Semantics Exercises in Isabelle

Basic Isabelle
Using just the function Suc and pattern matching, define a function add that adds two natural numbers. Prove the following

\[
\begin{align*}
\text{add } x \ y &= \text{add } y \ x \\
\text{add } (\text{add } x \ y) \ z &= \text{add } x \ (\text{add } y \ z) \\
\text{add } x \ y &= x + y
\end{align*}
\]

Big and Small Step

1. Define an Expression Compiler (*)
   Define a toy machine language for a stack machine with 3 instructions. The state of the stack machine contains a stack (a list of values), and an IMP state (mapping from vnames to values). The 3 instructions are ADD, which adds the top two values on the stack, LOADI i that puts an integer i on the stack, and Load vn that loads the value of variable vn onto the stack.

   Define a function that executes a single instruction, a function that executes a list of instructions, and a function that compiles an arithmetic expression into an equivalent list of machine instructions.

2. Prove Correctness of the Expression Compiler (*)
   State and prove that your expression compiler from the exercise above is correct: executing a compiled arithmetic expression on the machine should give the same result as the semantics of the arithmetic expression.

3. Extending IMP (*/**)
   Extend the IMP language with the construct REPEAT c UNTIL b. Update the proofs up to equivalence of small step and big step semantics.

Types

1. Alternative typed expression evaluation (*)
   Instead of an inductive definition, write taval in theory IMP/Types_Exp.thy as a function from expression and state to value option. Do the same for boolean expressions. Prove that your definition is equivalent to the inductive definition from the lecture.

2. Extending the Language (**)
   Complete the REPEAT c UNTIL b language extension for the type system proofs.
The Isabelle IMP theories are available in the Isabelle distribution under

```
isabelle/src/HOL/IMP
```

in particular in the theories `AExp`, `Big_Step`, `Small_Step`, and `Types`.

This directory also contains formalisations of many additional semantics concepts. They are explained in the book *Concrete Semantics*, available from

```
http://www.in.tum.de/~nipkow/Concrete-Semantics/
```