Hardware & Software Verification

John Wickerson & Pete Harrod

Lecture 5: Isabelle

Lecture Outline

- Proving simple theorems by hand.
- Proving simple theorems using Isabelle.
- **Next lecture:** proving the correctness of a logic synthesiser.

First proof

- **Theorem.** $\sqrt{2}$ is irrational.

Isabelle

- Invented by Lawrence Paulson around 1986. Developed ever since at the University of Cambridge and at TU München.

- Has been used for large mathematical proofs, such as the Kepler conjecture.

- Has been used to build a verified operating system! The OS implementation is about 7.5k lines of C, the proof has about 200k steps, and it uncovered hundreds of bugs in the initial implementation.

seL4

Security. Performance. Proof.


DEMO

Observations

- Use `sorry` to skip a proof.
- Use `find_theorems` to search Isabelle's database of theorems.
- CTRL+click (or CMD+click) on a name to jump to its definition.
- Use `thm` to print out a theorem. Use `thm[of x]` or `thm[OF f]` to print out an instantiated theorem.
- Refer to facts using `` `backticks` `` or by naming them.
- Use `try` to invoke the Sledgehammer.

Second proof

- **Theorem.** There is no greatest even number.

- **Proof.** To show that the greatest number does *not* exist, we shall assume that it *does*, and deduce a contradiction. To this end, suppose there *is* a greatest even number, and call it *n*. But if n is even, then so is *n*+2, which is greater than *n*. This contradicts the assumption that *n* is the greatest even number. Therefore, the greatest even number does not exist.


DEMO

Observations

- Use `moreover..ultimately` to avoid labelling each fact.
- Isabelle proofs can use the "structured" style or the "procedural" style.
- The procedural style offers various low-level commands like `defer` and `prefer`, and low-level methods like `thin_tac` and `rename_tac`.
- There are a range of automated methods: `auto`, `simp`, `clarify`, `clarsimp`, `blast`, etc.

Some constructions

- `fix` *<variable name>*
- `assume` *<new fact>*
- `have` *<new fact>* `by` *<method>*
- ~~`from this have`~~ `hence` *<new fact>* `by` *<method>*
- `with` *<name of old fact>* `have` *<new fact>* `by` *<method>*
- `have` *<new fact>* `using` *<name of old fact>* `by` *<method>*
- `show` *<thesis>* `by` *<method>*
- ~~`from this show`~~ `thus` *<thesis>* `by` *<method>*
- `moreover..ultimately`

Meta vs Object logic

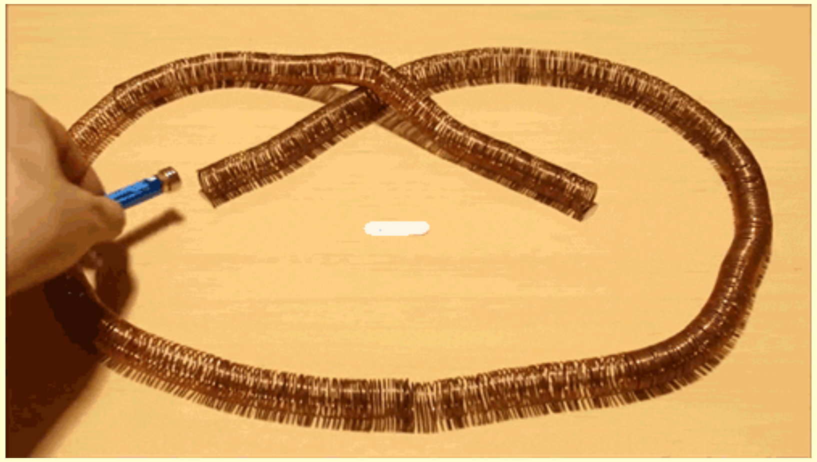
- This is the difference between making a judgement about a logical statement and the logical statement itself.
- Examples:
  - For every x, if it is the case that even(x) holds and it is the case that odd(x) holds then it is the case that x=0 holds.
  - For every x, if it is the case that $\text{even}(x) \wedge \text{odd}(x)$ holds then it is the case that x=0 holds.
  - For every x, it is the case that $(\text{even}(x) \wedge \text{odd}(x)) \longrightarrow x{=}0$ holds.
  - It is the case that $\forall x.\ (\text{even}(x) \wedge \text{odd}(x)) \longrightarrow x{=}0$ holds.

