Software Testing & Verification 2013/2014
Universiteit Utrecht

2nd Jul. 2014, 13:30 - 16:30, BBL 001

Lecturer: Wishnu Prasetya

You are allowed to bring along the Appendix of the LN.

Part I [3pt (6×0.5)]

For each question, choose *one* correct answer.

1. What is the **weakest** pre-condition of the following statement with respect to the given post-condition?

   $\{*\ ?\ *\}$ `x := x+y ; y := x+3` $\{*\ xy = 0; *\}$

   (a) $x^2 + y^2 + 2xy + 3x + 3y = 0$
   (b) $2x^2 + 9x + 9 = 0$
   (c) $(x+y)(x+3) = 0$
   (d) $(x = 0) \wedge (y = 0)$

   Answer: Just do the substitution (in the correct order). If you the simplify the result, the answer is (a).

2. What is the **weakest** pre-condition of the following statement with respect to the given post-condition?

   $\{*\ ?\ *\}$ $a[0] := a[0] - a[k]$ $\{*\ a[k]{=}0\ *\}$

   (a) $a[k] = 0$
   (b) $k = 0$
   (c) $(k{=}0 \rightarrow a[0]{-}a[k] \mid a[k]) = 0$
   (d) $a(0\ \textbf{repby}\ (a\ \textbf{repby}\ 0) - (a\ \textbf{repby}\ k))[k] = 0$

   Answer: c

3. Consider the following program to search for a prime number between a and b. It's body is not fully shown: *body* below is some statement, and e is some expression. The parameter a is passed by value, and b by copy-restore. The *body* is known to modify a and b.

$$\texttt{find}(\texttt{a} : \texttt{int},\ \texttt{OUT}\ \texttt{b} : \texttt{int})\ :\ \texttt{bool}\ \{\ \textit{body}\ ;\ \textbf{return}\ e\ \}$$

Here is the specification of the program:

$$\{*\ \texttt{0<a}\le\texttt{b}\ *\}$$

$$\texttt{B}_0 := \texttt{b}\ ;\ \texttt{find}(\texttt{a}, \texttt{OUT}\ \texttt{b})$$

$$\{*\ (\texttt{return} = (\exists \texttt{x} : \texttt{a}\le\texttt{x}<\texttt{B}_0 : \texttt{isPrime(x)}))\ \wedge\ (\texttt{return} \Rightarrow \texttt{isPrime(b)})\ *\}$$

Which of the following specifications is a correct reduction of the above specification to the corresponding specification of the program's body?

(a)

$$\{*\ \texttt{0<a}\le\texttt{b}*\}$$

$$\textit{body}\ ;\ \texttt{return} := e$$

$$\{*\ (\texttt{return} = (\exists \texttt{x} : \boxed{a}\le\texttt{x}<\boxed{b} : \texttt{isPrime(x)}))\ \wedge\ (\texttt{return} \Rightarrow \texttt{isPrime(}\boxed{b}\texttt{)})\ *\}$$

(b)

$$\{*\ \texttt{0<a}\le\texttt{b}*\}$$

$$\texttt{B}_0 := \texttt{b}\ ;\ \textit{body}\ ;\ \texttt{return} := e$$

$$\{*\ (\texttt{return} = (\exists \texttt{x} : \boxed{a}\le\texttt{x}<\boxed{B_0} : \texttt{isPrime(x)}))\ \wedge\ (\texttt{return} \Rightarrow \texttt{isPrime(}\boxed{B_0}\texttt{)})\ *\}$$

(c)

$$\{*\ \texttt{0<a}\le\texttt{b}*\}$$

$$\texttt{A}_0, \texttt{B}_0 := \texttt{a}, \texttt{b}\ ;\ \textit{body}\ ;\ \texttt{return} := e$$

$$\{*\ (\texttt{return} = (\exists \texttt{x} : \boxed{A_0}\le\texttt{x}<\boxed{B_0} : \texttt{isPrime(x)}))\ \wedge\ (\texttt{return} \Rightarrow \texttt{isPrime(}\boxed{b}\texttt{)})\ *\}$$

