Technical University of Munich
Chair for Logic and Verification
Prof. Tobias Nipkow, Ph.D.
J. Rädle, L. Stevens, K. Kappelmann, MC M. Eberl

WS 2019/20
15.11.2019
Deadline: 25.11.2019, 23:59

Functional Programming and Verification

Sheet 5

IMPORTANT: We use the cyp ("Check your proof") format for our proofs. Use the templates from the submission website to check your proofs by structural induction. The submission system can check these proofs against the provided templates.

Each proof step must use *one* of these defining equations, the inductive hypothesis (IH), or an axiom. You have to state the justification for each step.

Tutorial Exercises

Exercise T5.1 Type Inference

Determine the most general types of the following definitions:

```
f u v = u < 0 && u < v
f u v w = u + v
f u v = min (head u) v
f = 0
f u v = let w = u && v in []
f u v = let w = u && v in if w then [u] else []
f u v = concat [u | (d,_)<-v, u==d]
f u = [x | ((x:xs):zs)<-u]
```

Exercise T5.2 Structural Induction ☐

We define two functions `snoc :: [a] -> a -> [a]` and `reverse :: [a] -> [a]` as follows:

```
snoc [] y = [y]
snoc (x : xs) y = x : snoc xs y

reverse [] = []
reverse (x : xs) = snoc (reverse xs) x
```

1. Use structural induction to prove the following equation

   ```
   reverse (snoc xs x) = x : reverse xs
   ```

2. Use structural induction to prove the following equation

   ```
   reverse (reverse xs) = xs
   ```

Exercise T5.3 (Optional) Pattern Matching $?x \equiv ?y$

Write a function `match :: String -> String -> Bool`, that compares a string to a pattern. The pattern is itself a string that uses `?` to match a single arbitrary character and `*` to match an arbitrary sequence of (zero or more) characters. In order to match, the pattern must recognize the entire target string. The function call `match ps ys` uses `ps` as the pattern and `ys` as the target string.

The following examples should all return `True`:

```
match "abc" "abc"               not (match "ab" "abc")
match "?bc" "abc"               not (match "a" "")
match "*" "abc"                 not (match "a*b" "bba")
match "a*f" "abcdef"            not (match "a*f" "abcde")
match "a***b?" "abc"            not (match "a***b" "abc")
```

Homework

IMPORTANT: The submission system can process multiple file at once. You have to upload all files (h51.cprf, h52.cprf, and Exercise_5.hs) at once. Moreover, you have to use the provided filenames when uploading, that is do not change the filenames of the template files.

You need to collect 2 out of 3 points (P) to pass this sheet.

The needed background theories/definitions (*.cthy files) for the following exercises can be found on moodle and on the submission system. You can use axioms in a similar way as you use definitions or the IH. For example:

```
Lemma zeroAddzeroAdd: 0 + 0 + a .=. a
Proof
                                   0 + 0 + a
    (by zeroAdd)               .=. 0 + a
    (by zeroAdd)               .=. a
QED
```

Exercise H5.1 sum sum [1P]

Use structural induction to prove the following identity

```
  sum (xs ++ ys) = sum xs + sum ys
```

Exercise H5.2 sumsibum [1P]

Use structural induction to prove the following identity

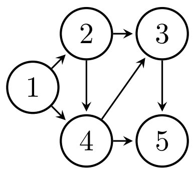
```
    sum (mapLength (mapAppend xs yss))
  = sum (mapLength yss) + (length xs * length yss)
```

Exercise H5.3 (**Competition**) Procrastination in Garchosibirsk [1P]

The MC Jr. is overloaded with work. As a natural consequence, he is trying to procrastinate by doing a walk around the beautiful Garchosibirsk campus. His fellow *Übungsleiter* Lukas, has a strong love for optimisation and hence suggests him to make the most out of his walk by maximising its length using a longest path algorithm. The MC Jr. does like the idea, but he also needs to make sure that he will not be out for too long since he needs to catch the last *U-Bahn*. Moreover, the MC Jr. sadly forgot all the graph theory he had to learn during his Bachelor's and hence decides to delegate the work to you.

You are given an unweighted, directed graph $G = (V, E)$ consisting of vertices $V \subseteq \mathbb{Z}$ and edges $E \subseteq V \times V$, where $(v, w) \in E$ corresponds to an edge from v to w. To our great luck, the campus is an acyclic graph, that is there are no cycles. Moreover, there is exactly one node that has no incoming edges, which represents the starting point of the MC Jr. Your task is to implement a function `longestPath :: Graph -> Vertex -> Int` such that `longestPath g t` returns the length of the longest path in the given directed acylic graph `g` from the starting point of the MC Jr. to a given target node `t`. For example:

```
longestPath
  ([1..5], [(1,2),(2,3),(2,4),(3,5),(1,4),(4,3),(4,5)]) 5 = 4
```



The MC Jr. does like to have long walks, but he does detest long waiting times for computations. He is thus willing to award precious competition points to the fastest solutions – only by kind permission of the MC Sr. of course! But even if you do not want to impress the MC Jr., make sure that your algorithm at least runs in polynomial time. As a kind gesture, he also provides you with a function `genDag` in the template that you can use to generate graphs satisfying the constraints described above.

The MC Sr. in fact could not resist creating an optimised solution to help the poor MC Jr. His solution is able to evaluate a graph with 1500 vertices and 400 000 edges in about 2 seconds. That's quite good, but the MC Jr. wishes for even more efficiency! Can you beat the Senior?

Important: If you submit a competition exercise, you agree that we are allowed to publish your name as part of the competition on our website. If you just want to submit a competition exercise as part of your homework without taking part in the competition, you can just remove the `{-WETT-}` ... `{-TTEW-}` comments of your submission.

Trivia

If the uncertainty of testing makes you wild and you instead find great pleasure in having absolute certainty by using rigorous proofs, then you will love the field of *interactive theorem proving* (ITP). No matter if you like program verification, mathematics, or even philosophy: ITPs enable you to reason about all of them in a fully verified way.

In fact, one very popular ITP – Isabelle – is developed and heavily used by our chair. It even ships with some automated reasoners so that proofs as done on this sheet can be solved automatically in many cases. Reach out to one of our researchers if you are curious!

No more "proofs" that look more like LSD trips than coherent chains of logical arguments.

— Scott Aaronson