Program Verification using Hoare Logic
- An Introduction -

Peter Heinig

Technical University of Munich

March 2007, JASS 2007

```
function recursive(x,y)
```

```
function recursive(x,y)
  if x == 0
    disp (y);
```

```
function recursive(x,y)
  if x == 0
    disp (y);
  else
    recursive(x-1,y+1);
  end
```

```
function recursive(x,y)
  if x == 0
    disp (y);
  else
    recursive(x-1,y+1);
  end
```

$\left[x \in \mathbb{N}_0 \ \wedge \ y \in \mathbb{Z}\right]$

```
function recursive(x,y)
  if x == 0
    disp (y);
  else
    recursive(x-1,y+1);
  end
```

$\left[\text{If the program terminates, the value of } x + y \text{ gets printed.}\right]$

$\left[x \in \mathbb{N}_0 \ \wedge \ y \in \mathbb{Z}\right]$

```
function recursive(x,y)
  if x == 0
    disp (y);
  else
    recursive(x-1,y+1);
  end
```

$\left[\text{If the program terminates, the value of } x + y \text{ gets printed.}\right]$

How can we prove this assertion?

$$\left[x \in \mathbb{N}_0 \ \wedge \ y \in \mathbb{Z}\right]$$

```
function recursive(x,y)
  if x == 0
    disp (y);
  else
    recursive(x-1,y+1);
  end
```

$$\left[\text{If the program terminates, the value of } x + y \text{ gets printed.}\right]$$

How can we prove this assertion? Easy.

$\left[x \in \mathbb{N}_0 \ \wedge \ y \in \mathbb{Z}\right]$

```
function recursive(x,y)
  if x == 0
    disp (y);
  else
    recursive(x-1,y+1);
  end
```

[If the program terminates, the value of $x + y$ gets printed.]

$\left[x \in \mathbb{N}_0 \ \wedge \ y \in \mathbb{Z}\right]$

```
function recursive(x,y)
  if x == 0
    disp (y);
  else
    recursive(x-1,y+1);
  end
```

$\left[\text{If the program terminates, the value of } x + y \text{ gets printed.}\right]$

Proof of Correctness: (Induction on the first argument)

$\left[x \in \mathbb{N}_0 \ \wedge \ y \in \mathbb{Z}\right]$

```
function recursive(x,y)
  if x == 0
    disp (y);
  else
    recursive(x-1,y+1);
  end
```

$\left[\text{If the program terminates, the value of } x + y \text{ gets printed.}\right]$

Proof of Correctness: (Induction on the first argument)

If $\left[x = 0\right]$, then

$\forall y \in \mathbb{Z}: \ \left[\texttt{recursive(x,y)}\right] \overset{\text{Function Definition}}{\Rightarrow} \left[y = x + y \text{ gets printed}\right].$

$\left[x \in \mathbb{N}_0 \;\wedge\; y \in \mathbb{Z}\right]$

```
function recursive(x,y)
  if x == 0
    disp (y);
  else
    recursive(x-1,y+1);
  end
```

$\left[\text{If the program terminates, the value of } x + y \text{ gets printed.}\right]$

Proof of Correctness: (Induction on the first argument)

If $\left[x = 0\right]$, then

$\forall y \in \mathbb{Z}: \; \left[\texttt{recursive(x,y)}\right] \overset{\text{Function Definition}}{\Rightarrow} \left[y = x + y \text{ gets printed}\right].$

Now assume that for some $0 \leq x \in \mathbb{N}_0$:

$\left[x \in \mathbb{N}_0 \ \wedge \ y \in \mathbb{Z}\right]$

```
function recursive(x,y)
  if x == 0
    disp (y);
  else
    recursive(x-1,y+1);
  end
```

$\left[\text{If the program terminates, the value of } x + y \text{ gets printed.}\right]$

Proof of Correctness: (Induction on the first argument)
If $\left[x = 0\right]$, then
$\forall y \in \mathbb{Z}: \ \left[\texttt{recursive(x,y)}\right] \overset{\text{Function Definition}}{\Rightarrow} \left[y = x + y \text{ gets printed}\right].$
Now assume that for some $0 \leq x \in \mathbb{N}_0$:
$\forall y \in \mathbb{Z}: \ \left[\texttt{recursive(x,y)}\right] \Rightarrow \left[x + y \text{ gets printed}\right].$