(d)

$$\{*\ \texttt{0<a}\le\texttt{b}*\}$$

$$\texttt{A}_0, \texttt{B}_0 := \texttt{a}, \texttt{b}\ ;\ \textit{body}\ ;\ \texttt{return} := e$$

$$\{*\ (\texttt{return} = (\exists \texttt{x} : \boxed{A_0}\le\texttt{x}<\boxed{b} : \texttt{isPrime(x)}))\ \wedge\ (\texttt{return} \Rightarrow \texttt{isPrime(}\boxed{b}\texttt{)})\ *\}$$

Answer: c

4. Which of the following proofs is correct (according the the proof system of the LN)? Read the steps carefully.

   (a) PROOF
   [A1:] $(\forall x :: P\ x)$
   [A2:] $Q\ x$
   [G:] $(\forall x :: P\ x \wedge Q\ x)$
   1. { $\forall$-elimination on A1 } $P\ x$
   2. { conjunction of 1 and A2 } $P\ x \wedge Q\ x$
   3. { $\forall$-introduction on 2 } $(\forall x :: P\ x \wedge Q\ x)$
   END

   (b) PROOF
   [A1:] $a = b$
   [G:] $a \vee (\exists k :: x[k]) \ = \ b \vee (\exists k :: x[k])$
   1. { $\vee$-introduction } $a \vee (\exists k :: x[k])$
   2. { $\vee$-introduction } $b \vee (\exists k :: x[k])$
   3. { combining 1 and 2 } $a \vee (\exists k :: x[k]) \ = \ b \vee (\exists k :: x[k])$
   END

   (c) PROOF
   [A1:] $(\exists x :: P\ x)$
   [A2:] $Q\ a$
   [G:] $(\exists a :: P\ a \wedge Q\ a)$
   1. { $\exists$-elimination on A1 } $P\ a$
   2. { conjunction of 1 and A2 } $P\ a \wedge Q\ a$
   3. { $\exists$-introduction on 2 } $(\exists a :: P\ a \wedge Q\ a)$
   END

   (d) PROOF
   [A1:] $\neg(\exists x :: P\ x)$
   [A2:] $P\ a$
   [G:] false
   1. { $\exists$-introduction on A2 } $(\exists a :: P\ a)$
   2. { contradiction between A1 and 1 } false
   END

   Answer: d

5. A statement S satisfies the following specifications:

   (a) $\{* \ P \ *\} \ \ S \ \ \{* \ Q_1 \ *\}$
   (b) $\{* \ Q_2 \ *\} \ \ S \ \ \{* \ R \ *\}$, where $Q_2 \Rightarrow Q_1$ (note the direction!)

   Which of the folowing specifications is a valid consequence of (a) and (b) above ?

   (a) $\{* \ P \ *\} \ \ S; S \ \ \{* \ R \ *\}$
   (b) $\{* \ P \wedge Q_2 \ *\} \ \ S \ \ \{* \ Q_1 \wedge R \ *\}$
   (c) $\{* \ P \vee Q_2 \ *\} \ \ S \ \ \{* \ Q_1 \wedge R \ *\}$
   (d) $\{* \ P \ *\} \ \ S; S \ \ \{* \ Q_2 \Rightarrow R \ *\}$

Answer: b

6. Consider the loop below; `x` is of type `int` and `even(x)` is a side-effect-free function that checks if x is an even integer.

$\{*\ \mathtt{1 < x < N}\ *\}$

`while x<N do { if even(x) then x := 2 * x  else x := x − 1 }`

$\{*\ \mathtt{even(x)}\ *\}$

Which of the predicates below is a correct invariant of the loop, that is enough to prove that the above specification is valid, under the partial correctness interpretation?

