Reasoning About Programs

Week 6 Tutorial - Basic Imperative Reasoning

Sophia Drossopoulou and Mark Wheelhouse

1st Question:

Consider the following Java method that triples its input through repeated addition:

```
1   int triple(int x){
2   // PRE: true
3   // POST: r = 3 * x
4       int y = x;
5       // MID: M1
6       y = y + x;
7       // MID: M2
8       y = y + x;
9       // MID: M3
10      return y;
11  }
```

a) Add mid-conditions M_1, M_2 and M_3 which are strong enough to prove that the method satisfies its specification.

b) Construct the **four** separate proof obligations that would be required to prove that the method satisfies its specification. (You do not need to prove them, but it should be trivial.)

A possible answer:

a)

$$\begin{array}{lcl} M_1 & \longleftrightarrow & \mathtt{y} = \mathtt{x} \\ M_2 & \longleftrightarrow & \mathtt{y} = 2 * \mathtt{x} \\ M_3 & \longleftrightarrow & \mathtt{y} = 3 * \mathtt{x} \end{array}$$

b) **line 5:** $\mathtt{PRE} \wedge \mathtt{y}' = \mathtt{x} \wedge \mathtt{x}' = \mathtt{x} \longrightarrow M_1[\mathtt{y} \mapsto \mathtt{y}', \mathtt{x} \mapsto \mathtt{x}']$

$$\begin{array}{c} \mathtt{y}' = \mathtt{x} \wedge \mathtt{x}' = \mathtt{x} \\ \longrightarrow \\ \mathtt{y}' = \mathtt{x}' \end{array}$$

line 7: $M_1 \wedge \mathtt{y}' = \mathtt{y} + \mathtt{x} \wedge \mathtt{x}' = \mathtt{x} \longrightarrow M_2[\mathtt{y} \mapsto \mathtt{y}', \mathtt{x} \mapsto \mathtt{x}']$

$$\begin{array}{c} \mathtt{y} = \mathtt{x} \wedge \mathtt{y}' = \mathtt{y} + \mathtt{x} \wedge \mathtt{x}' = \mathtt{x} \\ \longrightarrow \\ \mathtt{y}' = 2 * \mathtt{x}' \end{array}$$

line 9: $M_2 \wedge \mathtt{y}' = \mathtt{y} + \mathtt{x} \wedge \mathtt{x}' = \mathtt{x} \longrightarrow M_3[\mathtt{y} \mapsto \mathtt{y}', \mathtt{x} \mapsto \mathtt{x}']$

$$\begin{array}{c} \mathtt{y} = 2 * \mathtt{x} \wedge \mathtt{y}' = \mathtt{y} + \mathtt{x} \wedge \mathtt{x}' = \mathtt{x} \\ \longrightarrow \\ \mathtt{y}' = 3 * \mathtt{x}' \end{array}$$

line 11: $M_3 \wedge \mathtt{x} = \mathtt{x}_0 \wedge \mathbf{r} = \mathtt{y} \longrightarrow \mathtt{POST}[\mathtt{x} \mapsto \mathtt{x}_0]$

$$\begin{array}{c} \mathtt{y} = 3 * \mathtt{x} \wedge \mathbf{r} = \mathtt{y} \wedge \mathtt{x} = \mathtt{x}_0 \\ \longrightarrow \\ \mathbf{r} = 3 * \mathtt{x}_0 \end{array}$$

In this last case we must remember that due to Java's pass by value nature we have to consider the correctness of the post-condition with respect to the initial value of `x` (namely $\mathtt{x}_0$). However, because the method does not locally modify any of its arguments, we are allowed make use of the implicit fact that $\mathtt{x} = \mathtt{x}_0$ in our assumptions.

2nd Question:

Consider the following Java method that doubles its input through repeated addition, but also locally modifies its input argument:

```
1   int dodgeyDouble(int x){
2   // PRE: true
3   // POST: r = 2 * x
4       int y = x;
5       // MID: M1
6       x = x + 1;
7       // MID: M2
8       y = y + y;
9       // MID: M3
10      return y;
11  }
```

a) Add mid-conditions M_1, M_2 and M_3 which are strong enough to prove that the method satisfies its specification.

b) Construct the **four** separate proof obligations that would be required to prove that the method satisfies its specification. (You do not need to prove them, but it should be trivial.)