$\left[x \in \mathbb{N}_0 \ \wedge \ y \in \mathbb{Z}\right]$

```
function recursive(x,y)
  if x == 0
    disp (y);
  else
    recursive(x-1,y+1);
  end
```

$\left[\text{If the program terminates, the value of } x + y \text{ gets printed.}\right]$

Proof of Correctness: (Induction on the first argument)

If $\left[x = 0\right]$, then

$\forall y \in \mathbb{Z} : \ \left[\texttt{recursive(x,y)}\right] \overset{\text{Function Definition}}{\Rightarrow} \left[y = x + y \text{ gets printed}\right].$

Now assume that for some $0 \leq x \in \mathbb{N}_0$:

$\forall y \in \mathbb{Z} : \ \left[\texttt{recursive(x,y)}\right] \Rightarrow \left[x + y \text{ gets printed}\right].$

Then $\left[\texttt{recursive(x+1,y)}\right]$

$\left[x \in \mathbb{N}_0 \ \wedge \ y \in \mathbb{Z}\right]$

```
function recursive(x,y)
  if x == 0
    disp (y);
  else
    recursive(x-1,y+1);
  end
```

$\left[\text{If the program terminates, the value of } x + y \text{ gets printed.}\right]$

Proof of Correctness: (Induction on the first argument)
If $\left[x = 0\right]$, then

$\forall y \in \mathbb{Z} : \ \left[\texttt{recursive(x,y)}\right] \overset{\text{Function Definition}}{\Rightarrow} \left[y = x + y \text{ gets printed}\right].$

Now assume that for some $0 \leq x \in \mathbb{N}_0$:
$\forall y \in \mathbb{Z} : \ \left[\texttt{recursive(x,y)}\right] \Rightarrow \left[x + y \text{ gets printed}\right].$

Then $\left[\texttt{recursive(x+1,y)}\right] \overset{\text{Function Definition}}{\Rightarrow} \left[\texttt{recursive(x,y+1)}\right]$

$\left[x \in \mathbb{N}_0 \;\wedge\; y \in \mathbb{Z}\right]$

```
function recursive(x,y)
  if x == 0
    disp (y);
  else
    recursive(x-1,y+1);
  end
```

$\left[\text{If the program terminates, the value of } x + y \text{ gets printed.}\right]$

Proof of Correctness: (Induction on the first argument)

If $\left[x = 0\right]$, then

$\forall y \in \mathbb{Z} : \left[\texttt{recursive(x,y)}\right] \overset{\text{Function Definition}}{\Rightarrow} \left[y = x + y \text{ gets printed}\right]$.

Now assume that for some $0 \leq x \in \mathbb{N}_0$:

$\forall y \in \mathbb{Z} : \left[\texttt{recursive(x,y)}\right] \Rightarrow \left[x + y \text{ gets printed}\right]$.

Then $\left[\texttt{recursive(x+1,y)}\right] \overset{\text{Function Definition}}{\Rightarrow} \left[\texttt{recursive(x,y+1)}\right]$

$\overset{\text{Inductive Assumption}}{\Rightarrow} \left[x + (y + 1) \text{ gets printed}\right]$,

$\left[x \in \mathbb{N}_0 \;\wedge\; y \in \mathbb{Z}\right]$

```
function recursive(x,y)
  if x == 0
    disp (y);
  else
    recursive(x-1,y+1);
  end
```

[If the program terminates, the value of $x + y$ gets printed.]

Proof of Correctness: (Induction on the first argument)
If $[x = 0]$, then
$\forall y \in \mathbb{Z}:$ [`recursive(x,y)`] $\overset{\text{Function Definition}}{\Longrightarrow}$ [$y = x + y$ gets printed].
Now assume that for some $0 \leq x \in \mathbb{N}_0$:
$\forall y \in \mathbb{Z}:$ [`recursive(x,y)`] $\Rightarrow$ [$x + y$ gets printed].
Then [`recursive(x+1,y)`] $\overset{\text{Function Definition}}{\Longrightarrow}$ [`recursive(x,y+1)`]
$\overset{\text{Inductive Assumption}}{\Longrightarrow}$ [$x + (y + 1)$ gets printed],
and $x + (y + 1) = (x + 1) + y$.

