# Exam Program Verification 2016/2017 (SAMPLE VERSION)

22nd Dec 2016, 15:15–17:15

Lecturer: Wishnu Prasetya

1. **Program Semantic** [1.5 pt].

   Consider a simple programming language $E^+$ with the following syntax:

   $$\begin{array}{lll} Program & \rightarrow & \textbf{vars}\ declarations\ ;\ assignments \\ declarations & \rightarrow & \text{one or more declaration separated by ";"} \\ declaration & \rightarrow & identifier = expression \\ assignments & \rightarrow & \text{one or more assignment separated by ";"} \\ assignment & \rightarrow & identifier := expression \\ expression & \rightarrow & \quad \text{numeric constants like 0,1,2,...} \\ & & \mid\ identifier \\ & & \mid\ identifier^{--} \\ & & \mid\ identifier^{++} \\ & & \mid\ expression + expression \end{array}$$

   The meaning of the above constructs, except for $x^{--}$ and $x^{++}$, is as usual. For example:

   $$\textbf{vars}\ x{=}1\ ;\ y{=}x\ ;\quad y := y + 1;\ x := y$$

   is a program that creates the variables $x$ and $y$, both initialized with the value 1. The program then does the specified sequence of assignments, and ends up with the value of $x$ and $y$ both equal to 2.

   Expressions like $x^{--}$ and $x^{++}$ have side effect. For example, if $x$ and $y$ both are currently 1, then executing:

   $$z \ := \ x^{--} \ + \ y^{++}$$

   first decreases the value of $x$ (so now $x$ is 0), then adds $y$ to it to calculate the sum (so the sum is 1). After evaluating $y$, its value is increased (so now $y$ is 2). So the above assignments results in the value of $x, y, z$ to become respectively $0, 2, 1$.

   In general, the evaluation of the expression $x^{--}$ proceeds by first decreasing the value of $x$ by one, then we return this value as the result of the evaluation. Whereas the evaluation of the expression $x^{++}$ proceeds by first remembering the current value of $x$, say $x_0$, then we increase $x$ by 1, then we return $x_0$ as the result of the evaluation.

   (a) *Provide an operational semantic* for the above programming language. You can choose whether you want to provide a small step or a big step semantic.

   **Answer:** Note: the above explained semantic of $x^{--}$ is actually that of ${}^{--}x$. We will approve both versions then.

   I'll only show the semantic of expressions:

   - $(c, s) \rightarrow (c, s)$

- $(x, s) \rightarrow (s\ x, s)$
- $$\frac{(e_1, s) \rightarrow (v_1, s_1) \quad , \quad (e_2, s_1) \rightarrow (v_2, s_2)}{(e_1 + e_2, s) \rightarrow (v_1 + v_2, stmt_2)}$$
- $(x{-}{-}, s) \ \rightarrow \ (s\ x - 1\ ,\ \textsf{update}\ s\ x\ (s\ x - 1))$
- $(x{+}{+}, s) \ \rightarrow \ (s\ x\ ,\ \textsf{update}\ s\ x\ (s\ x + 1))$
- The semantic of one or more assignments:
$$\frac{(e, s) \rightarrow (v, t)}{(x := e, s) \ \rightarrow \ \textsf{update}\ t\ x\ v}$$
$$\frac{(x := e, s) \rightarrow t \quad , \quad (rest, t) \rightarrow u}{(x := e; rest, s) \ \rightarrow \ u}$$
- The semantic of one or more declarations:
$$\frac{(e, s) \rightarrow (v, t)}{(x = e, s) \ \rightarrow \ (x, v) : t}$$
$$\frac{(x = e, s) \rightarrow t \quad , \quad (rest, t) \rightarrow u}{(x = e; rest, s) \ \rightarrow \ u}$$
- The semantic of programs:
$$\frac{(decls, []) \rightarrow t \quad , \quad (asgs, t) \rightarrow u}{(\textbf{vars}\ decls\ ;\ asgs\ ,\ []) \ \rightarrow \ u}$$

(b) Propose a definition of Hoare triple $\{P\}\ S\ \{Q\}$ in terms of the semantic you define above. Here $S$ is a series of assignments from $L^+$. $P$ and $Q$ are predicates which can be evaluated on a state. You can assume that there is a function $eval(P, s)$ that evaluates whether $P$ holds on the state $s$.

**Answer:**

$$\{P\}\ S\ \{Q\} \quad = \quad \text{for all well-formed states } s, t\text{: } eval(P, s) \text{ and } (S, s) \rightarrow t \text{ implies } eval(Q, t)$$

A state is "well-formed" with respect to the above triple if all free variables mentioned there are defined in the state.

2. **Loop Invariant** [1.5 pt].

Give an invariant for each of the GCL loops below. It should be an invariant that is consistent, strong enough to realize the asked post-condition, and realistic to be established by the pre-condition or initialization of the loop. Use the partial correctness interpretation of Hoare triples.

Below, `a` is an infinite array of `int`; `b` is of type `bool`; other variables are of type `int`.