(a) $\mathtt{1{<}x} \wedge (\exists \mathtt{x} :: \mathtt{even(x)})$
(b) $(\mathtt{even(x)} \Rightarrow \mathtt{even(2x)})\ \wedge\ (\neg\mathtt{even(x)} \Rightarrow \mathtt{even(x-1)})$
(c) $\mathtt{1{<}x{\leq}N} \wedge \mathtt{even(x)}$
(d) $\mathtt{x{\geq}\ N} \Rightarrow \mathtt{even(x)}$

Answer: a and b won't imply the post condition; c is not implied by the pre-condition. The candidate left is d. It implies the post-condition, and can be established by the pre-condition. Calculating its `wp` over the loop's body gives:

$$(even(x) \Rightarrow (2x \geq N \Rightarrow even(2x)))\ \wedge\ (\neg even(x) \Rightarrow (x{-}1 \geq N \Rightarrow even(x-1)))$$

The first conjust is valid due to $even(2x)$. The second conjunct is implied by the guard $x < N$, which implies that $x{-}1 \geq N$ cannot be true.
So, the answer is indeed (d).

Part II [7pt]

When asked to write a formal proof you need to produce one that is readable, augmented with sufficient comments to explain and convincingly defend your steps. An incomprehensible solution may lose all points.

1. [1.5 pt] **Termination**

Consider again this program, with the same pre-condition:

$\{*\ \mathtt{1 < x < N}\ *\}$

`while x<N do { if even(x) then x := 2 * x  else x := x − 1 }`

Use the Loop Reduction Rule (the inference rule for loop as discussed in the lectures) to prove that this program terminates when executed on the given pre-condition. You only need to prove termination; we do not care in which state the program would terminate.

Answer: Using $I : 1 < x$ and $m = even(x) \rightarrow N{-}x \mid N{+}2$. Implicitly $N > 1$ is part of the invariant, since the loop does not change the value of N.

If the guard $x < N$ is true, $N{-}x$ as well as $N{+}2$ are > 0. So, m has a lower bound.

To show that m decreases, we calculate the weakest pre-condition of:

$$1 < x \wedge x < N \;\Rightarrow\; \texttt{wp}\ (C := m\ ;\ body)\ (m < C)$$

Calculating the wp gives:

$$\begin{array}{l} (even(x) \Rightarrow (even(2x) \to N{-}2x \mid N{+}2) < (even(x) \to N{-}x \mid N{+}2)) \\ \wedge \\ (\neg even(x) \Rightarrow (even(x{-}1) \to N{-}2x{+}2 \mid N{+}2) < (even(x) \to N{-}x \mid N{+}2)) \end{array}$$

This can be simplified to:

$$(even(x) \Rightarrow N{-}2x < N{-}x) \quad \wedge \quad (\neg even(x) \Rightarrow N{-}2x{+}2 < N{+}2)$$

Both conjuncts are implied by the pre-conditon $1 < x$.

It remains to be proven tha the invariant $i < x$ can be maintained. Calculating the wp gives us:

$$(even(x) \Rightarrow 1 < 2x) \quad \wedge \quad (\neg even(x) \Rightarrow 1 < x{-}1)$$

The left conjunct is obviously impied by $I : 1 < x$. For the second conjunct, notice that if x is odd, and $1 < x$, then $x \geq 3$. So, $x{-}1 > 1$.

Done.

2. [3 pt] **Loop**

Here is a program to check if all elements of an array a[0..N) are the same.

```
{*  N > 0  *}     // pre-condition

   i := 1 ;
   uniform:= true ;
   while i<N do {
      uniform := uniform ∧ (a[i]=a[0]) ;
      i := i+1
   } ;

{*  uniform  =  (∀k : 0≤k<N : a[k] = a[0])  *}     // post-condition
```

Give a formal proof that the program is correct. You can skip the termination proof.