$\left[x \in \mathbb{N}_0 \;\wedge\; y \in \mathbb{Z}\right]$

```
function recursive(x,y)
  if x == 0
    disp (y);
  else
    recursive(x-1,y+1);
  end
```

$\left[\text{If the program terminates, the value of } x + y \text{ gets printed.}\right]$

Proof of Correctness: (Induction on the first argument)

If $\left[x = 0\right]$, then

$\forall y \in \mathbb{Z}: \; \left[\texttt{recursive(x,y)}\right] \overset{\text{Function Definition}}{\Rightarrow} \left[y = x + y \text{ gets printed}\right].$

Now assume that for some $0 \leq x \in \mathbb{N}_0$:

$\forall y \in \mathbb{Z}: \; \left[\texttt{recursive(x,y)}\right] \Rightarrow \left[x + y \text{ gets printed}\right].$

Then $\left[\texttt{recursive(x+1,y)}\right] \overset{\text{Function Definition}}{\Rightarrow} \left[\texttt{recursive(x,y+1)}\right]$

$\overset{\text{Inductive Assumption}}{\Rightarrow} \left[x + (y + 1) \text{ gets printed}\right],$

and $x + (y + 1) = (x + 1) + y.$ $\square$

```
function iterative(x,y)
  while x > 0
    x = x-1;
    y = y+1;
  end
  disp (y);
```

```
function iterative(x,y)
  while x > 0
    x = x-1;
    y = y+1;
  end
  disp (y);
```

$\left[x \in \mathbb{N}_0 \ \wedge \ y \in \mathbb{Z}\right]$

```
function iterative(x,y)
  while x > 0
    x = x-1;
    y = y+1;
  end
  disp (y);
```

[If the program terminates, the value of $x + y$ gets printed.]

$\left[x \in \mathbb{N}_0 \ \wedge \ y \in \mathbb{Z}\right]$

```
function iterative(x,y)
  while x > 0
    x = x-1;
    y = y+1;
  end
  disp (y);
```

$\left[\text{If the program terminates, the value of } x + y \text{ gets printed.}\right]$

How can we prove this assertion?

$\left[x \in \mathbb{N}_0 \;\wedge\; y \in \mathbb{Z}\right]$

```
function iterative(x,y)
  while x > 0
    x = x-1;
    y = y+1;
  end
  disp (y);
```

[If the program terminates, the value of $x + y$ gets printed.]

How can we prove this assertion? Easy?

$$\left[x \in \mathbb{N}_0 \ \wedge \ y \in \mathbb{Z}\right]$$

```
function iterative(x,y)
  while x > 0
    x = x-1;
    y = y+1;
  end
  disp (y);
```

$$\left[\text{If the program terminates, the value of } x + y \text{ gets printed.}\right]$$

How can we prove this assertion? Easy? Lets try it again.

```
function iterative(x,y)
  while x > 0
    x = x-1;
    y = y+1;
  end
  disp (y);
```

```
function iterative(x,y)
  while x > 0
    x = x-1;
    y = y+1;
  end
  disp (y);
```

Proof of Correctness:

```
function iterative(x,y)
  while x > 0
    x = x-1;
    y = y+1;
  end
  disp (y);
```

Proof of Correctness: (Induction on the first argument

```
function iterative(x,y)
  while x > 0
    x = x-1;
    y = y+1;
  end
  disp (y);
```

Proof of Correctness: (Induction on the first argument ???)

```
function iterative(x,y)
  while x > 0
    x = x-1;
    y = y+1;
  end
  disp (y);
```

Proof of Correctness: (Induction on the first argument ???)

If $\left[x = 0\right]$, then

$\forall y \in \mathbb{Z} : \left[\texttt{iterative(x,y)}\right] \overset{\text{Function Definition}}{\Rightarrow} \left[y = x + y \text{ gets printed}\right]$.

```
function iterative(x,y)
  while x > 0
    x = x-1;
    y = y+1;
  end
  disp (y);
```

Proof of Correctness: (Induction on the first argument ???)

