AR Coursework Lecture

Filip Smola

24th October 2024

Information

- Demonstrator/TA: Filip Smola `f.smola@ed.ac.uk`
- Lab sessions: Mondays from 9am to 11am in 4.12, Appleton Tower
- Submission deadline: 12 noon on 18th November
- Isabelle is installed on DICE machines
- You should try the Isabelle exercises from the course page

Part 1

Prove some propositional and first-order statements (procedural style)

Only allowed tactics:

- `rule`, `rule_tac`,
- `drule`, `drule_tac`,
- `erule`, `erule_tac`,
- `frule`, `frule_tac`,
- `cut_tac`,
- `assumption`

Only allowed rules:

- `exI`, `exE`,
- `allI`, `allE`, `spec`,
- `conjI`, `conjE`, `conjunct1`, `conjunct2`,
- `ccontr`, `excluded_middle`,
- `notI`, `notE`, `notnotD`,
- `impI`, `impE`, `mp`,
- `iffI`, `iffE`, `iffD1`, `iffD2`,
- `disjI1`, `disjI2`, `disjE`,
- And any lemmas you prove in this way

Part 1

Prove some propositional and first-order statements (procedural style)

Only allowed tactics:

- `rule`, `rule_tac`,
- `drule`, `drule_tac`,
- `erule`, `erule_tac`,
- `frule`, `frule_tac`,
- `cut_tac`,
- `assumption`

Only allowed rules:

- `exI`, `exE`,
- `allI`, `allE`, `spec`,
- `conjI`, `conjE`, `conjunct1`, `conjunct2`,
- `ccontr`, `excluded_middle`,
- `notI`, `notE`, `notnotD`,
- `impI`, `impE`, `mp`,
- `iffI`, `iffE`, `iffD1`, `iffD2`,
- `disjI1`, `disjI2`, `disjE`,
- And any lemmas you prove in this way

No automatic proof methods (`auto`, `blast`, `simp`, ...)!

Part 1

Structure

First 21 marks is seven conjectures you are asked to prove

Part 1

Structure

First 21 marks is seven conjectures you are asked to prove

Remaining 25 marks is four conjectures in a puzzle setting:

- Island contains only sparrows and magpies.
- Sparrows always tell the truth.
- Magpies always like.
- There are two puzzles based on conversations with the inhabitants.

Part 2

Verify software using Hoare logic (structured style)

Additionally allowed any rules (without changing the imports) and the following tactics:

- `subst`, `unfold`,
- `auto`, `simp`, `blast`, `safe`,
- `fast`, `force`, `fastforce`,
- `presburger`,
- `algebra`, `arith`, `linarith`,
- `vcg`, `vcg_simp`

Part 2

Verify software using Hoare logic (structured style)

Additionally allowed any rules (without changing the imports) and the following tactics:

- `subst`, `unfold`,
- `auto`, `simp`, `blast`, `safe`,
- `fast`, `force`, `fastforce`,
- `presburger`,
- `algebra`, `arith`, `linarith`,
- `vcg`, `vcg_simp`

You are allowed to use Sledgehammer, but **not** allowed to use the tactics `metis`, `meson` and `smt`.

Part 2

Simple Algorithms

First 9 marks is three simple programs:

- Iteratively copying an integer,
- Multiplying two integers,
- Dividing one integer by another.

$\rightarrow$ Provide all loop invariants, one post condition and verify them

Part 2

Minimal Sum

Remainder revolves around an algorithm that finds the minimal sum of an array, described in detail in Huth & Ryan Section 4.3.3

Two specifications:

- At the end of the program, `s` is less than or equal to the sum of any section of the array.
- There exists a section of the array that has sum `s`.

Part 2

Minimal Sum

Remainder revolves around an algorithm that finds the minimal sum of an array, described in detail in Huth & Ryan Section 4.3.3

Two specifications:

- At the end of the program, `s` is less than or equal to the sum of any section of the array.
- There exists a section of the array that has sum `s`.

Given the program and specification formalisations:

- Prove the first specification given the relevant invariants and two intermediate conjectures. (25 marks)
- Prove the second specification, coming up with your own invariant. (20 marks)

Useful Tips

- You can add theorems to the simplifier to be used automatically:
  apply `(simp add: add_0_left)`

Useful Tips

- You can add theorems to the simplifier to be used automatically:
  apply `(simp add: add_0_left)`
- You can search for theorems based on constants:
  find_theorems `"(+)"` `zero`

Useful Tips

- You can add theorems to the simplifier to be used automatically:
  apply `(simp add: add_0_left)`
- You can search for theorems based on constants:
  find_theorems `"(+)" zero`
- You can instantiate meta-variables in theorems:

```
lemma foo:
      fixes k m n :: int
    assumes "k + m ≤ k + n"
      shows "m ≤ n"
    using assms by simp

lemma "(200 :: int) ≤ 345"
  apply (rule foo)
  sorry
```

Useful Tips

- You can add theorems to the simplifier to be used automatically:
  apply `(simp add: add_0_left)`
- You can search for theorems based on constants:
  find_theorems `"(+)" zero`
- You can instantiate meta-variables in theorems:

```
lemma foo:
      fixes k m n :: int
    assumes "k + m ≤ k + n"
      shows "m ≤ n"
    using assms by simp
lemma "(200 :: int) ≤ 345"
    apply (rule_tac k = "-200" in foo)
    sorry
```

Useful Tips

- You can add theorems to the simplifier to be used automatically:
  apply `(simp add: add_0_left)`
- You can search for theorems based on constants:
  find_theorems `"(+)" zero`
- You can instantiate meta-variables in theorems:

```
lemma foo:
      fixes k m n :: int
    assumes "k + m ≤ k + n"
      shows "m ≤ n"
    using assms by simp
lemma "(200 :: int) ≤ 345"
    apply (rule foo[of "-200"])
    sorry
```

Useful Tips

- You can add theorems to the simplifier to be used automatically:
  apply `(simp add: add_0_left)`
- You can search for theorems based on constants:
  find_theorems `"(+)" zero`
- You can instantiate meta-variables in theorems:

```
lemma foo:
      fixes k m n :: int
    assumes "k + m ≤ k + n"
      shows "m ≤ n"
    using assms by simp
lemma "(200 :: int) ≤ 345"
  apply (rule foo[where k = "-200"])
  sorry
```

Useful Tips

- You can add theorems to the simplifier to be used automatically:
  apply `(simp add: add_0_left)`
- You can search for theorems based on constants:
  find_theorems `"(+)" zero`
- You can instantiate meta-variables in theorems:

```
lemma foo:
      fixes k m n :: int
    assumes "k + m ≤ k + n"
      shows "m ≤ n"
    using assms by simp
lemma "(200 :: int) ≤ 345"
  apply (rule foo[where k = "-200" and m = 200 and n = 345])
  sorry
```

Overview

- Submission deadline: 12 noon on 18th November
- Automated tactics (`auto`, `simp`, ...) can be used from Part 2 onwards
- Use "Query" panel or **find_theorems** to search for useful theorems
- Break things into sublemmas, especially in Part 2
- Ask questions (labs, Piazza, email)