(a) $\{\ \texttt{x} = \texttt{10}\ \}$ **while** $\texttt{x}{>}\texttt{0}$ **do** $\{\texttt{x} := \texttt{x}{-}\texttt{1}\ \}$ $\{\ \texttt{x}{=}\texttt{0}\ \}$
**Answer:** $x \geq 0$

(b) $\{\ \texttt{x}{=}\texttt{10} \wedge \texttt{y}{=}\texttt{0}\ \}$ **while** $\texttt{x}{>}\texttt{0}$ **do** $\{\texttt{x} := \texttt{x}{-}\texttt{1}\ ;\ \texttt{y} := \texttt{y}{+}\texttt{1}\ \}$ $\{\ \texttt{x}{=}\texttt{0} \wedge \texttt{y}{=}\texttt{10}\ \}$
**Answer:** $x{\geq}0 \wedge x{+}y{=}10$

(c) $\{\ \texttt{x}{=}\texttt{10} \wedge \texttt{y}{=}\texttt{1}\ \}$ **while** $\texttt{x}{>}\texttt{y}$ **do** $\{\texttt{y} := \texttt{y}{+}\texttt{2}\ \}$ $\{\ \texttt{y}{=}\texttt{11}\ \}$
**Answer:** $x{=}10 \wedge y{\leq}11 \wedge \textsf{odd}\ y$

(d) $\{\ (\exists \mathtt{k} : 0 \leq \mathtt{k} < 10 : \mathtt{a}[\mathtt{k}] < 0)\ \}$

$\mathtt{k}, \mathtt{found} := 0, (\mathtt{a}[0] < 0)\ ;$
**while** $\neg\mathtt{found}$ **do** $\{$**var** $\mathtt{i}\ ;\ \mathtt{k} := \mathtt{i}\ ;\ \mathtt{found} := 0 \leq \mathtt{i} < 10 \wedge \mathtt{a}[\mathtt{i}] < 0\ \}$

$\{\ \mathtt{a}[\mathtt{k}] < 0\ \}$

Note that a new variable declared in a **var**-block is uninitialized (it takes an arbitrary value, but of the right type).
**Answer:** $\mathit{found} = (0 \leq k < 10\ \wedge\ a[k] < 0)$

(e) $\{$ **true** $\}$

$\mathtt{b}, \mathtt{i}\ :=$ **true**, $1\ ;$
**while** $\mathtt{i} < 10 \wedge \mathtt{b}$ **do** $\{$
  $--$check if $a[i]$ is equal to $a[i-1]$
  $\mathtt{b} := (\mathtt{a}[\mathtt{i}] = \mathtt{a}[\mathtt{i}-1])\ ;\ \mathtt{i} := \mathtt{i}+1$
$\}$

$\{\ \mathtt{b} = (\forall \mathtt{k} : 0 \leq \mathtt{k} < 10 : \mathtt{a}[\mathtt{k}] = \mathtt{a}[0])\ \}$

**Answer:**
$1 \leq i \leq 10 \wedge b = (\forall k : 0 \leq k < i : a[k] = a[0])$

3. **Weakest pre-condition** [1.5 pt].

(a) Consider a new statement construct for GCL: $([]k : 0 \leq k < n : \mathit{stmt}_k)$, where $k$ can be assumed to be a fresh variable, $n$ is an existing variable, and $\mathit{stmt}_k$ is a statement which may use $k$. Example:

$$([]k : 0 \leq k < n : \textbf{if } a[k] > 0 \textbf{ then } a[k] := a[k]-1 \textbf{ else skip})$$

The construct non-deterministically chooses one of the $\mathit{stmt}_k$ and executes it.
Propose a definition of the $\mathsf{wlp}$ of such a construct.
**Answer:**

$$\mathsf{wlp}\ ([]k : 0 \leq k < n : \mathit{stmt}_k)\ Q \quad = \quad (\forall k : k < n : \mathsf{wlp}\ \mathit{stmt}_k\ Q)$$

(b) Give the definition of **refby** and propose a definition of the $\mathsf{wlp}$ of assignments that target a two dimensional array.
**Answer:**

$$a(i, j\ \mathbf{refby_2}\ e) \quad = \quad a(i\ \mathbf{refby_1}\ (a[i](j\ \mathbf{refby_1}\ e)))$$

So, $a(i, j\ \mathbf{refby_2}\ e)[i][j] \ = \ a(i\ \mathbf{refby_1}\ (a[i](j\ \mathbf{refby_1}\ e)))[i][j]$, which is equal to:

$$(a[i](j\ \mathbf{refby_1}\ e))[j]$$

which is equal to $e$. If we assume $j \neq 0$, then notice that $a(i, j\ \mathbf{refby_2}\ e)[i][0]$ by applying the same unfolding is:

$$(a[i](j\ \mathbf{refby_1}\ e))[0]$$

which is then equal to $a[i][0]$.