If $[x = 0]$, then

$\forall y \in \mathbb{Z}$: [iterative(x,y)] $\overset{\text{Function Definition}}{\Rightarrow}$ $[y = x + y$ gets printed$]$.

Now assume that for some $0 \leq x \in \mathbb{N}_0$:

```
function iterative(x,y)
  while x > 0
    x = x-1;
    y = y+1;
  end
  disp (y);
```

Proof of Correctness: (Induction on the first argument ???)

If $[x = 0]$, then

$\forall y \in \mathbb{Z} : [\texttt{iterative(x,y)}] \overset{\text{Function Definition}}{\Rightarrow} [y = x + y \text{ gets printed}]$.

Now assume that for some $0 \leq x \in \mathbb{N}_0$:

$\forall y \in \mathbb{Z} : [\texttt{iterative(x,y)}] \Rightarrow [x + y \text{ gets printed}]$.

```
function iterative(x,y)
  while x > 0
    x = x-1;
    y = y+1;
  end
  disp (y);
```

Proof of Correctness: (Induction on the first argument ???)

If $[x = 0]$, then

$\forall y \in \mathbb{Z}: \ [\texttt{iterative(x,y)}] \overset{\text{Function Definition}}{\Rightarrow} [y = x + y \text{ gets printed}]$.

Now assume that for some $0 \leq x \in \mathbb{N}_0$:

$\forall y \in \mathbb{Z}: \ [\texttt{iterative(x,y)}] \Rightarrow [x + y \text{ gets printed}]$.

Then $[\texttt{iterative(x+1,y)}]$

```
function iterative(x,y)
  while x > 0
    x = x-1;
    y = y+1;
  end
  disp (y);
```

Proof of Correctness: (Induction on the first argument ???)

If $[x = 0]$, then

$\forall y \in \mathbb{Z}: \left[\texttt{iterative(x,y)}\right] \overset{\text{Function Definition}}{\Longrightarrow} \left[y = x + y \text{ gets printed}\right].$

Now assume that for some $0 \leq x \in \mathbb{N}_0$:

$\forall y \in \mathbb{Z}: \left[\texttt{iterative(x,y)}\right] \Rightarrow \left[x + y \text{ gets printed}\right].$

Then $\left[\texttt{iterative(x+1,y)}\right] \overset{\text{Function Definition}}{\Longrightarrow}$

```
function iterative(x,y)
  while x > 0
    x = x-1;
    y = y+1;
  end
  disp (y);
```

Proof of Correctness: (Induction on the first argument ???)

If $\left[x=0\right]$, then

$\forall y \in \mathbb{Z}: \left[\texttt{iterative(x,y)}\right] \overset{\text{Function Definition}}{\Rightarrow} \left[y = x + y \text{ gets printed}\right]$.

Now assume that for some $0 \leq x \in \mathbb{N}_0$:

$\forall y \in \mathbb{Z}: \left[\texttt{iterative(x,y)}\right] \Rightarrow \left[x + y \text{ gets printed}\right]$.

Then $\left[\texttt{iterative(x+1,y)}\right] \overset{\text{Function Definition}}{\Rightarrow} \left[\,?\,\right]$

```
function iterative(x,y)
  while x > 0
    x = x-1;
    y = y+1;
  end
  disp (y);
```

Proof of Correctness: (Induction on the first argument ???)

If $[x = 0]$, then

$\forall y \in \mathbb{Z}: \; [\texttt{iterative(x,y)}] \overset{\text{Function Definition}}{\Rightarrow} [y = x + y \text{ gets printed}]$.

Now assume that for some $0 < x \in \mathbb{N}_0$:

$\forall y \in \mathbb{Z}: \; [\texttt{iterative(x,y)}] \Rightarrow [x + y \text{ gets printed}]$.

Then $[\texttt{iterative(x+1,y)}] \overset{\text{Function Definition}}{\Rightarrow} [\texttt{iterative(x+1,y)}]$

```
function iterative(x,y)
  while x > 0
    x = x-1;
    y = y+1;
  end
  disp (y);
```

Proof of Correctness: (Induction on the first argument ???)

If $[x = 0]$, then

$\forall y \in \mathbb{Z} :$ $[\texttt{iterative(x,y)}] \overset{\text{Function Definition}}{\Rightarrow} [y = x + y \text{ gets printed}]$.

