Reasoning About Programs

Week 8 Coursework - Loops (part 2)
Answers to be submitted to the SAO by 2pm on Monday, 6th March

Sophia Drossopoulou and Mark Wheelhouse

Learning Aims:

A1: discovering loop invariants

A2: reasoning using loop invariants and lemmas

A3: finding interesting variants

Summary

We will prove the correctness of a Java function `find`, which, given strings `s` and `t`, returns the *first* location in `s`, where `t` occurs as a substring.

Preliminaries

Recall that the `find` function is specified in terms of the predicate $Match$ and the function $Find$ which are defined as follows:

$$\begin{aligned} Match(\mathtt{s},\mathtt{t},m,n) \longleftrightarrow\ & 0 \le m+n \le \mathtt{s.length} \ \wedge\ 0 \le n \le \mathtt{t.length} \\ & \wedge\ \forall k \in [0..n).[\ \mathtt{s}[m+k] = \mathtt{t}[k]\ ] \end{aligned}$$

$$Find(\mathtt{s},\mathtt{t}) \quad = \quad \begin{cases} \mathtt{s.length} & \text{if } \ \nexists m.\ Match(\mathtt{s},\mathtt{t},m,\mathtt{t.length}) \\ min\{\, m \mid Match(\mathtt{s},\mathtt{t},m,\mathtt{t.length})\,\} & \text{otherwise} \end{cases}$$

Recall that we use the notation $k \in [m..n)$ to indicate that $m \le k < n$.

You can assume that the following properties hold (you do not need to prove them), for all strings `s` and `t`, and integers m, n and n':

Lemma 0: $0 \le n' \le n \ \wedge\ Match(\mathtt{s},\mathtt{t},m,n) \ \longrightarrow\ Match(\mathtt{s},\mathtt{t},m,n')$
Lemma 1: $0 \le m \le \mathtt{s.length} \ \longrightarrow\ Match(\mathtt{s},\mathtt{t},m,0)$
Lemma 2: $\mathtt{t.length} = 0 \ \longrightarrow\ Find(\mathtt{s},\mathtt{t}) = 0$
Lemma 3: $Find(\mathtt{s},\mathtt{t}) \ge 0$
Lemma 4: $Find(\mathtt{s},\mathtt{t}) = \mathtt{s.length} \ \vee\ Find(\mathtt{s},\mathtt{t}) \le \mathtt{s.length} - \mathtt{t.length}$

1st Question:

Consider the code for `find` given below. For this week's assignment, you are asked to:

a) Prove that the initialisation code (lines 5-7) establishes the invariant I.

b) Prove that the loop body (lines 16-23) re-establishes the invariant I.

c) Prove that all array access are valid.

d) Prove that the invariant I and termination of the loop imply the mid-condition M.

In answering these questions you may either use the invariant provided below[1], or your own invariant from the previous coursework.

The code and the specification of `find`

```
1   int find( char[] s, char[] t )
2   // PRE: s ≠ null  ∧  t ≠ null
3   // POST: r = Find(s0, t0)
4   {
5       int i = 0;
6       int j = 0;
7       boolean found = (t.length == 0);
8
9       // INV: s ≈ s0  ∧  t ≈ t0                                            (I1)
10      //      ∧  0 ≤ i ≤ s.length  ∧  0 ≤ j ≤ t.length  ∧  i ≥ j           (I2)
11      //      ∧  Match(s, t, i − j, j)                                      (I3)
12      //      ∧  i − j − 1 < Find(s, t)                                     (I4)
13      //      ∧  found  ⟷  j = t.length                                    (I5)
14      while ( j < t.length && i < s.length && !found ) {
15          if ( s[i] == t[j] ) {
16              i++;
17              j++;
18              if ( j == t.length ) { found = true; }
19          } else {
20              i = i - j + 1;
21              j = 0;
22          }
23      }
24      // MID: s ≈ s0  ∧  t ≈ t0                                            (M1)
25      //      ∧  found  ⟶  Find(s, t) = i − j                              (M2)
26      //      ∧  !found  ⟶  Find(s, t) = s.length                          (M3)
27
28      if ( found ) {
29          return i - j;
30      } else {
31          return s.length;
32      }
33  }
```

[1] Note that in our invariant we treat line-breaks as brackets to aid readability.

A possible answer:

Marking Scheme (35 marks)

While the UTAs have some discretion in marking the submissions, this marking scheme is indicative of how this work would be marked in exams, and the relative importance of the various parts.

(a) Prove that the initialization code establishes the invariant.

