Technische Universität München
Institut für Informatik
Prof. Tobias Nipkow, Ph.D.
Fabian Huch

WS 2022/23
25. 10. 2022

Semantics of Programming Languages

Exercise Sheet 1

Before beginning to solve the exercises, open a new theory file named `Ex01.thy` and write the the following three lines at the top of this file.

theory *Ex01*
imports *Main*
begin

Exercise 1.1 Calculating with natural numbers

Use the **value** command to turn Isabelle into a fancy calculator and evaluate the following natural number expressions:

"$2 + (2::nat)$" "$(2::nat) * (5 + 3)$" "$(3::nat) * 4 - 2 * (7 + 1)$"

Can you explain the last result?

Exercise 1.2 Natural number laws

Formulate and prove the well-known laws of commutativity and associativity for addition of natural numbers.

Exercise 1.3 Counting elements of a list

Define a function which counts the number of occurrences of a particular element in a list.

fun *count* :: "$'a\ list \Rightarrow 'a \Rightarrow nat$"

Test your definition of *count* on some examples and prove that the results are indeed correct.

Prove the following inequality (and additional lemmas, if necessary) about the relation between *count* and *length*, the function returning the length of a list.

theorem "$count\ xs\ x \leq length\ xs$"

Exercise 1.4 Adding elements to the end of a list

Recall the definition of lists from the lecture. Define a function *snoc* that appends an element at the right end of a list. Do not use the existing append operator @ for lists.

fun *snoc* :: *"'a list ⇒ 'a ⇒ 'a list"*

Convince yourself on some test cases that your definition of *snoc* behaves as expected, for example run:

value *"snoc [] c"*

Also prove that your test cases are indeed correct, for instance show:

lemma *"snoc [] c = [c]"*

Next define a function *reverse* that reverses the order of elements in a list. (Do not use the existing function *rev* from the library.) Hint: Define the reverse of *x # xs* using the *snoc* function.

fun *reverse* :: *"'a list ⇒ 'a list"*

Demonstrate that your definition is correct by running some test cases, and proving that those test cases are correct. For example:

value *"reverse [a, b, c]"*

lemma *"reverse [a, b, c] = [c, b, a]"*

Prove the following theorem. Hint: You need to find an additional lemma relating *reverse* and *snoc* to prove it.

theorem *"reverse (reverse xs) = xs"*

Homework Registration

Submissions are handled via https://do.proof.in.tum.de. Register an account in the system and send the tutor an e-mail of the following format:

- subject: `[Semantics 2022/23] homework registration`
- content: `<username>,<student id (Matrikelnr.)>,<first name>,<last name>`

(replacing `<...>` by actual values).

Homework Submission

- Use the template from the competition "Semantics 2022/23". **Do not** change the existing code of the template (except for the `sorry`s and `undefined`s), only add your solution.
- submit your solution following the instructions on the website.
- The system will check that your solution can be loaded in Isabelle-2021 **without any errors**.
- You can upload multiple times; the last upload before the deadline is the one that will be graded.
- The submission system will give you feedback which checks were passed. Some checks are listed multiple times for weighting.

Homework Guidelines

- Only submissions with status "Passed" will be graded. If you have any problems uploading, or if the submission seems to be rejected for reasons you cannot understand, please contact the tutor before the deadline. Make sure that the submission (and check file) runs through locally without errors.
- Partial credits may be given for:
  - nearly correct definitions
  - finished intermediate lemmas
  - incomplete proofs, if they do not contain sorry and missing steps are extracted into succinct lemmas (which are assumed by using sorry).
- To claim partial credit (e.g., if you made progress in a proof but didn't finish it), **Mark it as `(*incomplete*)`**.
- We will be using a clone detection tool to compare solutions so please do not add any personal or identifying information.

Homework 1 Prefix of a list

Submission until Monday, October 31, 23:59pm.

Specify a function that given a list of natural numbers and a possible element in the list, gives the shortest prefix of the given list that ends with the given element.

fun *find_pfx* :: *"nat list ⇒ nat ⇒ nat list"*

Test cases:

value *"find_pfx [1::nat,2,3] 2 = [1,2]"*
value *"find_pfx [] (1::nat) = []"*

Prove that the prefix of a list (l1@l2) is the same as the prefix of l1, as long as the element we are searching for is at the end of l1.

lemma *find_pfx_append*:
"(find_pfx (xs1 @ [x] @ xs2) x) = (find_pfx (xs1 @ [x]) x)"

Show that the last element in the computed prefix is the element we search for. Note: in Isabelle, to express that an element is in a list, we write $x \in$ *set xs*. (to enter the $\in$ sign, type in backslash-in.) Also, *last xs* denotes the last element of the list *xs*.

lemma *last_find_pfx_val*:
"last (find_pfx (xs @ [x]) x) = x"

Hint: You'll need an auxiliary lemma showing that the computed prefix is not empty if the elem is in the list.

Prove that searching in a 'duplicated' list returns the same result:

lemma *pfx_append_same*: *"x ∈ set xs ⟹ find_pfx (xs @ xs) x = find_pfx xs x"*

Hint: You need to generalize the lemma first.