Now assume that for some $0 < x \in \mathbb{N}_0$:

$\forall y \in \mathbb{Z} :$ $[\texttt{iterative(x,y)}] \Rightarrow [x + y \text{ gets printed}]$.

Then $[\texttt{iterative(x+1,y)}] \overset{\text{Function Definition}}{\Rightarrow} [\texttt{iterative(x+1,y)}]$

Useless.

```
function iterative(x,y)
  while x > 0
    x = x-1;
    y = y+1;
  end
  disp (y);
```

Proof of Correctness: (Induction on the first argument ???)

If $[x = 0]$, then

$\forall y \in \mathbb{Z}:\ [\texttt{iterative(x,y)}] \stackrel{\text{Function Definition}}{\Rightarrow} [y = x + y \text{ gets printed}]$.

Now assume that for some $0 < x \in \mathbb{N}_0$:

$\forall y \in \mathbb{Z}:\ [\texttt{iterative(x,y)}] \Rightarrow [x + y \text{ gets printed}]$.

Then $[\texttt{iterative(x+1,y)}] \stackrel{\text{Function Definition}}{\Rightarrow} [\texttt{iterative(x+1,y)}]$

Useless.

We need new tools.

What is the problem?

What is the problem?

- No expression replacement rule anymore.

What is the problem?

- No expression replacement rule anymore.
- Assignments.

What is the problem?

- No expression replacement rule anymore.
- Assignments.
- While loop.

What is the problem?

- No expression replacement rule anymore.
- Assignments.
- While loop.
- Variables exist in different states during execution.

Adapting to the new situation:

$$\left[x \in \mathbb{N}_0 \;\wedge\; y \in \mathbb{Z}\right]$$

```
function iterative(x,y)
  while x > 0
    x = x-1;
    y = y+1;
  end
  disp (y);
```

[If the program terminates, the value of $x + y$ gets printed.]

Taking care of states:

$\left[A \in \mathbb{N}_0 \ \wedge \ B \in \mathbb{Z} \ \wedge \ \texttt{iterative(A,B)} \text{ is called.} \right]$

```
function iterative(x,y)
```

$\left[x = A \in \mathbb{N}_0 \ \wedge \ y = B \in \mathbb{Z} \right]$

```
   while x > 0
     x = x-1;
     y = y+1;
   end
   disp (y);
```

$\left[\text{If the program terminates, the value of } A + B \text{ gets printed.} \right]$

$\left[A \in \mathbb{N}_0 \ \wedge \ B \in \mathbb{Z} \ \wedge \ \texttt{iterative(A,B)} \text{ is called.}\right]$

```
function iterative(x,y)
```

$\left[x = A \in \mathbb{N}_0 \ \wedge \ y = B \in \mathbb{Z}\right]$

```
    while x > 0
      x = x-1;
      y = y+1;
    end
    disp (y);
```

$\left[\text{If the program terminates, the value of } A + B \text{ gets printed.}\right]$

Clearly:

$\left[A \in \mathbb{N}_0 \ \wedge\ B \in \mathbb{Z} \ \wedge\ \texttt{iterative(A,B)} \text{ is called.}\right]$

```
function iterative(x,y)
```

$\left[x = A \in \mathbb{N}_0 \ \wedge\ y = B \in \mathbb{Z}\right] \Rightarrow$

$\left[x + y = A + B \ \wedge\ x \geq 0\right]$

```
    while x > 0
      x = x-1;
      y = y+1;
    end
    disp (y);
```

[If the program terminates, the value of $A + B$ gets printed.]

