IIT CS536: Science of Programming

Proofs, Loop Invariants

My Dinh

March 28, 2022

1 Minimal and Full Proof Outlines

Task 1.1 (Written, 7 points).

$$
\begin{array}{ll}
 & \{n \le 0\} \\
i := \overline{0}; & \{n \le 0 \ \wedge i = 0\} \\
s := \overline{0}; & \{n \le 0 \wedge i = 0 \wedge s = 0\} \\
\{\textbf{inv } i \le n \wedge s = i^2\} & \\
\mathsf{while}\ (i < n)\{ & \{i < n \wedge i \le n \wedge s = i^2\} \Rightarrow \{i < n \wedge s = i^2\} \\
\quad s := s + (2 * i + 1); & \{i < n \wedge s = i^2 + (2 * i + 1)\} \Rightarrow \{i < n \wedge s = (i+1)^2\} \\
\quad i := i + \overline{1} & \{i < n + 1 \wedge s = i^2\} \Rightarrow \{i \le n \wedge s = i^2\} \\
\} & \{i \ge n \wedge i \le n \wedge s = i^2\} \Rightarrow \{s = n^2\}
\end{array}
$$

Task 1.2 (Written, 5 points).

$$
\begin{array}{ll}
 & \{n \le 0\} \\
i := \overline{0}; & \{n \le 0 \ \wedge i = 0\} \\
s := \overline{0}; & \{n \le 0 \wedge i = 0 \wedge s = 0\} \\
\{\textbf{inv } i \le n \wedge s = i^2\} & \\
\mathsf{while}\ (i < n)\ \{ & \{i < n \wedge i \le n \wedge s = i^2\} \Rightarrow \{i < n \wedge s = i^2\} \\
\quad s := s + (2 * i); & \{i < n \wedge s = i^2 + (2 * i)\} \Rightarrow \{i < n \wedge s = (i+1)^2 - 1\} \\
\quad i := i + \overline{1} & \{i < n + 1 \wedge s = i^2 - 1\} \Rightarrow \{i \le n \wedge s < i^2\} \\
\} & \{s = n^2\}
\end{array}
$$

We know that $\{i \le n \wedge s < i^2\} \not\Rightarrow \{i \le n \wedge s = i^2\}$. Thus the logic obligation after assigning new value to i cannot be proven and the loop invariant does not hold at the end of each loop. The program is incorrect.

2 Proofs with Loops

Task 2.1 (Programming, 8 points).

$$
\begin{array}{ll}
 & \{T\} \\
i := \overline{0}; & \\
s := \overline{0}; & \\
\{\textbf{inv } i \le |a| \wedge s = sumA(a, 0, i)\} & \\
\mathsf{while}\ (i < |a|)\ \{ & \\
\quad s := s + a[i]; & \\
\quad i := i + \overline{1} & \\
\} & \{s = sumA(a, 0, |a|)\}
\end{array}
$$

Dafny code of the program in `sumarray.dfy`:

```
function arrsum (a: seq<int>, i: int, j: int) : int
requires i >= 0 && j <= |a|
decreases (j - i)
  { if j <= i then 0 else a[j-1] + arrsum(a, i, j - 1) }

method sumArray (a: seq<int>) returns (s: int)
ensures s == arrsum(a, 0, |a|)
{
    var i := 0;
    s := 0;
    while (i < |a|)
    invariant (i <= |a| && s == arrsum(a, 0, i))
    {
      s := s + a[i];
      i := i + 1;
    }
}
```

Task 2.2 (Programming, 12 points).

a) **Postcondition:** $(i < |a| \rightarrow a[i] > x \land (\forall j.0 \leq j \land j < i \rightarrow a[j] \leq x)) \land (i = |a| \rightarrow (\forall j.0 \leq j \land j < |a| \rightarrow a[j] \leq x))$

b) Program and proof outline:

$\{T\}$
$i := \overline{0};$
$\{\textbf{inv } i \leq |a| \land (\forall j.0 \leq j \land j < i \rightarrow a[j] \leq x)\}$
while $(i < |a| \land a[i] \leq x)$ {
  $i := i + \overline{1}$
}
$\{(i < |a| \rightarrow a[i] > x \land (\forall j.0 \leq j \land j < i \rightarrow a[j] \leq x))$
$\land (i = |a| \rightarrow (\forall j.0 \leq j \land j < |a| \rightarrow a[j] \leq x))\}$

Dafny code of the program in `find.dfy`:

```
method findFirstGreater (a: seq<int>, x: int) returns (i : int)
ensures (0 <= i < |a| ==> a[i] > x && forall j :: 0 <= j < i ==> a[j] <= x) &&
(i == |a| ==> forall j :: 0 <= j < |a| ==> a[j] <= x)
{
    i := 0;
    while (i < |a| && a[i] <= x)
    invariant (i <= |a| && forall j :: 0 <= j < i ==> a[j] <= x)
    {
        i := i + 1;
    }
}
```

Task 2.3 (Programming, 12 points).