A possible answer:

a)

$$\begin{array}{lcl} M_1 & \longleftrightarrow & \mathtt{y} = \mathtt{x} \wedge \mathtt{x} = \mathtt{x}_0 \\ M_2 & \longleftrightarrow & \mathtt{y} = \mathtt{x}_0 \wedge \mathtt{x} = \mathtt{x}_0 + 1 \\ M_3 & \longleftrightarrow & \mathtt{y} = 2 * \mathtt{x}_0 \end{array}$$

b) **line 5:** $\mathtt{PRE} \wedge \mathtt{y}' = \mathtt{x} \wedge \mathtt{x}' = \mathtt{x} \wedge \mathtt{x} = \mathtt{x}_0 \longrightarrow M_1[\mathtt{y} \mapsto \mathtt{y}', \mathtt{x} \mapsto \mathtt{x}']$

$$\begin{array}{c} \mathtt{y}' = \mathtt{x} \wedge \mathtt{x}' = \mathtt{x} \wedge \mathtt{x} = \mathtt{x}_0 \\ \longrightarrow \\ \mathtt{y}' = \mathtt{x}' \wedge \mathtt{x}' = \mathtt{x}_0 \end{array}$$

line 7: $M_1 \wedge \mathtt{y}' = \mathtt{y} \wedge \mathtt{x}' = \mathtt{x} + 1 \longrightarrow M_2[\mathtt{y} \mapsto \mathtt{y}', \mathtt{x} \mapsto \mathtt{x}']$

$$\begin{array}{c} \mathtt{y} = \mathtt{x} \wedge \mathtt{x} = \mathtt{x}_0 \wedge \mathtt{y}' = \mathtt{y} \wedge \mathtt{x}' = \mathtt{x} + 1 \\ \longrightarrow \\ \mathtt{y}' = \mathtt{x}_0 \wedge \mathtt{x}' = \mathtt{x}_0 + 1 \end{array}$$

line 9: $M_2 \wedge \mathtt{y}' = \mathtt{y} + \mathtt{y} \wedge \mathtt{x}' = \mathtt{x} \longrightarrow M_3[\mathtt{y} \mapsto \mathtt{y}', \mathtt{x} \mapsto \mathtt{x}']$

$$\begin{array}{c} \mathtt{y} = \mathtt{x}_0 \wedge \mathtt{x} = \mathtt{x}_0 + 1 \wedge \mathtt{y}' = \mathtt{y} + \mathtt{y} \wedge \mathtt{x}' = \mathtt{x} \\ \longrightarrow \\ \mathtt{y}' = 2 * \mathtt{x}_0 \end{array}$$

line 11: $M_3 \wedge \mathbf{r} = \mathtt{y} \longrightarrow \mathtt{POST}[\mathtt{x} \mapsto \mathtt{x}_0]$

$$\begin{array}{c} \mathtt{y} = 2 * \mathtt{x}_0 \wedge \mathbf{r} = \mathtt{y} \\ \longrightarrow \\ \mathbf{r} = 2 * \mathtt{x}_0 \end{array}$$

In this last case we must remember that due to Java's pass by value nature we have to consider the correctness of the post-condition with respect to the initial value of `x` (namely $\mathtt{x}_0$).

However, because the method does locally modify `x`, we have had to more carefully track the value of `x` throughout the program. In particular, on line 7, we have to explicitly track that the value stored in `y` is that of the original value of `x` (namely $\mathtt{x}_0$) and not the current value of `x`.

3rd Question:

Consider the following Java method that performs a cheap, but rather inaccurate primality test:

```
1   boolean crudePrime(int x) {
2   // PRE: x ≥ 0
3   // POST: r  ⟶  ¬(∃m∈ℕ.[x = 2 * m ∨ x = 3 * m ∨ x = 5 * m])
4       boolean test = false
5       // MID: M1
6       test = (x % 2 == 0);
7       // MID: M2
8       test = test || (x % 3 == 0);
9       // MID: M3
10      test = test || (x % 5 == 0);
11      // MID: M4
12      return !test;
13  }
```

a) Add mid-conditions M_1, M_2, M_3 and M_4 which are strong enough to prove that the method satisfies its specification.

b) Construct the **five** separate proof obligations that would be required to prove that the method satisfies its specification. (You do not need to prove them, but it should be trivial.)