$\left[A \in \mathbb{N}_0 \;\wedge\; B \in \mathbb{Z} \;\wedge\; \texttt{iterative(A,B)} \text{ is called.}\right]$

```
function iterative(x,y)
```

$\left[x = A \in \mathbb{N}_0 \;\wedge\; y = B \in \mathbb{Z}\right] \Rightarrow$

$\left[x + y = A + B \;\wedge\; x \geq 0\right]$

```
   while x > 0
```

$\left[\left[x + y = A + B \;\wedge\; x \geq 0\right] \;\wedge\; x > 0\right]$

```
      x = x-1;
      y = y+1;
   end
   disp (y);
```

$\left[\text{If the program terminates, the value of } A + B \text{ gets printed.}\right]$

We claim that this is a loop invariant

$$\left[A \in \mathbb{N}_0 \ \wedge \ B \in \mathbb{Z} \ \wedge \ \texttt{iterative(A,B)} \text{ is called.}\right]$$

```
function iterative(x,y)
```

$$\left[x = A \in \mathbb{N}_0 \ \wedge \ y = B \in \mathbb{Z}\right] \Rightarrow$$

$$\left[x + y = A + B \ \wedge \ x \geq 0\right]$$

```
while x > 0
```

$$\left[\left[x + y = A + B \ \wedge \ x \geq 0\right] \ \wedge \ x > 0\right]$$

```
x = x-1;
y = y+1;
```

$$\left[x + y = A + B \ \wedge \ x \geq 0\right]$$

```
end
disp (y);
```

$$\left[\text{If the program terminates, the value of } A + B \text{ gets printed.}\right]$$

$\left[A \in \mathbb{N}_0 \ \wedge \ B \in \mathbb{Z} \ \wedge \ \texttt{iterative(A,B)} \text{ is called.}\right]$

```
function iterative(x,y)
```

$\left[x = A \in \mathbb{N}_0 \ \wedge \ y = B \in \mathbb{Z}\right] \Rightarrow$

$\left[x + y = A + B \ \wedge \ x \geq 0\right]$

```
    while x > 0
```

$\left[\left[x + y = A + B \ \wedge \ x \geq 0\right] \ \wedge \ x > 0\right]$

```
        x = x-1;
```

$\left[x + (y + 1) = A + B \ \wedge \ x \geq 0\right]$

```
        y = y+1;
```

$\left[x + y = A + B \ \wedge \ x \geq 0\right]$

```
    end
    disp (y);
```

$\left[\text{If the program terminates, the value of } A + B \text{ gets printed.}\right]$

$$\left[A \in \mathbb{N}_0 \;\wedge\; B \in \mathbb{Z} \;\wedge\; \texttt{iterative(A,B)} \text{ is called.}\right]$$

```
function iterative(x,y)
```

$$\left[x = A \in \mathbb{N}_0 \;\wedge\; y = B \in \mathbb{Z}\right] \Rightarrow$$

$$\left[x + y = A + B \;\wedge\; x \geq 0\right]$$

```
   while x > 0
```

$$\left[\left[x + y = A + B \;\wedge\; x \geq 0\right] \;\wedge\; x > 0\right]$$

$$\left[(x-1) + (y+1) = A + B \;\wedge\; (x-1) \geq 0\right]$$

```
      x = x-1;
```

$$\left[x + (y+1) = A + B \;\wedge\; x \geq 0\right]$$

```
      y = y+1;
```

$$\left[x + y = A + B \;\wedge\; x \geq 0\right]$$

```
   end
   disp (y);
```

$$\left[\text{If the program terminates, the value of } A + B \text{ gets printed.}\right]$$

$\left[A \in \mathbb{N}_0 \ \wedge \ B \in \mathbb{Z} \ \wedge \ \texttt{iterative(A,B)} \text{ is called.}\right]$

```
function iterative(x,y)
```

$\left[x = A \in \mathbb{N}_0 \ \wedge \ y = B \in \mathbb{Z}\right] \Rightarrow$

$\left[x + y = A + B \ \wedge \ x \geq 0\right]$

```
   while x > 0
```

$\left[\left[x + y = A + B \ \wedge \ x \geq 0\right] \ \wedge \ x > 0\right]$

$\left[(x-1) + (y+1) = A + B \ \wedge \ (x-1) \geq 0\right]$

```
      x = x-1;
```

$\left[x + (y+1) = A + B \ \wedge \ x \geq 0\right]$

```
      y = y+1;
```

$\left[x + y = A + B \ \wedge \ x \geq 0\right]$

```
   end
   disp (y);
```

$\left[\text{If the program terminates, the value of } A + B \text{ gets printed.}\right]$

$\left[A \in \mathbb{N}_0 \ \wedge \ B \in \mathbb{Z} \ \wedge \ \texttt{iterative(A,B)} \text{ is called.}\right]$

```
function iterative(x,y)
```