Because the code does not modify the arrays `s` or `t`, we will not distinguish between `t.length` and $\mathtt{t}_0$`.length`, nor between `s.length` and $\mathtt{s}_0$`.length`.

Given:

(C1)	$\mathtt{s} \approx \mathtt{s}_0 \quad \land \quad \mathtt{t} \approx \mathtt{t}_0$	implicit from code
(C2)	$\mathtt{i} = 0$	from code line 5
(C3)	$\mathtt{j} = 0$	from code line 6
(C4)	$\mathtt{found} \leftrightarrow \mathtt{t.length} = 0$	from code line 7

To Show:

(I1) $\mathtt{s} \approx \mathtt{s}_0 \quad \land \quad \mathtt{t} \approx \mathtt{t}_0$

(I2) $0 \le \mathtt{i} \le \mathtt{s.length} \quad \land \quad 0 \le \mathtt{j} \le \mathtt{t.length} \quad \land \quad \mathtt{i} \ge \mathtt{j}$

(I3) $Match(\mathtt{s}, \mathtt{t}, \mathtt{i\text{-}j}, \mathtt{j})$

(I4) $\mathtt{i} - \mathtt{j} - 1 < Find(\mathtt{s}, \mathtt{t})$

(I5) $\mathtt{found} \ \leftrightarrow \ \mathtt{j} = \mathtt{t.length}$

Proof:

(I1) follows from (C1).

(I2) follows from (C2) and (C3).

(I3) follows from Lemma 1 and (C2), (C3).

(I4) follows (C2), (C3), which give that $\mathtt{i\text{-}j\text{-}1} = \mathtt{\text{-}1} < 0$, and application of Lemma 3.

For the proof of (I5) there are two approaches:

The first approach is a direct proof substituting (C3) into (C4) to get:

$$\mathtt{found} \ \leftrightarrow \ \mathtt{t.length} = \mathtt{j}$$

You can then rearrange the right-hand side to (I5).

The second approach is to proceed by case analysis on the length of `t`:

1st Case: (D) $\mathtt{t.length} = 0$

From (C4) and (D) we obtain that $\mathtt{found} = \mathtt{true}$. Moreover from (D) and (C3) we have that $\mathtt{j} = \mathtt{t.length}$. This establishes that (I5) holds when both sides are true.

2nd Case: (E) $\mathtt{t.length} \neq 0$

From (C4) and (E) we obtain that $\mathtt{found} = \mathtt{false}$. Moreover from (E) and (C3) we have that $\mathtt{j} \neq \mathtt{t.length}$. This establishes that (I5) holds when both sides are false.

Both approaches are equally valid.

[9 marks] Suggested Marking Scheme:

- **(1 mark)** for code effect in givens (or clear in proof).
- **(1 mark)** for implicit code effect in givens (or clear in proof).
- **(1 mark)** for invariant in to show.
- **(1 mark)** for a proof of $(I1)$.
- **(1 mark)** for a proof of $(I2)$.
- **(1 mark)** for a proof of $(I3)$.
- **(1 mark)** for a proof of $(I4)$.
- **(1 mark)** for a proof of $(I5)$ (first case).
- **(1 mark)** for a proof of $(I5)$ (second case).

(b) Prove that the loop body re-establishes the invariant.

Given:

(I1)	$\texttt{s} \approx \texttt{s}_0 \quad \land \quad \texttt{t} \approx \texttt{t}_0$	INV
(I2)	$0 \leq \texttt{i} \leq \texttt{s.length} \quad \land \quad 0 \leq \texttt{j} \leq \texttt{t.length} \quad \land \quad \texttt{i} \geq \texttt{j}$	INV
(I3)	$Match(\texttt{s}, \texttt{t}, \texttt{i-j}, \texttt{j})$	INV
(I4)	$\texttt{i} - \texttt{j} - 1 < Find(\texttt{s}, \texttt{t})$	INV
(I5)	$\texttt{found} \ \leftrightarrow \ \texttt{j} = \texttt{t.length}$	INV
(cond)	$\texttt{i<s.length} \land \texttt{j<t.length} \land \texttt{!found}$	from loop condition
code	`i'`, `j'`, as in code from lines 16 - 22	

To Show:

(I1') $\texttt{s}' \approx \texttt{s}_0 \quad \land \quad \texttt{t}' \approx \texttt{t}_0$
(I2') $0 \leq \texttt{i'} \leq \texttt{s}'\texttt{.length} \quad \land \quad 0 \leq \texttt{j'} \leq \texttt{t}'\texttt{.length} \quad \land \quad \texttt{i'} \geq \texttt{j'}$
(I3') $Match(\texttt{s}', \texttt{t}', \texttt{i'-j'}, \texttt{j'})$
(I4') $\texttt{i'} - \texttt{j'} - 1 < Find(\texttt{s}', \texttt{t}')$
(I5') $\texttt{found'} \ \leftrightarrow \ \texttt{j'} = \texttt{t}'\texttt{.length}$

Proof:

Because the code in lines 16-22 does not modify the arrays `s` and `t`, we obtain (I1') from (I1).

