GCD Proof

This is an example of a program to compute the greatest common divisor (GCD) of two positive integers — this is the largest number that is a whole divisor of each number. In this proof we rely on the following properties of GCD without proving them

- $X>Y \Rightarrow GCD(X,Y) = GCD(X-Y,Y)$
- $GCD(X,Y) = GCD(Y,X)$
- $GCD(X,X) = X$

In this example, only a proof of partial correctness of the program is provided. A separate analysis is required to see that the program is totally correct, that is, that it halts for all values that satisfy the pre-condition. This is shown by noticing that one of the inner loops will keep iterating as long as the values of A and B are not equal, and each such iteration must reduce the absolute difference of these values. Hence the total number of iterations of the inner and the outer loops can be no more than this difference.

To prove partial correctness of

```
    { X > 0 ∧ Y > 0 }
A := X;  B := Y;
     { ℙ }
while A ≠ B do
begin
while A > B do A := A–B;
while B > A do B := B–A
end
    { A = GCD(X,Y) }
```

we first need to formulate a loop invariant ℙ. The loop invariant characterizes the "approximating strategy" of this program. In this case, that strategy is to preserve GCD(X,Y) = GCD(A,B) at all times, while always decreasing the absolute difference of A and B. Therefore, when finally A = B and all loops terminate, GCD(X,Y) = GCD(A,B) = GCD(A,A) = A.

The loop invariant ℙ is taken to be $GCD(X,Y) = GCD(A,B) \wedge A > 0 \wedge B > 0$.

<u>Step 1</u> (actually it's three steps that we abbreviate to one):
Clearly,

```
    ⊢ { X > 0  ∧ Y > 0 }
A := X;  B := Y;
     { ℙ }
```

by two applications of the axiom of assignment, plus the sequential rule.

Since this GCD program involves two nested loops, we need to formulate the loop invariants for these loops in order to complete the proof. Fortunately, the same loop invariant suffices for all the loops.

```
        { $\mathbb{P}$ }
   while A ≠ B do
   begin
        { $\mathbb{P}$ }
   while A > B do A := A–B;
        { $\mathbb{P}$ }
   while B > A do B := B–A
   end
{ A = GCD(X,Y) }
```

Step 2:

$\vdash$ { $\mathbb{P}$ }
while A > B **do** A := A–B;
{ $\mathbb{P}$ }

is proven by step 2A and the while rule since the post-condition $\mathbb{P} \wedge A \le B$ can be weakened to $\mathbb{P}$.

Step 2A:

$\vdash$ { $\mathbb{P} \wedge A > B$ } A := A–B { $\mathbb{P}$ }

by the axiom of assignment since

$\mathbb{P}[A \rightarrow A–B] \equiv GCD(X,Y) = GCD(A–B,B) \wedge A–B > 0 \wedge B > 0$

which is logically equivalent to $\mathbb{P} \wedge A > B$.

Step 3:

$\vdash$ { $\mathbb{P}$ }
while B > A **do** B := B–A
{ $\mathbb{P}$ }

is proven by step 3A and the while rule since the post-condition $\mathbb{P} \wedge B \le A$ can be weakened to $\mathbb{P}$.

Step 3A:

$\vdash$ { $\mathbb{P} \wedge B > A$} B := B–A { $\mathbb{P}$ }

by the axiom of assigment since

$\mathbb{P}\,[B \rightarrow B–A] \equiv GCD(X,Y) = GCD(A,B–A) \wedge A > 0 \wedge B–A > 0$

which is logically equivalent to $\mathbb{P} \wedge B > A$.

Step 4:

By Steps 2 and 3 and the sequential execution rule

$\vdash$ { $\mathbb{P}$ }

```
   begin
   while A > B do A := A–B;
   while B > A do B := B–A
   end
        { ℙ }
```

Step 5:
By Step 4 and the while rule

```
      | {   ℙ }
   while A ≠ B do
   begin
   while A > B do A := A–B;
   while B > A do B := B–A
   end
{ ℙ ∧ A=B }
```

Step 6:
$\mathbb{P} \wedge A=B \equiv (GCD(X,Y) = GCD(A,B) \wedge A > 0 \wedge B > 0 \wedge A=B) \Rightarrow A = GCD(X,Y)$, so by Step 5 and weakening the post-condition

```
      | {   ℙ }
   while A ≠ B do
   begin
   while A > B do A := A–B;
   while B > A do B := B–A
   end
{ A = GCD(X,Y) }
```

Step 7:
By Steps 1 and 6 and the sequential rule, the program is proven.