$\left[x = A \in \mathbb{N}_0 \ \wedge \ y = B \in \mathbb{Z}\right] \Rightarrow$

$\left[x + y = A + B \ \wedge \ x \geq 0\right]$

```
    while x > 0
```

$\left[\left[x + y = A + B \ \wedge \ x \geq 0\right] \ \wedge \ x > 0\right] \Rightarrow$

$\left[(x-1) + (y+1) = A + B \ \wedge \ (x-1) \geq 0\right]$

```
        x = x-1;
```

$\left[x + (y+1) = A + B \ \wedge \ x \geq 0\right]$

```
        y = y+1;
```

$\left[x + y = A + B \ \wedge \ x \geq 0\right]$

```
    end
    disp (y);
```

$\left[\text{If the program terminates, the value of } A + B \text{ gets printed.}\right]$

$\left[A \in \mathbb{N}_0 \ \wedge \ B \in \mathbb{Z} \ \wedge \ \texttt{iterative(A,B)} \text{ is called.}\right]$

```
function iterative(x,y)
```

$\left[x = A \in \mathbb{N}_0 \ \wedge \ y = B \in \mathbb{Z}\right] \Rightarrow$

$\left[x + y = A + B \ \wedge \ x \geq 0\right]$

```
    while x > 0
```

$\left[\left[x + y = A + B \ \wedge \ x \geq 0\right] \ \wedge \ x > 0\right] \Rightarrow$

$\left[(x-1) + (y+1) = A + B \ \wedge \ (x-1) \geq 0\right]$

```
        x = x-1;
```

$\left[x + (y+1) = A + B \ \wedge \ x \geq 0\right]$

```
        y = y+1;
```

$\left[x + y = A + B \ \wedge \ x \geq 0\right]$

```
    end
    disp (y);
```

$\left[\text{If the program terminates, the value of } A + B \text{ gets printed.}\right]$

$\left[A \in \mathbb{N}_0 \ \wedge \ B \in \mathbb{Z} \ \wedge \ \texttt{iterative(A,B)} \text{ is called.}\right]$

```
function iterative(x,y)
```

$\left[x = A \in \mathbb{N}_0 \ \wedge \ y = B \in \mathbb{Z}\right] \Rightarrow$

$\left[x + y = A + B \ \wedge \ x \geq 0\right]$

```
    while x > 0
```

$\left[\left[x + y = A + B \ \wedge \ x \geq 0\right] \ \wedge \ x > 0\right] \Rightarrow$

$\left[(x-1) + (y+1) = A + B \ \wedge \ (x-1) \geq 0\right]$

```
        x = x-1;
```

$\left[x + (y+1) = A + B \ \wedge \ x \geq 0\right]$

```
        y = y+1;
```

$\left[x + y = A + B \ \wedge \ x \geq 0\right]$

```
    end
    disp (y);
```

$\left[\text{If the program terminates, the value of } A + B \text{ gets printed.}\right]$

$\left[A \in \mathbb{N}_0 \ \wedge \ B \in \mathbb{Z} \ \wedge \ \texttt{iterative(A,B)} \text{ is called.}\right]$

```
function iterative(x,y)
```

$\left[x = A \in \mathbb{N}_0 \ \wedge \ y = B \in \mathbb{Z}\right] \Rightarrow$

$\left[x + y = A + B \ \wedge \ x \geq 0\right]$

```
    while x > 0
```

$\left[\left[x + y = A + B \ \wedge \ x \geq 0\right] \ \wedge \ x > 0\right] \Rightarrow$

$\left[(x-1) + (y+1) = A + B \ \wedge \ (x-1) \geq 0\right]$

```
        x = x-1;
```

$\left[x + (y+1) = A + B \ \wedge \ x \geq 0\right]$

```
        y = y+1;
```

$\left[x + y = A + B \ \wedge \ x \geq 0\right]$

```
    end
```

$\left[\left[x + y = A + B \ \wedge \ x \geq 0\right] \ \wedge \ \neg\left[x > 0\right]\right]$

```
    disp (y);
```

$\left[\text{If the program terminates, the value of } A + B \text{ gets printed.}\right]$

