MATH/CSCI 2112, DISCRETE STRUCTURES I, FALL 2005

Handout 7: Program Correctness Proof Example

General strategy

Proving the correctness of a program involves 5 steps:

1. Write down the program
2. Write down the specification
3. Guess a loop invariant for each loop
4. Generate proof obligations
5. Discharge the proof obligations

Example: Division with remainder

As an example, we give the correctness proof of a program for division with remainder.

Step 1: Write down the program

The input to this program are two integers a and d. The output are the quotient q and the remainder r. Here is the program:

```
r := a
q := 0
while r ⩾ d do
  r := r − d
  q := q + 1
endwhile
```

Step 2: Write down the specification

To write down the "specification" of the program means to write down a *precondition* and *postcondition*. The precondition specifies on what inputs this program can be run. The postcondition specifies the properties of the output.

For the above program, we can see that if d is zero or negative, the program will loop forever. The program is not designed to work for $d \leqslant 0$, so we state that $d > 0$ as a precondition for the program.

Also, if a is negative, the program will return an incorrect result, because in this case, the body of the while loop is never executed, so the quotient will always be given (incorrectly) as $q = 0$. The program is not designed to work for $a < 0$. We therefore state $a \geqslant 0$ as a precondition.

The postcondition specifies what the program is supposed to compute. We want that q is the integer quotient, and r is the remainder of the division of a by d. Mathematically, this is specified by the following two conditions: $a = qd + r$ and $0 \leqslant r < d$.

The specification for this program is therefore:

Precondition: $\{a \geqslant 0, d > 0\}$
Postcondition: $\{a = qd + r, \quad 0 \leqslant r < d\}$

Step 3: Guess a loop invariant for each loop

Writing down a loop invariant is a tricky business. We need to do this for each while loop in the program. A loop invariant is a condition that is always satisfied just before the boolean condition of the while loop is evaluated. We must find a loop in variant that is (a) correct, (b) strong enough to imply the postcondition, and (c) strong enough to imply termination of the loop.

Guessing a correct loop invariant can be aided by running the program on some test inputs, and recording the values of all the program variables just before the boolean condition in the "while" statement is evaluated. This is called "tracing" the program.

We run the program with inputs $a = 10$ and $d = 3$. We obtain the following trace table. Here, k is a counter of how many times the loop body has been evaluated, and B is the boolean condition of the while loop ($B \equiv \{r \geqslant d\}$ in our example).

k	a	d	q	r	B
0	10	3	0	10	T
1	10	3	1	7	T
2	10	3	2	4	T
3	10	3	3	1	F

The loop invariant will be a formula $I(k)$ that is true after the kth iteration of the loop. By looking at the trace table and at the program, we can express each of the changeable variables as a function of k:

$$q = k$$
$$r = a - kd$$

We also know that a and d never change, so we can record that the precondition continues to hold:

$$a \geqslant 0$$
$$d > 0$$

Finally, we need a boundary condition. Since r is only decremented when $r \geqslant d$, and since r is decremented by d, we know that $r \geqslant 0$ will hold all the time.

$$r \geqslant 0$$

Our loop invariant is the conjunction of these five conditions.

$$I(k) \equiv \{q = k, \quad r = a - kd, \quad a \geqslant 0, \quad d > 0, \quad r \geqslant 0\}$$

Step 4: Generate proof obligations

A "proof obligation" is something that must be proven in order to establish the correctness of the program. We generate proof obligations by filling in a table of pre- and postconditions, using the rules from Handout 6. We do this in a three-column format: the center column contains the program (split into individual lines for each statement), and the left and right columns contain pre- and postconditions, respectively. We also number the lines, with some room to spare, for easy reference.

We start with the information we already have: the pre- and postcondition, and the loop invariant. By the while rule, the precondition for the body of the while loop (in line 5) is $I(k) \wedge B$, and the postcondition for the body of the while loop (in line 7) is $I(k+1)$. Also, the precondition for the entire while loop (in line 4) is $I(0)$, and the postcondition for the entire while loop (in line 8) is $I(k) \wedge \sim\!B$. Here, $B \equiv \{r \geqslant d\}$ is the boolean condition of the while loop, as before, and $I(k)$ is the loop invariant from step 3. The pre- and postconditions for the whole program, from step 2, go in lines 1 and 9, respectively.

	Precondition	Statement	Postcondition
1.	$S \quad\quad \equiv \{a \geqslant 0, d > 0\}$		
2.		`r := a`	
3.		`q := 0`	
4.	$P \equiv I(0) \equiv \{q = 0, r = a - 0d,$ $a \geqslant 0, d > 0, r \geqslant 0\}$	while `r` $\geqslant$ `d` do	
5.	$I(k) \wedge B \equiv \{q = k, r = a - kd,$ $a \geqslant 0, d > 0, r \geqslant 0, r \geqslant d\}$		
6.		`r := r - d`	
7.		`q := q + 1`	$I(k+1) \quad \equiv \{q = k+1, r = a - (k+1)d,$ $a \geqslant 0, d > 0, r \geqslant 0\}$
8.		endwhile	$I(k) \wedge \sim B \equiv \{q = k, r = a - kd,$ $a \geqslant 0, d > 0, r \geqslant 0, r < d\}$
9.			$Q \quad\quad \equiv \{a = qd + r, \quad 0 \leqslant r < d\}$

Note that there are already three proof obligations, from the hypotheses of the while-rule.

Proof obligation 1: $P \Rightarrow I(0)$
Proof obligation 2: $I(k) \wedge \sim B \Rightarrow Q$
Proof obligation 3: $\exists k \geqslant 0 (I(k) \longrightarrow \sim B)$

In proof obligation 1, P is the precondition for the while loop; we solve this obligation by simply letting $P = I(0)$. Proof obligation 2 is because we must show that line 8 implies line 9. Proof obligation 3 is to ensure that the loop terminates (it is also taken from while-rule).