For the remaining properties we proceed by case analysis:

1st Case: (A) `s[i]=t[j]`

Then, by code, we have that

(C) $\texttt{i'} = \texttt{i+1} \ \land \ \texttt{j'} = \texttt{j+1}$.

From (cond), (I2) and (C) we obtain (I2').

From (I3) and definition of $Match$, we obtain that:

(D) $\forall k \in [0..j).\texttt{s[i-j+k]} = \texttt{t[k]}$.

This, together with (A), gives that:

(E) $\forall k \in [0..j+1).\texttt{s[i-j+k]} = \texttt{t[k]}$.

Moreover, from (C) we have that `i'-j'=i-j`, and `j'=j+1`. This, together with (E) gives that:

(F) $\forall k \in [0..j')$.`s[i'-j'+k]` = `t[k]`.

By folding the definition of $Match$, we obtain (I3').

Moreover, from (C) we obtain that `i-j-1=i'-j'-1`. Therefore, from (I4), we obtain (I4').

1.a Case: (G) `j'==t.length`

Then, by code, we have that `found'=true`. This, together with (G) establishes the validity of (I5').

1.b Case: (G) `j'!=t.length`

Then, by code, we have that `found'=found`. This, together with (cond) gives that `found'=false`. This, together with (G) establishes the validity of (I5').

2nd Case: (A) `s[i]`$\neq$`t[j]` Then, by code we obtain

(B) `i'=i-j+1` $\wedge$ `j'=0`.

From (I2) we have `i` $\geq$ `j`, and therefore,

(C) `i'` ≥ 0.

Moreover, `i-j+1` $\leq$ `i+1`, and therefore, using using (cond) we obtain $0 \leq$ `i'` $\leq$ `s.length`. From (C) and (B) we obtain that `i'` $\geq$ `j'`. From (B) we obtain that `j'` $\leq$ `t.length`. All this together establishes (I2').

From (B) and Lemma 1, we obtain (I3').

Because of (A) we have that $NOT(Match($`s`, `t`, `i-j`, `j+1`$)$. This, together with the fact that `j` $<$ `t.length` (by cond) and Lemma 0, gives that

(D) $NOT(Match($`s`, `t`, `i-j`, `t.length`$))$.

From (D) and (I4) we obtain

(E) `i-j` $< Find($`s`, `t`$)$.

From (B), we have that `i'-j'-1=i-j+1-0-1=i-j`. Replacing this into (E) gives (I4').

From (cond) we have that `found=false`. Therefore, because the code does not modify `found`, we obtain

(F) `found'` = `false`.

We also have that `j` $<$ `t.length` (from (cond)), and $0 \leq$ `j` (from (I2)). These two facts give that $0 <$ `t.length`. Therefore, using (B), we get

(G) `j'` $\neq$ `t.length`.

From (F) and (G) we get (I5').

[14 marks] Suggested Marking Scheme:

- **(1 mark)** for invariant in givens.
- **(1 mark)** for loop condition in givens.
- **(1 mark)** for first case code effect.
- **(1 mark)** for second case code effect.
- **(1 mark)** for identifying the case split (either within one proof or over two proofs).
- **(1 mark)** for a proof of $I1$.
- **(1 mark)** for case 1 proof of $I2$.
- **(1 mark)** for case 1 proof of $I3$.

- **(1 mark)** for case 1 proof of $I4$.
- **(1 mark)** for case 1 proof of $I5$.
- **(1 mark)** for case 2 proof of $I2$.
- **(1 mark)** for case 2 proof of $I3$.
- **(1 mark)** for case 2 proof of $I4$.
- **(1 mark)** for case 2 proof of $I5$.

(c) Prove that array accesses are valid.

