Technische Universität München
Institut für Informatik
Prof. Tobias Nipkow, Ph.D.
Andrei Popescu, Peter Lammich

WS 2012/13
15. 1. 2013

Semantics of Programming Languages

Exercise Sheet 12

Homework 12 Verification Condition Generator for Total Correctness

Submission until Tuesday, 22. 1. 2013, 10:00am.

In this homework, your task is to implement a verification condition generator for total correctness, which is also optimized for handling automatically the termination of certain FOR-like while loops.

We start by defining the datatype of total-correctness annotated programs. Notice, compared to the partial correctness case, the extra measure argument of *Awhile*, of type *state* ⇒ *nat*, aimed at handling termination.

```
datatype acom =
  ASKIP |
  Aassign vname aexp    ("(_ ::= _)" [1000, 61] 61) |
  Aseq   acom acom      ("_;/ _"  [60, 61] 60) |
  Aif bexp acom acom    ("(IF _/ THEN _/ ELSE _)"  [0, 0, 61] 61) |
  Awhile assn "state ⇒ nat" bexp acom  ("({_,_}/ WHILE _/ DO _)"  [0, 0, 61] 61)
```

The types of both commands and annotated commands are made instances of the *vars* class by defining suitable operators (see the homework template).

```
instantiation com :: vars
instantiation acom :: vars
```

Recall from homework 6 the following facts about the interaction betwee evaluation/execution and *vars*:

```
lemma aval_vars: "⟦s1 = s2 on X; vars a ⊆ X⟧ ⟹ aval a s1 = aval a s2"
lemma confinement: "(c,s) ⇒ t ⟹ s = t on (UNIV − vars c)"
```

1. Write a function for identifying certain annotated while loops trivially well-behaved w.r.t. termination, which we call "FOR loops". Namely, a FOR loop is an annotated command of the form *Awhile I M* (*Less* (*V x*) *a*) (*c* ; *x* ::= (*Plus* (*V x*) (*N 1*))) where *x* does not appear in *a* or *c* and the sets of variables of *a* and *c* are disjoint. FOR loops should be identified via a function *isF*, where *isF b d* tests if *Awhile I M b d* is a FOR loop:

fun isF :: *"bexp* $\Rightarrow$ *acom* $\Rightarrow$ *bool"* **where**

isF should be executable—some tests are found in the template.

2. Define a verification condition generator vc for total correctness. The "precondition" function, pre similar to that from the partial-correctness case, is given in the template:

fun pre :: *"acom* $\Rightarrow$ *assn* $\Rightarrow$ *assn"* **where**

The recursive clauses for vc are essentially the ones from the partial-correctness case, except for the case of WHILE loops, where you need to take two further aspects into account:

- incorporate the measure annotation M in the generated conditions (hint: by contrast to the partial-correctness case, use $pre\ c\ (\lambda s'.\ I\ s' \wedge M\ s' < M\ s)\ s$ and $vc\ c\ (\lambda s'.\ I\ s' \wedge M\ s' < M\ s)$ instead of $pre\ c\ I$ and $vc\ c\ I$);
- the above only if the WHILE is not a FOR loop—otherwise, M should be ignored.

fun vc :: *"acom* $\Rightarrow$ *assn* $\Rightarrow$ *bool"* **where**

Note that, unlike for partial correctness, here vc has type $acom \Rightarrow assn \Rightarrow bool$ instead of $acom \Rightarrow assn \Rightarrow assn$. Here, $vc\ c\ Q$ should play a similar role as $\forall\ s.\ vc\ c\ Q\ s$ from partial correctness.

Some tests for your definition of vc are given in the template.

3. Define a function that strips away annotations:

fun $strip$ :: *"acom* $\Rightarrow$ *com"* **where**

4. Prove the following facts about your operators (analogous to the partial-correctness case), culminating with soundness. For the theorem *vc_sound*, in the WHILE case, you will need to distinguuish between FOR loops and non FOR loops, and provide a suitable measure in the case of the former. (Recall that the verification condition generator should ignore the measure annotation at FOR loops.)

lemma *pre_mono*: "$\forall s.\ P\ s \longrightarrow P'\ s \Longrightarrow pre\ c\ P\ s \Longrightarrow pre\ c\ P'\ s$"

lemma *vc_mono*: "$\forall s.\ P\ s \longrightarrow P'\ s \Longrightarrow vc\ c\ P \Longrightarrow vc\ c\ P'$"

lemma *vc_sound*: "$vc\ c\ Q \Longrightarrow \vdash_t \{pre\ c\ Q\}\ strip\ c\ \{Q\}$"

corollary *vc_sound'*: "$vc\ c\ Q \wedge (\forall s.\ P\ s \longrightarrow pre\ c\ Q\ s) \Longrightarrow \vdash_t \{P\}\ strip\ c\ \{Q\}$"