A possible answer:

a)

$$\begin{array}{lcl} M_1 & \longleftrightarrow & \neg\texttt{test} \\ M_2 & \longleftrightarrow & \texttt{test} \longleftrightarrow \exists m{\in}\mathbb{N}.[\texttt{x} = 2 * m] \\ M_3 & \longleftrightarrow & \texttt{test} \longleftrightarrow \exists m{\in}\mathbb{N}.[\texttt{x} = 2 * m \vee \texttt{x} = 3 * m] \\ M_4 & \longleftrightarrow & \texttt{test} \longleftrightarrow \exists m{\in}\mathbb{N}.[\texttt{x} = 2 * m \vee \texttt{x} = 3 * m \vee \texttt{x} = 5 * m] \end{array}$$

b) **line 5:** $\texttt{PRE}[\texttt{x} \mapsto \texttt{x}_0] \wedge \texttt{test} = \texttt{false} \wedge \texttt{x}' = \texttt{x} \wedge \texttt{x} = \texttt{x}_0 \longrightarrow M_1[\texttt{x} \mapsto \texttt{x}']$

$$\begin{array}{c} \texttt{x}_0 \geq 0 \wedge \texttt{test} = \texttt{false} \wedge \texttt{x}' = \texttt{x} \wedge \texttt{x} = \texttt{x}_0 \\ \longrightarrow \\ \neg\texttt{test} \end{array}$$

line 7: $M_1 \wedge (\texttt{test}' \longleftrightarrow \exists m{\in}\mathbb{N}.[\texttt{x} = 2 * m]) \wedge \texttt{x}' = \texttt{x} \longrightarrow M_2[\texttt{test} \mapsto \texttt{test}', \texttt{x} \mapsto \texttt{x}']$

$$\begin{array}{c} \neg\texttt{test} \wedge (\texttt{test}' \longleftrightarrow \exists m{\in}\mathbb{N}.[\texttt{x} = 2 * m]) \wedge \texttt{x}' = \texttt{x} \\ \longrightarrow \\ \texttt{test}' \longleftrightarrow \exists m{\in}\mathbb{N}.[\texttt{x}' = 2 * m] \end{array}$$

line 9: $M_2 \wedge (\mathtt{test}' \longleftrightarrow (\mathtt{test} \vee \exists m \in \mathbb{N}.[\mathtt{x} = 3 * m])) \wedge \mathtt{x}' = \mathtt{x} \longrightarrow M_3[\mathtt{test} \mapsto \mathtt{test}', \mathtt{x} \mapsto \mathtt{x}']$

$$\begin{array}{c} (\mathtt{test} \longleftrightarrow \exists m \in \mathbb{N}.[\mathtt{x} = 2 * m]) \\ \wedge\, (\mathtt{test}' \longleftrightarrow (\mathtt{test} \vee \exists m \in \mathbb{N}.[\mathtt{x} = 3 * m])) \wedge \mathtt{x}' = \mathtt{x} \\ \longrightarrow \\ \mathtt{test}' \longleftrightarrow \exists m \in \mathbb{N}.[\mathtt{x}' = 2 * m \vee \mathtt{x}' = 3 * m] \end{array}$$

line 11: $M_3 \wedge (\mathtt{test}' \longleftrightarrow (\mathtt{test} \vee \exists m \in \mathbb{N}.[\mathtt{x} = 5 * m])) \wedge \mathtt{x}' = \mathtt{x} \longrightarrow M_4[\mathtt{test} \mapsto \mathtt{test}', \mathtt{x} \mapsto \mathtt{x}']$

$$\begin{array}{c} (\mathtt{test} \longleftrightarrow \exists m \in \mathbb{N}.[\mathtt{x} = 2 * m \vee \mathtt{x} = 3 * m]) \\ \wedge\, (\mathtt{test}' \longleftrightarrow (\mathtt{test} \vee \exists m \in \mathbb{N}.[\mathtt{x} = 5 * m])) \wedge \mathtt{x}' = \mathtt{x} \\ \longrightarrow \\ \mathtt{test}' \longleftrightarrow \exists m \in \mathbb{N}.[\mathtt{x}' = 2 * m \vee \mathtt{x}' = 3 * m \vee \mathtt{x}' = 5 * m] \end{array}$$

line 13: $M_4 \wedge \mathtt{x} = \mathtt{x}_0 \wedge \mathbf{r} = \neg \mathtt{test} \longrightarrow \mathtt{POST}[\mathtt{x} \mapsto \mathtt{x}_0]$

$$\begin{array}{c} (\mathtt{test} \longleftrightarrow \exists m \in \mathbb{N}.[\mathtt{x} = 2 * m \vee \mathtt{x} = 3 * m \vee \mathtt{x} = 5 * m]) \wedge \mathtt{x} = \mathtt{x}_0 \wedge \mathbf{r} = \neg \mathtt{test} \\ \longrightarrow \\ \mathbf{r} \rightarrow \neg(\exists m \in \mathbb{N}.[\mathtt{x}_0 = 2 * m \vee \mathtt{x}_0 = 3 * m \vee \mathtt{x}_0 = 5 * m]) \end{array}$$

As with Question 1 above, because the method does not locally modify any of its arguments, we are allowed make use of the implicit fact that $\mathtt{x} = \mathtt{x}_0$ in our assumptions.