(c) Describe a procedure to calculate the $\mathsf{wlp}$ of a `while`-loop through a fix-point iteration.
**Answer:** We start with $I_0 = \mathit{true}$. Then $I_{k+1}$ is calculated as:

$$I_{k+1} \ = \ (g \wedge \mathsf{wlp}\ S\ I_k) \vee (\neg g \wedge Q)$$

If this process ends in a fix point we have a solution.

4. **Basic HOL** [1 pt].

   (a) `DISCH` is a rule of the type `term` $\rightarrow$ `thm` $\rightarrow$ `thm`. If $t$ is a member of the assumptions of a theorem $A \vdash u$, `DISCH` $t$ will do the following:

   $$\frac{A \vdash u}{A{-}t \ \vdash \ t \Rightarrow u}\ \texttt{DISCH}\ t$$

   where $A{-}t$ means all the assumptions in $A$, but without $t$.

   In HOL, a tactic is a function of the type:

   $$goal \rightarrow (goal\ \texttt{list}\ \#\ proofFunction)$$

   where $goal = (\texttt{term list}\ \#\ \texttt{term})$ and $proofFunction = \texttt{thm list} \rightarrow \texttt{thm}$.

   Show how this works by demonstrating how the tactic `DISCH_TAC` can be constructed from the `DISCH` rule.

   **Answer:** The code below is pseudo (not real ML):

   $$\begin{array}{l} DISCH\_TAC\ (B\ ?{-}\ t \Rightarrow u)\ \ = \\ \quad \textbf{let}\ pf\ thms = DISCH\ t\ thms_0 \\ \quad \textbf{in}\ ([B + \{t\}\ ?{-}\ u]\ ,\ pf) \end{array}$$

   (b) Show how the quantifiers $\forall$ and $\exists$ are defined in the primitive HOL. If you use operators other than function application, $\lambda$, $=$, $\Rightarrow$, and `T` define your operators as well.

   **Answer:**

   $$\forall P \ = \ (P = (\lambda x.\, T))$$

   $$\exists P \ = \ P(@P)$$

   where @ is defined through an axiom, namely, for all $P, x$, $P\ x \Rightarrow P(@P)$

5. **Program Semantic** [0.5 pt, challenging].

   Consider again the language $E^+$ in the question No. 1. *Propose a definition* of $\mathsf{wlp}\ (x := e)\ Q$ for this language. Keep in mind that expressions in $E^+$ may have side effect. We want to have a sound and complete $\mathsf{wlp}$. That is, it should satisfy:

   $$\{P\}\ x := e\ \{Q\} \quad \equiv \quad P \Rightarrow \mathsf{wlp}\ (x := e)\ Q$$

   You can assume that all variables in $e$ and $Q$ are defined/declared.

   *Motivate* why you think that your proposal is sound and complete.

   **Answer:**

   We first define a transformation. The variables $z$, $z_i$ are assumed to be fresh.

   - $T(c, z) \ = \ z := c$
   - $T(x, z) \ = \ z := x$
   - $T(x{-}{-}, z) \ = \ x := x{-}1\ ;\ z := x$
   - $T(x{+}{+}, z) \ = \ z := x\ ;\ z := x{+}1$
   - $T(e_1 + e + 2, z) \ = \ T(e_1, z_1)\ ;\ T(e_2, z_2)\ ;\ z := z_1 + z_2$

   Then we can define $\mathsf{wlp}\ (x := e)$ as $\mathsf{wlp}\ T(x := e)$. To prove the soundness and completeness, we first need to define the meaning of Hoare triple with

6. **HOL** [4 subquestions for total 4 pt, time: 48 hrs].

From the PV website, you can download the file xxx.smx. This is basically the same as in the HOL-tutorial.

It contains the following parts:

**Section 1** defines an embedding of a subset of GCL in HOL. It also contains an example of how a simple GCL program is expressed in HOL.

**Section 2** defines the semantic of GCL constructs, the semantic of Hoare triple, and provides a definition of wlp.

**Section 3** provides the proofs of some basic laws of Hoare logic, for example these:

**pre-condition strengthening:**

$$\frac{\{q\}\ stmt\ \{r\} \quad , \quad p \Rightarrow q}{\{p\}\ stmt\ \{r\}}$$

**post-condition weakening:**

$$\frac{\{p\}\ stmt\ \{q\} \quad , \quad q \Rightarrow r}{\{p\}\ stmt\ \{r\}}$$

**Section 4** proves the soundness the wlp defined in Section 3. 'Sound' here means that any final state that results from executing a GCL statement *stmt* from any state in the pre-condition produced by wlp *stmt* $q$ will satisfy $q$. In other words, the following Hoare triple is always valid:

$$\{\ \mathsf{wlp}\ stmt\ q\ \}\quad stmt \quad \{\ q\ \}$$

for any GCL statement *stmt*.

**Section 5** shows how to prove the correctness of the example from Section 1, with respect to some post-condition.

The problems that you have to solve are listed below (REMOVED in this version). Send your solution in the form of a modified script.