



BITS Pilani
Pilani Campus



CS/IS F214 Logic in Computer Science

MODULE: **PROGRAM VERIFICATION**

Floyd-Hoare Logic: Total Correctness: Termination Arguments

Termination of Programs

- Given the limited set of constructs in our language, we need to verify termination only for loops:
 - When will a sequence of statements $S1; S2$ terminate?
 - When will a conditional statement $\text{if}(B) \text{ then } S1; \text{ else } S2$ terminate?



Algorithm for termination?

- **term(p)** */* p is a program */*
 - case p is assignment statement:
 - TRUE
 - case p is a sequence of the form S1; S2:
 - **term(S1) AND term(S2)**
 - case p is a conditional of the form if (B) then S1; else S2 :
 - **term(S1) AND term(S2)**
/ AND term(B) if B can be complex
 e.g. B can include calls to functions*/*
 - case p is a loop of the form while (B) { S; } :
 - **termWhile(B, S);**



Hoare Logic – Partial Correctness vs. Total Correctness

Consider the following program intended to compute the *gcd* of *x* and *y*:

/* Pre-condition:

$x=A \wedge y=B \wedge x \geq 0 \wedge y \geq 0$ */

while (*y* != 0) {

t = *x* % *y*;

x = *y*;

y = *t*;

}

/* Post-condition: $x = \text{gcd}(A, B)$ */

How do you prove that the loop terminates?

This states that *if the loop terminates* then $x = \text{gcd}(A, B)$

Termination Argument – Finite and Reducing Quantity

Program intended to compute the *gcd* of *x* and *y*:

/* Pre-condition:

$x=A \wedge y=B \wedge x \geq 0 \wedge y \geq 0$ */

while (y != 0) {

 t = x % y;

 x = y;

 y = t;

}

/* Post-condition: $x = \text{gcd}(A, B)$ */

(Typical) Termination Argument:

Identify a “quantity” that

*a) is finite before the loop
and*

b) reduces in each iteration

Termination Argument – Finite and Reducing Quantity - Example

Program intended to compute the *gcd* of *x* and *y*:

/* Pre-condition:

$x=A \wedge y=B \wedge x \geq 0 \wedge y \geq 0$ */

while (*y* != 0) {

t = *x* % *y*;

x = *y*;

y = *t*;

}

/* Post-condition: $x = \text{gcd}(A, B)$

*/

(Typical) Termination Argument:

Identify a “quantity” that

a) is finite before the loop

and

b) reduces in each iteration

/ y is finite before the loop */*

...

/ y gets smaller in each iteration */*

Termination Argument – Finite and Reducing Quantity - Proof

Program intended to compute the *gcd* of *x* and *y*:

/* Pre-condition:

$x=A \wedge y=B \wedge x \geq 0 \wedge y \geq 0$ */

while (*y* != 0) {

t = *x* % *y*;

x = *y*;

y = *t*;

}

/*Post-condition:

$x = \text{gcd}(A, B)$ */

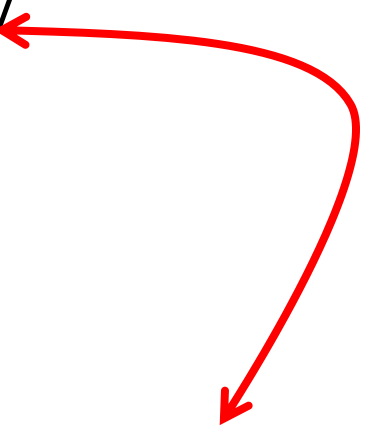
Identify a “quantity” that

- is finite before the loop and
- reduces in each iteration

```

/* y=y0 ... */
t=x%y;
/* ... */
x=y;
/* ... */
y=t;
/* ... y<y0 */

```



Finite and Reducing Quantity – Proof using Hoare Logic

Program intended to compute the ***gcd*** of ***x*** and ***y***:

/* Pre-condition:

$x=A \wedge y=B \wedge x \geq 0 \wedge y \geq 0$ */

while ($y \neq 0$) {

$t = x \% y$;

$x = y$;

$y = t$;

}

/*Post-condition:

$x = \text{gcd}(A, B)$ */

Identify a “quantity” that

- a) is finite before the loop
- and
- b) reduces in each iteration

Use rules of Hoare logic to prove this!

