Lecture 11:
More Invariants

Aims:

- To discuss how to discover invariants;
- To see more examples of proofs that use invariants.

11.1 Hints for Discovering Invariants

- As mentioned in the previous lecture, finding invariants may require some ingenuity. In this lecture, we look at some rules of thumb that may help you.

11.2 Hint 1: Base the Invariant on the Postcondition

- Look at the postcondition of the **while** command.

  If it's a conjunction, can you break it into two $\mathit{Inv} \wedge \neg B$?

- Suppose you have a program, *ProgA* below, which, it is claimed, computes $x \operatorname{div} y$ and $x \bmod y$, i.e.

$$(\!|\, x \geq 0 \wedge y > 0 \,|\!)\ \mathit{ProgA}\ (\!|\, x = q \times y + r \wedge 0 \leq r \wedge r < y \,|\!)$$

  There are three conjuncts in the postcondition. So there are several ways to split it up. But we know the loop test is $r \geq y$. If we negate this, we get $r < y$. So this suggests that the rest of the postcondition is the invariant:

$$x = q \times y + r \wedge 0 \leq r$$

$q := 0;$

$r := x;$

while $r \geq y$
{

  $q := q + 1;$

  $r := r - y;$

}

11.3 Hint 2: Replace a Constant by a Variable

- Take the postcondition, replace a constant by a variable and then split the postcondition into two as per Hint 1.
- In this example, a is an array of integers, indexed from 1 to n. We want to prove that

  $$\vdash_{\text{par}} (\!|\, 0 < n \,|\!)\ \mathit{ProgB}\ (\!|\, s = \sum_{i=1}^{i=n} a[i] \,|\!)$$

  where *ProgB* is below.

  To some of you, the invariant may be obvious anyway. But, in case it's not, let's see how Hint 2 helps us.

  Here is an equivalent postcondition: $(\!|\, s = \sum_{i=1}^{i=k} a[i] \wedge k = n \,|\!)$

  This is now easily split into invariant

  $$s = \sum_{i=1}^{i=k} a[i]$$

  and negation of loop test

  $$k = n$$

Let's confirm this by completing the proof.

$s := 0;$

$k := 0;$

while $k \neq n$
$\{$

$\quad k := k + 1;$

$\quad s := s + a[k]$

$\}$

11.4 Hint 3: Tracing

- The invariant can often be found by tracing the loop and inspecting the values of the variables.
- For example, let's trace the factorial program that we proved in the previous lecture. Here it is again for ease of reference:

$(\!|$ **True** $|\!)$
$y := 1;$
$z := 0;$
while $z \neq x$
$\{ \quad z := z + 1;$
$\quad y := y \times z;$
$\}$
$(\!| y = x! |\!)$

We will set x to 6. The first two commands (prior to the loop) assign 1 to y and 0 to z.

Here's a trace of the loop in the factorial program:

B	iteration	x	y	z
—	—	6	1	0
true	1	6	1	1
true	2	6	2	2
true	3	6	6	3
true	4	6	24	4
true	5	6	120	5
true	6	6	720	6
false	—			

We see that $y = z!$

11.5 Hint 4: Do Some Quick Checks

- When you have guessed an invariant, if possible, quickly check that you think that the proofs will go through.
  - Is the invariant *strong enough* that, when conjoined with the negation of the loop-test, it will imply the postcondition of the **while** command?
  - Is the invariant *weak enough* that it will be implied by the precondition of the **while** command?

Acknowledgements

I continue to base material on that in Chapter 4 of [HR00].

References

[HR00] M. Huth and M. Ryan. *Logic in Computer Science: Modelling and Reasoning about Systems*. Cambridge University Press, 2000.