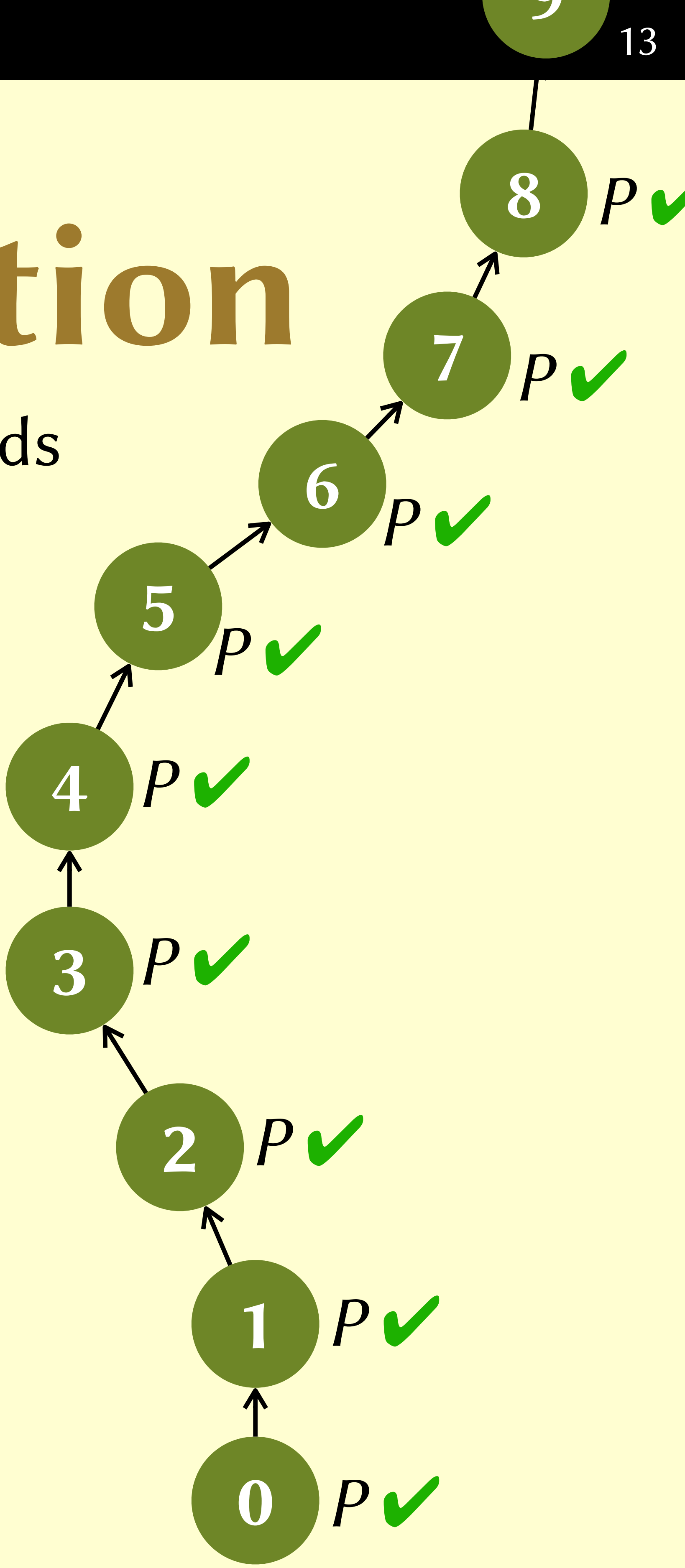
Meta vs Object logic

- This is the difference between making a judgement about a logical statement and the logical statement itself.
- Examples:
    - $\bigwedge x.\ [\![\mathrm{even}(x); \mathrm{odd}(x)]\!] \Longrightarrow x=0$
    - $\bigwedge x.\ \mathrm{even}(x) \wedge \mathrm{odd}(x) \Longrightarrow x=0$
    - $\bigwedge x.\ \mathrm{even}(x) \wedge \mathrm{odd}(x) \longrightarrow x=0$
    - $\forall x.\ (\mathrm{even}(x) \wedge \mathrm{odd}(x)) \longrightarrow x=0$