Next, we fill in the remaining lines of the table, using the assignment rule. We work backwards from line 7 to line 6, and from line 4 to line 2.

	Precondition	Statement	Postcondition
1.	$S \quad\quad \equiv \{a \geqslant 0, d > 0\}$		
2.	$\{0 = 0, a = a - 0d, a \geqslant 0, d > 0, 0 \geqslant 0\}$	`r := a`	$\{0 = 0, r = a - 0d, a \geqslant 0, d > 0, r \geqslant 0\}$
3.	$\{0 = 0, r = a - 0d, a \geqslant 0, d > 0, r \geqslant 0\}$	`q := 0`	$\{q = 0, r = a - 0d, a \geqslant 0, d > 0, r \geqslant 0\}$
4.	$P \equiv I(0) \equiv \{q = 0, r = a - 0d,$ $a \geqslant 0, d > 0, r \geqslant 0\}$	while `r` $\geqslant$ `d` do	
5.	$I(k) \wedge B \equiv \{q = k, r = a - kd,$ $a \geqslant 0, d > 0, r \geqslant 0, r \geqslant d\}$		
6.	$\{q + 1 = k + 1, r - d = a - (k+1)d,$ $a \geqslant 0, d > 0, r - d \geqslant 0\}$	`r := r - d`	$\{q + 1 = k + 1, r = a - (k+1)d,$ $a \geqslant 0, d > 0, r \geqslant 0\}$
7.	$\{q + 1 = k + 1, r = a - (k+1)d,$ $a \geqslant 0, d > 0, r \geqslant 0\}$	`q := q + 1`	$I(k+1) \quad \equiv \{q = k+1, r = a - (k+1)d,$ $a \geqslant 0, d > 0, r \geqslant 0\}$
8.		endwhile	$I(k) \wedge \sim B \equiv \{q = k, r = a - kd,$ $a \geqslant 0, d > 0, r \geqslant 0, r < d\}$
9.			$Q \quad\quad \equiv \{a = qd + r, \quad 0 \leqslant r < d\}$

We now have two additional proof obligations. The first one is that the precondition from line 5 implies that on line 6:

Proof obligation 4: $I(k) \wedge B \Rightarrow \{q + 1 = k + 1, r - d = a - (k+1)d,$
$a \geqslant 0, d > 0, r - d \geqslant 0\}$

This establishes $\{I(k) \wedge B\}\, C\, \{I(k+1)\}$, and therefore finishes the last remaining obligation of the while rule.

We also have an obligation to prove the correctness of the precondition, i.e., to prove that the precondition on line 1 implies that of line 2:

Proof obligation 5: $S \Rightarrow \{0 = 0, a = a - 0d, a \geqslant 0, d > 0, 0 \geqslant 0\}$

Step 5: Discharge the proof obligations

"Discharging" a proof obligation means to prove it. We have to prove the five statements collected in step 4.

Proof obligation 1. There is nothing to show, since we decided to let $P \equiv I(0)$.

Proof obligation 2. Assume $I(k) \land \not B$. In other words, assume the following are true:

$$\{q = k, r = a - kd, a \geqslant 0, d > 0, r \geqslant 0, r \ngeqslant d\}$$

We need to prove Q, i.e., $\{a = qd + r, 0 \leqslant r < d\}$. But clearly, since $r = a - kd$, we have $a = kd + r$, and since $q = k$, therefore $a = qd + r$. Also, $0 \leqslant r$ is true, because it was assumed. Finally, $r < d$ is true, because $r \ngeqslant d$ was assumed. Therefore, this proof obligation has been met.

Proof obligation 3. We have to show that there exists some $k \geqslant 0$ such that $I(k) \rightarrow \sim B$ is true. Since we only have to prove that such a k exists, we are free to choose any k we want. Let $k = a$, and assume $I(k)$. Then $\{q = k, r = a - kd, a \geqslant 0, d > 0, r \geqslant 0\}$. Since $d > 0$, we have $1 - d \leqslant 0$ (by algebra, and the fact that d is an integer). Since $a = k$, we have $r = a - kd = a - ad = a(1 - d)$. Since $a \geqslant 0$ and $1 - d \leqslant 0$, we have $r = a(1-d) \leqslant 0$. On the other hand, $d > 0$, therefore $r = 0 < d$, hence $r < d$. It follows that the condition B, which is $r \geqslant d$, is false. This finishes this proof obligation.

Proof obligation 4. Assume $I(k) \land B$, in other words, $\{q = k, r = a - kd, a \geqslant 0, d > 0, r \geqslant 0, r \geqslant d\}$. We have to prove $\{q + 1 = k + 1, r - d = a - (k + 1)d,$. But we have assumed $q = k$, therefore $q + 1 = k + 1$. We have
$a \geqslant 0, d > 0, r - d \geqslant 0\}$
assumed $r = a - kd$, and therefore $r - d = a - kd - d = a - (k + 1)d$, by algebra. Also, $a \geqslant 0$ and $d > 0$ hold because they have been assumed. Finally, $r - d \geqslant 0$ holds because we assumed $r \geqslant d$. This finishes the fourth proof obligation.

Proof obligation 5. Assume S, i.e., assume $\{a \geqslant 0, d > 0\}$. We must prove $\{0 = 0, a = a - 0d, a \geqslant 0, d > 0, 0 \geqslant 0\}$. But clearly, $0 = 0$ and $0 \geqslant 0$ are true. Also, $a = a - 0d$ is true since $0d = 0$. And $a \geqslant 0$ and $d > 0$ are true by assumption. This proves the fifth and last proof obligation.

The correctness of the program has therefore been proved.