \vdash X \land Y}$ (conjl)	$\frac{S \cup \{X,Y\} \vdash Z}{S \cup \{X \land Y\} \vdash Z} \text{ (conjE)}$
Proof:		

1.	$\{A, B\} \vdash B$	(by assumption)
2.	$\{A,B\}\vdash A$	(by assumption)
3.	$\{A,B\} \vdash B \land A$	(by conjl with 1 and 2)
4.	$\{A \land B\} \vdash B \land A$	(by conjE with 3)
5.	$\{\} \vdash A \land B \longrightarrow B \land A$	(by impl with 4)

Slide 11

#### 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)
- → based on rules and axioms
- → can deliver proofs

There are other (algorithmic) verification tools:

- → model checking, static analysis, ...
- → usually do not deliver proofs

Slide 10

#### Why theorem proving?



NICTA

- → Analysing systems/programs thoroughly
- → Finding design and specification errors early
- → High assurance (mathematical, machine checked proof)
- → it's not always easy
- → it's fun

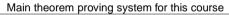
#### What is Isabelle?



#### 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

0





Slide 13

Isabelle

→ used here for applications, learning how to prove

#### Why Isabelle?



- → free
- → widely used systems
- → active development
- ➔ high expressiveness and automation
- → reasonably easy to use
- → (and because we know it best ;-))

Slide 14

Slide 16

### **NICTA**

NICTA

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



NICTA

#### 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 right architecture
- → inconsistent logic reduced by implementing and analysing it
- → wrong theorem reduced by expressive/intuitive logics

No guarantees, but assurance immensly higher than manual proof

Slide 19

Slide 17

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

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
- ( compiler could be faulty
- (5) implementation could be faulty
- 6 logic could be inconsistent
- $\ensuremath{\mathfrak{O}}$  theorem could mean something else

## If I prove it on the computer, it is correct, right? Soundness architectures careful implementation PVS

LCF approach, small proof kernel	HOL4 Isabelle
explicit proofs + proof checker	Coq Twelf
	Isabelle
	HOL4

Slide 18

#### Meta logic:

Examples:

Meta Logic

Meta language:

The logic used to formalize another logic

#### Example: Mathematics used to formalize derivations in formal logic

The language used to talk about another language.

Isabelle's Meta Logic



NICTA



Slide 23

Slide 21

Meta	Logic – Ex	ample	NIC
Syntax:	Formulae:	$\begin{array}{cccc} F ::= V &   & F \longrightarrow F &   & F \wedge F \\ V ::= & [A-Z] \end{array}$	False
	Derivable:	$S \vdash X$ X a formula, S a set of for	mulae
		logic / meta logic	
		$\frac{X \in S}{S \vdash X} \qquad \qquad \frac{S \cup \{X\} \vdash Y}{S \vdash X \longrightarrow Y}$	

D	•	
C	ГA	

\_

NICTA

Syntax:  $\bigwedge x. F$  (*F* another meta level formula) in ASCII: 11x. F

→ universal quantifier on the meta level

- → used to denote parameters
- → example and more later

Slide 22

 $\frac{S \vdash X \quad S \vdash Y}{S \vdash X \land Y}$ 

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

		-

NICTA

**Syntax:**  $A \Longrightarrow B$  (A, B other meta level formulae) in ASCII:  $A \implies B$ 

#### Binds to the right:

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

#### Abbreviation:

 $\implies$ 

 $\llbracket A;B\rrbracket \Longrightarrow C \quad = \quad A \Longrightarrow B \Longrightarrow C$ 

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

#### Slide 25

Example: a	theorem	NICTA
		mont
mathematics:	if $x < 0$ and $y < 0$ , then $x + y < 0$	
formal logic:	$\vdash \ x < 0 \land y < 0 \longrightarrow x + y < 0$	
variation:	$x<0; y<0 \ \vdash \ x+y<0$	
Isabelle:	lemma " $x < 0 \land y < 0 \longrightarrow x + y < 0$ "	
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 Image: X Y X A Y logic: $\frac{X Y}{X A Y}$ variation: $\frac{S \vdash X S \vdash Y}{S \vdash X A Y}$ lsabelle: $[X;Y] \Longrightarrow X \land Y$

Slide 27

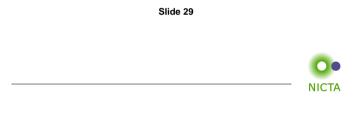
Example: a r	ule with nested implication	- NICTA
logic:	$\begin{array}{ccc} X & Y \\ \vdots & \vdots \\ \underline{X \lor Y} & \underline{Z} & \underline{Z} \\ Z \end{array}$	
variation:	$\frac{S \cup \{X\} \vdash Z  S \cup \{Y\} \vdash Z}{S \cup \{X \lor Y\} \vdash Z}$	
Isabelle:	$\llbracket X \lor Y; X \Longrightarrow Z; Y \Longrightarrow Z \rrbracket \Longrightarrow Z$	

Slide 26



**Syntax:**  $\lambda x. F$  (*F* another meta level formula) in ASCII: x. F

- → lambda abstraction
- → used for functions in object logics
- → used to encode bound variables in object logics
- → more about this in the next lecture



ENOUGH THEORY! GETTING STARTED WITH ISABELLE