Technische Universität München
Fakultät für Informatik
Dr. Peter Lammich
Simon Wimmer

WS 2018/19
18.12.2018

Semantics of Programming Languages

Exercise Sheet 10

Exercise 10.1 Procedure Monotonicity

Prove that if we only *add* procedures to the program context, any program execution in the old context is still valid in the new context:

theorem
  assumes "$\forall\, a \in dom\ \pi.\ \pi\ a = \pi'\ a$"
  assumes "π:$(c,s) \Rightarrow t$"
  shows "π':$(c,s) \Rightarrow t$"

Here *dom* π denotes the domain of π, i.e. the set of procedure names in π.

Exercise 10.2 DFS Search, Again

Consider the DFS search from last tutorial. This time we want to refactor the program to pull out elementary operations such as *push* and *pop* on a stack as separate procedures.

- Define *push* and *pop* and insert them in the old program
- Take a look at the existing specifications and invariants, and try to find specifications for *push* and *pop* that will ease the proof
- Use the correct procedures that you know from the lecture for the visited set
- You may need to adjust the invariants from the old program but in the end the proofs should become simple(r)
- Finally turn the code for computing successors into a proper procedure instead of inlining it

Assuming that the input graph is finite, we can also prove that the algorithm terminates. We will thus use an *Isabelle context* to fix a certain finite graph and a start state:

context
  fixes *start* :: *int* **and** *edges*
  assumes *finite_graph*: "*finite* ((*Edges edges*)* `` {*start*})"
begin

Prove the following specification for your program:

program_spec *dfs1*
  assumes *“$0 \le x \wedge 0 \le s \wedge start = s \wedge edges = a \wedge visited$ ‘ $\{x.\ True\} = \{0\}$”*
    ensures *“$b = 1 \longrightarrow x \in (Edges\ a)^*$ ‘‘ $\{s\}$”*
  for *visited*[]

Homework 10.1 Be Original!

Submission until Tuesday, January 8, 2019, 10:00am. (20 regular points, plus bonus points for nice submissions)

Think up a nice formalization yourself. Some inspirations:

- You are particularly encouraged to find an interesting algorithm, to formalize it in the extended IMP language, and to verify with the help of the tools you've seen during the last weeks. Examples:
  - A completeness proof and other extensions of the DFS search
  - BFS on graphs (a proof that BFS computes *shortest* paths is particularly interesting)
  - ...
- Formalize some interesting algorithms/data structures functionally and verify them
- Prove some interesting result about graph/automata/formal language theory
- Formalize some results from mathematics
- Find interesting modifications of IMP material and prove interesting properties about them
- ...

You should set yourself a time limit before starting your project. Also incomplete/unfinished formalizations are welcome and will be graded!

Please comment your formalization well, such that we can see what it does/is intended to do.

Do not start in the last two days or you will not succeed!

You are welcome to discuss your plans with the tutor before starting your project.