$\{T\}$

$i := \overline{0};$
$n := \overline{0};$
$p := \overline{0};$
$\{\textbf{inv } i \leq |a| \wedge p = numPos(a, 0, i) \wedge n = numNeg(a, 0, i)\}$
while $(i < |a|)$ {
  if $a[i] > \overline{0}$ then $\{p := p + \overline{1}\}$ else {skip};
  if $a[i] < \overline{0}$ then $\{n := n + \overline{1}\}$ else {skip};
  $i := i + \overline{1}$
}
$e := n = p$ $\{e \rightarrow numPos(a, 0, |a|) = numNeg(a, 0, |a|)\}$

Dafny code of the program in `posneg.dfy`:

```
function numPos(a: seq<int>, i: int, j: int) : int
requires i >= 0 && j <= |a|
{
    if i >= j then 0
    else if a[j - 1] > 0 then 1 + numPos(a, i, j - 1) else numPos(a, i, j - 1)
}

function numNeg(a: seq<int>, i: int, j: int) : int
requires i >= 0 && j <= |a|
{
    if i >= j then 0
    else if a[j - 1] < 0 then 1 + numNeg(a, i, j - 1) else numNeg(a, i, j - 1)
}

method eqPosNeg (a: seq<int>) returns (e: bool)
ensures e ==> numPos(a, 0, |a|) == numNeg(a, 0, |a|)
{
    var i := 0;
    var npos := 0;
    var nneg := 0;
    while (i < |a|)
    invariant (i <= |a| && npos == numPos(a, 0, i) && nneg == numNeg(a, 0, i))
    {
        if a[i] > 0 { npos := npos + 1; }
        if a[i] < 0 { nneg := nneg + 1; }
        i := i + 1;
    }
    e := npos == nneg;
}
```

3 Weakest Preconditions with Array Assignments

Task 3.1 (Written, 6 points).

a) $$\begin{aligned} wlp(a[x=0\,?\,i\,:\,j] := \overline{1}, a[i]=1) &= [1/a[x=0\,?\,i\,:\,j]](a[i]=1) \\ &= ((x=0\,?\,i\,:\,j)=i)\,?\,(1=1)\,:\,(a[i]=1) \\ &= ((x=0\,?\,i\,:\,j)=i)\,?\,T\,:\,(a[i]=1) \\ &= (x=0\,?\,i=i\,:\,j=i)\,?\,T\,:\,(a[i]=1) \\ &= (x=0\,?\,T\,:\,j=i)\,?\,T\,:\,(a[i]=1) \\ &= (x=0 \vee j=i)\,?\,T\,:\,(a[i]=1) \\ &= (x=0 \vee j=i) \vee (a[i]=1) \\ &= (x=0) \vee (j=i) \vee (a[i]=1) \end{aligned}$$

b) $$\begin{aligned} wlp(a[i] := \overline{5}, a[a[1]]=5) &= [5/a[i]](a[a[1]]=5) \\ &= (e=i)\,?\,5=5\,:\,a[e]=5 \qquad \text{where } e=[5/a[i]](a[1]) = (i=1)\,?\,5\,:\,a[1] \\ &= (e=i)\,?\,T\,:\,a[e]=5 \\ &= (e=i) \vee (a[e]=5) \\ &= (((i=1)\,?\,5\,:\,a[1])=i) \vee ((a[(i=1)\,?\,5\,:\,a[1]])=5) \\ &= ((i=1)\,?\,5=i\,:\,a[1]=i) \vee ((i=1)\,?\,a[5]\,:\,a[a[1]]=5) \\ &= (i\neq 1 \wedge a[1]=i) \vee ((i=1)\,?\,a[5]=5\,:\,a[a[1]]=5) \\ &= (i\neq 1 \wedge a[1]=i) \vee ((i=1 \rightarrow a[5]=5) \wedge (i\neq 1 \rightarrow a[a[1]]=5)) \end{aligned}$$

c) $$\begin{aligned} wlp(a[j] := a[i]+1, a[j]>a[i]) &= [a[i]+1/a[j]](a[j]>a[i]) \\ &= [a[i]+1/a[j]](a[j]) > [a[i]+1/a[j]](a[i]) \\ &= a[i]+1 > (i=j\,?\,a[i]+1\,:\,a[i]) \\ &= i=j\,?\,a[i]+1>a[i]+1\,:\,a[i]+1>a[i] \\ &= i=j\,?\,F\,:\,T \\ &= i\neq j \end{aligned}$$

4 One more wrap-up question

I spent about 5 hours on this homework, in total 2 hours of actual working time.