Answer: Grading: 0.5 pt Pinit, 2 pt PIC, 0.5 pt EC. No separate point for the invariant; wrong invariant will show up in the proofs anyway.

Bad proof styles (BPS): deduction up to 1 pt.

Incorrect inv: 0.3 pt to maximum.

We'll this this invariant I:

$$1{\leq}i{\leq}N \;\wedge\; (u = (\forall k : 0{\leq}k{<}i : a[k] = a[0]))$$

Here, I will only give a **sketch** of the proof. PEC is quite trivial. For Pinit, we have to prove:

$$N > 0 \;\Rightarrow\; \texttt{wp}\ (i := 1\ ;\ u := true)\ I$$

Calculating the wp gives:

$$1 \leq 1 \leq N \;\wedge\; (true = (\forall k : 0 \leq k < 1 : a[k] = a[0]))$$

which is not difficult to prove.

For PIC we have to prove:

$$I \wedge i < N \;\Rightarrow\; \mathsf{wp}\ (u := u \wedge (a[i] = a[0])\ ;\ i := i+1)\ I$$

Calculating the wp gives:

$$1 \leq i+1 \leq N \;\wedge\; (u \wedge (a[i] = a[0]) = (\forall k : 0 \leq k < i+1 : a[k] = a[0]))$$

The first conjuct is not difficult to prove. For the second conjunct:

PROOF EQUATIONAL
$(\forall k : 0 \leq k < i+1 : a[k] = a[0])$
= { domain merge, $0 \leq i$ }
$(\forall k : 0 \leq k < i \vee (k = i) : a[k] = a[0])$
= { domain split }
$(\forall k : 0 \leq k < i : a[k] = a[0]) \;\wedge\; (\forall k : k = i : a[k] = a[0])$
= { I }
$u \;\wedge\; (\forall k : k = i : a[k] = a[0])$
= { quatification over singleton }
$u \;\wedge\; (a[i] = a[0])$
END

3. [1.5 pt] **Adding a break**

The program from No. 2 can be improved by letting the loop to break when $a[i-1] \neq a[0]$:

{* N > 0 *} // pre-condition

```
i := 1 ;
uniform:= true ;
while i<N ∧ [a[i-1]=a[0]] do {
   uniform := uniform ∧ (a[i]=a[0]) ;
   i := i+1
} ;
```

{* uniform = $(\forall k : 0 \leq k < N : a[k] = a[0])$ *} // post-condition

Give a new formal proof of the loop Exit Condition, that will prove that using the same invariant as in No. 2, the version above will also terminate in the specified post-condition above.

(You only need to give a new PEC proof)

Answer:

PROOF PEC

[A1:] $u = (\forall k : 0 \leq k < i : a[k] = a[0])$

[A2:] $1 \leq i \leq N$

[A4:] $\mathtt{i} \geq \mathtt{N} \vee \mathtt{a}[\mathtt{i}-1] \neq \mathtt{a}[0]$

[G:] $\mathtt{u} = (\forall \mathtt{k} : 0 \leq \mathtt{k} < \mathtt{N} : \mathtt{a}[\mathtt{k}] = \mathtt{a}[0])$

1. { this was already proven in No. 2 } $\mathtt{i} \geq \mathtt{N} \Rightarrow \mathtt{G}$
2. { see subproof below } $(\mathtt{a}[\mathtt{i}-1] \neq \mathtt{a}[0]) \Rightarrow \mathtt{G}$

