Prof. Dr. A. Poetzsch-Heffter
M.Sc. Peter Zeller
Dipl.-Inf. C. Feller

University of Kaiserslautern
Department of Computer Science
Software Technology Group

Exercise Sheet 10: Specification and Verification with Higher-Order Logic (Summer Term 2014)

Exercise 1 Hoare Logics – single procedure

Download the file `Sheet10_hoare_procedure.thy`. It contains a theory where the Imp language has been extended with local variables and one recursive procedure.

a) Consider the following recursive procedure:

```
IF vi ≤ 0 THEN
        SKIP
ELSE
        vi := vi − 1;
        CALL;
        vi := vi + 1
END
```

Prove that the value of i is not changed after calling the procedure:

```
proc, {} ⊢ {λN s. vi▷s = N} CALL {λN s. vi▷s = N}
```

Hint: In the statement above we keep `proc` as a context information left of the $\vdash$ symbol. Furthermore we also keep a set of assumptions about the procedure in the context information. This set is initially empty and for our language with just one procedure it will at most contain one element. The assertions now take an additional parameter (here N), which contains a state which does not change during one procedure call. We use this parameter to link the pre- and the post state.

b) Prove the following example which uses a local variable:

```
p, {} ⊢ {λz s. True}
  vx := 1;
  vy := 2;
  VAR vx := vy + 2  IN
     vy := vx
  END;
  vx := vx + 1
{λz s. vx▷s = 2 ∧ vy▷s = 4}
```

c) (optional) Prove the correctness of the Fibonacci procedure you find in the theory file.

Hint: The procedure takes a number n and stores the result $fib(n)$ in r. First show a stronger property, which states that `vr` contains the result and `vt` and `vx` are not changed after a `CALL`. For the recursive calls first use the rule `hoare_Conseq` and then the rule `hoare_Asm`. In the rule `hoare_Conseq` you should give the precondition `P` explicitly using `rule_tac`.

Exercise 2 Hoare Logics – nondeterminism

Download the file `Sheet10_hoare_choice.thy`. This file contains an extension of the Imp-language with a nondeterministic `Choice` command with the following syntax:

```
IF ? THEN c1 ELSE c2 END
```

When it is executed, it will either execute `c1` or `c2`.

a) Extend the Hoare Logic with a useful and sound rule about the `Choice` command. Adapt the soundness proof if necessary.

b) Prove that after executing the following program, `vx` and `vy` have the same value.

```
IF ? THEN
    vx := 0;
    vy := 0
ELSE
    vx := 1;
    vy := 1
END
```

c) Show that after executing the following program, the value of `vx` is `3`.

```
IF ? THEN
    vx := 0
ELSE
    vx := 3
END;
WHILE vx ≤ 0 DO
        vx := vx − 1
END
```

Exercise 3 Hoare Logics – termination

The statement in exercise 2 b) was only valid, because we considered only partial correctness. We now extend the Hoare Logic for our non-deterministic language from exercise 2 to total correctness: The statement $\models_t \{P\}c\{Q\}$ holds true, if $\models \{P\}c\{Q\}$ and if c always terminates, when started in a state which satisfies P. We formalize the statement that a command c always terminates when started in a state s with the predicate $c \downarrow s$.

You can find the theory for this exercise in `Sheet10_hoare_termination.thy`.

a) Transfer your proof about the sum-program from exercise e) on Sheet 9 to a total correctness proof.

Hint: When using the rule `hoare_While` you now have to give a well-founded relation on states r in addition to the invariant. The easiest way to define a well-founded relation is to use the function `measure`. This function takes a function m of type `'a ⇒ nat` and returns the following relation: $\{(x, y).\ m(x) < m(y)\}$.

b) Show the following fact about nontermination of while loops:
$[\![Inv\ s; \vdash \{Inv\}\, c\, \{Inv\}; \forall s.\ Inv\ s \longrightarrow \text{beval}\ s\ b]\!] \Longrightarrow \neg(\text{WHILE b DO c END} \downarrow s)$

c) Consider the definition of $c \downarrow s$ in the given theory file and this alternative definition:
$(c \downarrow s) \leftrightarrow (\exists s'.\ \langle c, s\rangle \rightarrow s')$
Show in Isabelle that the two definitions are not equivalent.

d) (optional) Transfer your proof about the binary-search program from exercise l) on Sheet 9 to a total correctness proof.