/* $y=y_0$ */
 /* $y=y_0 \wedge (x\%y < y_0)$ */
 $t=x\%y$;
 /* $y=y_0 \wedge t < y_0$ */
 $x=y$;
 /* $x=y_0 \wedge t < y_0$ */
 $y=t$;
 /* $x=y_0 \wedge y < y_0$ */

because $x\%y < y$

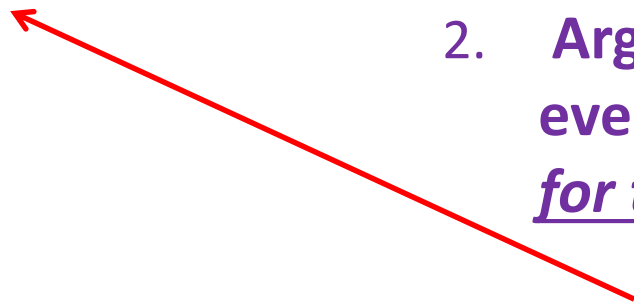
Hoare Logic – Termination Argument – Step 2

Program intended to compute the *gcd* of *x* and *y*:

/* Pre-condition:

$x=A \wedge y=B \wedge x \geq 0 \wedge y \geq 0$ */

while ($y \neq 0$) {

$t = x \% y$; 

$x = y$;

$y = t$;

}

/*Post-condition:

$x = \text{gcd}(A, B)$ */

(Typical) Termination Argument:

1. Identify a “quantity” (*y* in this case) that
 - a) is finite before the loop and
 - b) reduces in each iteration
2. Argue that the quantity will eventually satisfy the condition for termination
 i.e. *will negate the iterative condition*

Termination Argument – Meeting the Termination Condition

Program intended to compute the *gcd* of *x* and *y*:

/* Pre-condition:

$x=A \wedge y=B \wedge x \geq 0 \wedge y \geq 0$ */

while (*y* != 0) {

t = *x* % *y*;

x = *y*;

y = *t*;

}

/*Post-condition:

$x = \text{gcd}(A, B)$ */

(Typical) Termination Argument:

1. Identify a “quantity” (*y* in this case) that
 - a) is finite before the loop and
 - b) reduces in each iteration
2. Argue that the quantity will eventually satisfy the condition for termination

As *y* continues to reduce it will reach **0** i.e.

- it will not stop at some $d > 0$
- nor it will it jump over **0** and become negative!

WHY?

Finite and Reducing Quantity Need Not Result in Termination!

- [Recall from the previous example:]
 - *The decreasing sequence of remainders will stop at 0.*
- Can it be generalized?
 - *A decreasing natural number sequence will terminate.*
 - This is not necessarily true for programs: *e.g.*

```
/* Precondition:  $x > 0$  */  
while (x != 0) {  
    x = x - 2;  
}
```
 - Can you change the termination condition (in this example) so that the statement terminates?



Finite and Reducing Quantity Need Not Result in Termination!

- This can be generalized:
 - *An appropriate termination condition be defined for a decreasing natural number sequence.*
- How about real numbers?
 - Consider the sequence (for some finite real value r)
 - $r, r/2, r/4, \dots$
 - Will this sequence terminate (i.e. converge)?
 - Consider the same sequence, for floating point numbers:
 - will it terminate?
 - What is *underflow*?



Termination Proof - Example 2

- Consider the following program:

```
float x=1.0;  
int n=0;  
while (x!=0) {  
    p=x;  
    x = x/2;  
    n=n+1;  
}
```

- Prove that the program terminates.
- Guess a desired (i.e. *useful*) post-condition for the program.

