Isabelle/HOL Exercises
Projects

Compilation with Side Effects

This exercise deals with the compiler example in Section 3.3 of the Isabelle/HOL tutorial. The simple side effect free expressions are extended with side effects.

1. Read Sections 3.3 and 8.2 of the Isabelle/HOL tutorial. Study the section about `fun_upd` in theory `Fun` of HOL: `fun_upd f x y`, written `f(x:=y)`, is `f` updated at `x` with new value `y`.

2. Extend data type `('a, 'v) expr` with a new alternative `Assign x e` that shall represent an assignment `x = e` of the value of the expression `e` to the variable `x`. The value of an assignment shall be the value of `e`.

3. Modify the evaluation function `value` such that it can deal with assignments. Note that since the evaluation of an expression may now change the environment, it no longer suffices to return only the value from the evaluation of an expression.

   Define a function `se_free :: expr ⇒ bool` that identifies side effect free expressions. Show that `se_free e` implies that evaluation of `e` does not change the environment.

4. Extend data type `('a, 'v) instr` with a new instruction `Store x` that stores the topmost element on the stack in address/variable `x`, without removing it from the stack. Update the machine semantics `exec` accordingly. You will face the same problem as in the extension of `value`.

5. Modify the compiler `comp` and its correctness proof to accommodate the above changes.