    PROOF sub

    [A1:] $\mathtt{a}[\mathtt{i}-1] \neq \mathtt{a}[0]$

    [G:] same as PEC.G

    1. { follows from PEC.A2 } $0 \leq \mathtt{i}{-}1 < \mathtt{i}$
    2. { follows from PEC.A2 } $0 \leq \mathtt{i}{-}1 < \mathtt{N}$
    3. { $\exists$-intro on 1 and A1 } $(\exists \mathtt{k} : 0 \leq \mathtt{k} < \mathtt{i} : \mathtt{a}[\mathtt{k}] \neq \mathtt{a}[0])$
    4. { $\exists$-intro on 2 and A1 } $(\exists \mathtt{k} : 0 \leq \mathtt{k} < \mathtt{N} : \mathtt{a}[\mathtt{k}] \neq \mathtt{a}[0])$
    5. { negation of $\forall$ on 3 } $\neg(\forall \mathtt{k} : 0 \leq \mathtt{k} < \mathtt{i} : \mathtt{a}[\mathtt{k}] = \mathtt{a}[0])$
    6. { negation of $\forall$ on 4 } $\neg(\forall \mathtt{k} : 0 \leq \mathtt{k} < \mathtt{N} : \mathtt{a}[\mathtt{k}] = \mathtt{a}[0])$
    7. { rewrite 5 with PEC.A1 } $\neg \mathtt{u}$
    8. { 5 and 7 } $\neg \mathtt{u} = \neg(\forall \mathtt{k} : 0 \leq \mathtt{k} < \mathtt{N} : \mathtt{a}[\mathtt{k}] = \mathtt{a}[0])$
    9. { follows form 8 } $\mathtt{u} = (\forall \mathtt{k} : 0 \leq \mathtt{k} < \mathtt{N} : \mathtt{a}[\mathtt{k}] = \mathtt{a}[0])$

    END

END

4. [1 pt] **Program call**

   Consider the following specification of the program P:

   $$\{*\ \mathtt{y}{>}0\ *\}\ \ \mathtt{Y} := \mathtt{y};\ \mathtt{P}(\mathtt{x{:}int}, \mathtt{OUT}\ \mathtt{y{:}int})\ \ \{*\ (\mathtt{return}{+}\mathtt{y})/\mathtt{Y} > \mathtt{x}\ *\}$$

   Consider this call to P:

   $$\{*\ \mathtt{k}{>}0\ *\}\ \ \mathtt{r} := \mathtt{P}(\mathtt{k}{-}2, \mathtt{k})\ \ \{*\ \mathtt{r}{+}\mathtt{k}{>}0\ *\}$$

   To prove the correctness of the call, we first transform the call to the following equivalent statement:

   ```
   {* k>0 *}
   {* (1) ? *}  @x := k-2 ;
   {* (2) ? *}  @y := k ;
   {* (3) ? *}  r := P(@x,@y) ;
   {* (4) ? *}  k := @y ;
   {* r+k>0 *}
   ```

   (a) Fill in the intermediate predicates (1)..(4) above. Calculate them using the weakest-precondition function, and for (3) use the Black Box reduction rule for program call.

   Just give the answers; you do not have to show the calculation.

   Answer: For the purpose of applying the Black Box rule, you can treat the special variable `return` as if it is an implicit OUT-parameter. So, it is as if P has this header: `P(x, OUT y, OUT return)`, and the call $\mathtt{r} := \mathtt{P}(@\mathtt{x}, @\mathtt{y})$ can be seen as $\mathtt{P}(@\mathtt{x}, @\mathtt{y}, \mathtt{r})$.

   $$\begin{array}{lll} 4 & : & r + @y > 0 \\ 3 & : & @y > 0 \ \wedge\ ((r' + y')/@y > @x \ \Rightarrow\ r' + y' > 0) \\ 2 & : & k > 0 \ \wedge\ ((r' + y')/k > @x \ \Rightarrow\ r' + y' > 0) \\ 1 & : & k > 0 \ \wedge\ ((r' + y')/k > k{-}2 \ \Rightarrow\ r' + y' > 0) \end{array}$$

(b) Based on your calculation above, is the call correct? Motivate your answer.

Answer: No. The implication in the second conjunct is not implied by the given pre-condition. For example if both r' and y' are 0, then $r' + y'$ is not > 0.