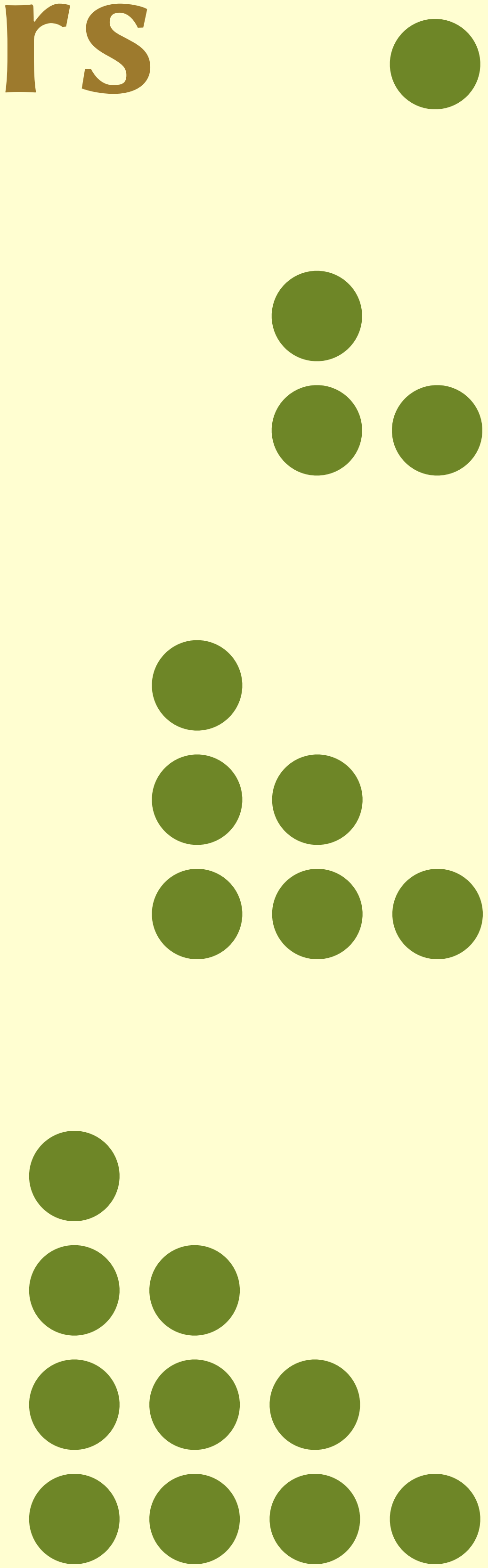
Proof by induction

- Suppose we want to show that property *P* holds for all natural numbers.
- To do this, it suffices to prove two things:
    - *P* holds for 0 (this is called the **base case**), and
    - for all k, if *P* holds for k, then *P* also holds for k+1 (this is called the **inductive step**).





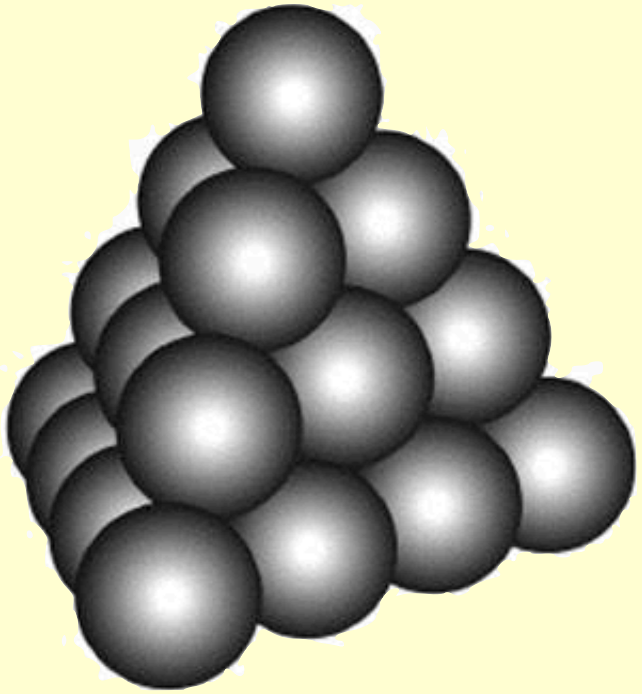
Triangle numbers



- triangle(n) = **if** n=0 **then** 0 **else** n + triangle(n-1)
- **Theorem.** triangle(n) = (n+1)n/2.
- **Proof.** We proceed by mathematical induction.
  - *Base case.* We have triangle(0) = (0+1)0/2 = 0.
  - *Inductive step.* Pick arbitrary k and assume triangle(k) = (k+1)k/2. It follows that triangle(k+1) = k+1 + triangle(k) = k+1 + (k+1)k/2 = (k+2)(k+1)/2, as required.


DEMO

Tetrahedral numbers



- tet(n) = **if** n=0 **then** 0 **else** triangle(n) + tet(n-1)
- **Theorem.** tet(n) = (n+2)(n+1)n/6.
- **Proof.** We proceed by mathematical induction.
  - *Base case.* We have tet(0) = (0+2)(0+1)0/6 = 0.
  - *Inductive step.* Pick arbitrary k and assume tet(k) = (k+2)(k+1)k/6. With the help of the previous theorem about triangle numbers, it follows that tet(k+1) = (k+3)(k+2)(k+1)/6.


DEMO

Observations

- Use `also..finally` for chains of equational reasoning.
- Isabelle will provide a bare-bones induction proof for you when you type `proof (induct ...)`.
- Use `{` braces `}` to delimit the scope of a local assumption.

Summary

- **This lecture:** how to conduct some basic proofs in Isabelle.
- **Next lecture:** How to implement a (small) logic synthesiser in Isabelle and verify that it is correct.