Exercise 10.1 Control Flow Graphs

A labeled transition system (LTS) is a directed graph with edge labels. We represent it by a predicate that holds for the edges.

\[
\text{type synonym} \quad (\text{'q', 'l}) \text{lts} = \"'q \Rightarrow 'l \Rightarrow 'q \Rightarrow bool\"
\]

For an LTS \(\delta\) over nodes of type \(\text{'q}\) and labels of type \(\text{'l}\), the expression \(\delta \ q \ l \ q'\) means that there is an edge from \(q\) to \(q'\) labeled with \(l\).

A word from source node \(u\) to target node \(v\) is the sequence of edge labels one encounters when going from \(u\) to \(v\).

\[
\text{inductive word :: } \"(\text{'q}, \text{'l}) \text{lts} \Rightarrow \text{'q} \Rightarrow \text{'l} \text{ list} \Rightarrow \text{'q} \Rightarrow bool\" \text{ for } \delta
\]

where

\[
\text{empty: } \text{word } \delta \ q \ [\ ] \ q
\]

| prepend: \(\; [\delta \ q \ l \ qh; \text{word } \delta \ qh \ ls \ q'] \Rightarrow \text{word } \delta \ q \ (l \ # \ ls) \ q'\)

A control flow graph is a labeled transition system where the edges are labeled with actions:

\[
\text{datatype action =}
\]

\[\begin{array}{l}
EAssign \ vname \ aexp \ — \text{Assign variable} \\
ETest \ bexp \ — \text{Only executable if expression is true} \\
ESkip
\end{array}
\]

\[
\text{type synonym} \quad 'q \ cfg = \"('q, action) \ lts\"
\]

Intuitively, the control flow graph is executed by following a path and applying the effects of the actions to the state.

Define the effect of an action to a state. Your function must return \(\text{None}\) if the action is not executable, i.e., a test of an expression that evaluates to \(\text{False}\):

\[
\text{fun eff :: } \text{action } \Rightarrow \text{state } \Rightarrow \text{state option}
\]

Lift your definition to paths. Again, only paths where all tests succeed must yield a result \(\neq \text{None}\).

\[
\text{fun eff_list :: } \text{action list } \Rightarrow \text{state } \Rightarrow \text{state option}
\]
The control flow graph of an IMP program can be defined over nodes that are commands. Complete the following definition. (Hint: Have a look at the small-step semantics first.)

\[
\text{inductive } \text{cfg} ::= \text{“com \ cfg” where}
\]
\[
\text{cfg,assign: “cfg } (\text{n ::= e}) \ (EAssign \ n \ e) \ (\text{SKIP})”
\]
\[
| \text{cfg,Seq2: “[ \text{cfg c1 e c1′ ] } \Rightarrow \text{cfg } (\text{c1 ;; c2}) \ e \ (\text{c1′ ;; c2})”}
\]

Prove that the effects of paths in the CFG match the small-step semantics:

\[
\text{lemma } eq\_path: “(c, s) \rightarrow^* (c′, s′) \iff (\exists \pi. \ \text{word } \text{cfg } c \ \pi \ c′ \land \ \text{eff\_list } \pi \ s = \text{Some } s′)”
\]

Hint: Prove the lemma for a single step first.

**Homework 10.1  Germany’s Next Top Formalizer (20 points plus bonus)**

Submission until Monday 20.07.2014 10:00 AM. Please send your submissions to jasmin.blanchette@mpi-inf.mpg.de and mathias.fleury@mpi-inf.mpg.de with “[CONCRETE10]” in the subject of your email. If your name is Mei Myouji, please call your theory file Myouji_Mei.thy.

Think up a nice formalization yourself, for example:

- Prove some interesting result about graph, automata, or formal language theory.
- Formalize some results from mathematics.
- Prove some results from program optimization.

You have two weeks at your disposal for this homework. Bonus points will be awarded for the most impressive solutions.

In case you do not have a good idea, here are some further suggestions:

- Add some new language features to IMP, and redo some proofs (e.g., compiler, typing).
- Compile commands to a register machine and show correctness.
- Prove correct some non-trivial program, e.g., square roots using the bisection method. Hint: A modular approach of writing and proving programs may help, e.g., you may try to reuse a program for multiplication and its correctness proof, rather than inlining the program and the proof.

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 or is intended to do.

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