Given:

(I1)	$\texttt{s} \approx \texttt{s}_0 \quad \wedge \quad \texttt{t} \approx \texttt{t}_0$	INV
(I2)	$0 \le \texttt{i} \le \texttt{s.length} \quad \wedge \quad 0 \le \texttt{j} \le \texttt{t.length} \quad \wedge \quad \texttt{i} \ge \texttt{j}$	INV
(I3)	$Match(\texttt{s}, \texttt{t}, \texttt{i-j}, \texttt{j})$	INV
(I4)	$\texttt{i} - \texttt{j} - 1 < Find(\texttt{s}, \texttt{t})$	INV
(I5)	$\texttt{found} \leftrightarrow \texttt{j} = \texttt{t.length}$	INV
(cond)	$\texttt{i<s.length} \ \wedge\ \texttt{j<t.length} \ \wedge\ \texttt{!found}$	from loop condition

To Show:

$0 \le \texttt{j} < \texttt{t.length}$
$0 \le \texttt{i} < \texttt{s.length}$

Proof: From (cond) and from (I2) we have that on line 16, it holds that $0 \le \texttt{j} < \texttt{t.length}$ and $0 \le \texttt{i} < \texttt{s.length}$.

[4 marks] Suggested Marking Scheme:

- **(1 mark)** for invariant in givens.
- **(1 mark)** for loop condition in givens.
- **(1 mark)** for correct identification of properties to show.
- **(1 mark)** proof follow-through.

(d) Prove that termination and the invariant imply the mid-condition.

Given:

(I1)	$\texttt{s} \approx \texttt{s}_0 \quad \wedge \quad \texttt{t} \approx \texttt{t}_0$	INV
(I2)	$0 \le \texttt{i} \le \texttt{s.length} \quad \wedge \quad 0 \le \texttt{j} \le \texttt{t.length} \quad \wedge \quad \texttt{i} \ge \texttt{j}$	INV
(I3)	$Match(\texttt{s}, \texttt{t}, \texttt{i-j}, \texttt{j})$	INV
(I4)	$\texttt{i} - \texttt{j} - 1 < Find(\texttt{s}, \texttt{t})$	INV
(I5)	$\texttt{found} \leftrightarrow \texttt{j} = \texttt{t.length}$	INV
(C6)	$\texttt{!(i<s.length)}\ \mathit{or}\ \texttt{!(j<t.length)}\ \mathit{or}\ \texttt{found}$	negation of loop condition

To Show:

(M1) $\texttt{s} \approx \texttt{s}_0 \quad \wedge \quad \texttt{t} \approx \texttt{t}_0$
(M2) $\texttt{found} \rightarrow Find(\texttt{s}, \texttt{t}) = \texttt{i} - \texttt{j}$
(M3) $\texttt{!found} \rightarrow Find(\texttt{s}, \texttt{t}) = \texttt{s.length}$

Proof: (M1) follows trivially from the invariant (I1).

We now prove (M2):
We assume that (A) `found` holds, and want to show that $\texttt{i-j} = Find(\texttt{s}, \texttt{t})$.
From (A) and (I5) we get `j=t.length`. From that and (I3), we get that $Match(\texttt{s}, \texttt{t}, \texttt{i-j}, \texttt{t.length})$.
This, together with (I4) gives that $\texttt{i-j} = Find(\texttt{s}, \texttt{t})$.
Therefore, (M2) is proven.

We finally prove (M3):
We assume that (A) `!found` holds, and want to show that $\texttt{s.length} = Find(\texttt{s}, \texttt{t})$.
From (A) and (I5) we obtain that $\texttt{j} \neq \texttt{t.length}$, and this, together with (I2) gives

(B) $\texttt{j} < \texttt{t.length}$.

From (B), (C6), and (A) we obtain that `!(i<s.length)`. The latter, combined with (I2) gives that

(D) $\texttt{i} = \texttt{s.length}$.

Therefore, (B) and (D) give that

(F) $\texttt{i-j-1} \geq \texttt{s.length} - \texttt{t.length}$.

Lemma 4, (F) and (I4) give that $Find(\texttt{s}, \texttt{t}) = \texttt{s.length}$.
Therefore, (M3) is proven.

[8 marks] Suggested Marking Scheme:

- **(1 mark)** for invariant in givens.
- **(1 mark)** for negation of loop condition in givens.
- **(1 mark)** for midcondition in to show.
- **(1 mark)** for a proof of $(M1)$.
- **(1 mark)** for a proof of $(M2)$.
- **(1 mark)** for establishing that $\texttt{j} < \texttt{t.length}$ in proof of $(M3)$.
- **(1 mark)** for establishing that $\texttt{i} = \texttt{s.length}$ in proof of $(M3)$.
- **(1 mark)** for a full proof of $(M3)$.