$\left[A \in \mathbb{N}_0 \ \land \ B \in \mathbb{Z} \ \land \ \texttt{iterative(A,B)} \text{ is called.}\right]$

```
function iterative(x,y)
```

$\left[x = A \in \mathbb{N}_0 \ \land \ y = B \in \mathbb{Z}\right] \Rightarrow$

$\left[x + y = A + B \ \land \ x \geq 0\right]$

```
    while x > 0
```

$\left[\left[x + y = A + B \ \land \ x \geq 0\right] \ \land \ x > 0\right] \Rightarrow$

$\left[(x-1) + (y+1) = A + B \ \land \ (x-1) \geq 0\right]$

```
        x = x-1;
```

$\left[x + (y+1) = A + B \ \land \ x \geq 0\right]$

```
        y = y+1;
```

$\left[x + y = A + B \ \land \ x \geq 0\right]$

```
    end
```

$\left[\left[x + y = A + B \ \land \ x \geq 0\right] \ \land \ \neg\left[x > 0\right]\right] \Rightarrow$

```
    disp (y);
```

[If the program terminates, the value of $A + B$ gets printed.]

$\left[A \in \mathbb{N}_0 \ \wedge \ B \in \mathbb{Z} \ \wedge \ \texttt{iterative(A,B)} \text{ is called.}\right]$

```
function iterative(x,y)
```

$\left[x = A \in \mathbb{N}_0 \ \wedge \ y = B \in \mathbb{Z}\right] \Rightarrow$

$\left[x + y = A + B \ \wedge \ x \geq 0\right]$

```
    while x > 0
```

$\left[\left[x + y = A + B \ \wedge \ x \geq 0\right] \ \wedge \ x > 0\right] \Rightarrow$

$\left[(x-1) + (y+1) = A + B \ \wedge \ (x-1) \geq 0\right]$

```
        x = x-1;
```

$\left[x + (y+1) = A + B \ \wedge \ x \geq 0\right]$

```
        y = y+1;
```

$\left[x + y = A + B \ \wedge \ x \geq 0\right]$

```
    end
```

$\left[\left[x + y = A + B \ \wedge \ x \geq 0\right] \ \wedge \ \neg\left[x > 0\right]\right] \Rightarrow$

$\left[x + y = A + B \ \wedge \ x = 0\right] \Rightarrow$

```
    disp (y);
```

$\left[\text{If the program terminates, the value of } A + B \text{ gets printed.}\right]$

$\left[A \in \mathbb{N}_0 \;\wedge\; B \in \mathbb{Z} \;\wedge\; \texttt{iterative(A,B)} \text{ is called.}\right]$

```
function iterative(x,y)
```

$\left[x = A \in \mathbb{N}_0 \;\wedge\; y = B \in \mathbb{Z}\right] \Rightarrow$

$\left[x + y = A + B \;\wedge\; x \geq 0\right]$

```
    while x > 0
```

$\left[\left[x + y = A + B \;\wedge\; x \geq 0\right] \;\wedge\; x > 0\right] \Rightarrow$

$\left[(x-1) + (y+1) = A + B \;\wedge\; (x-1) \geq 0\right]$

```
        x = x-1;
```

$\left[x + (y+1) = A + B \;\wedge\; x \geq 0\right]$

```
        y = y+1;
```

$\left[x + y = A + B \;\wedge\; x \geq 0\right]$

```
    end
```

$\left[\left[x + y = A + B \;\wedge\; x \geq 0\right] \;\wedge\; \neg\left[x > 0\right]\right] \Rightarrow$

$\left[x + y = A + B \;\wedge\; x = 0\right] \Rightarrow \left[y = A + B\right]$

```
    disp (y);
```

$\left[\text{If the program terminates, the value of } A + B \text{ gets printed.}\right]$

$\Big[A \in \mathbb{N}_0 \ \wedge \ B \in \mathbb{Z} \ \wedge$ `iterative(A,B)` is called.$\Big]$

```
function iterative(x,y)
```

$\Big[x = A \in \mathbb{N}_0 \ \wedge \ y = B \in \mathbb{Z}\Big] \Rightarrow$

$\Big[x + y = A + B \ \wedge \ x \geq 0\Big]$

```
    while x > 0
```

$\Big[\big[x + y = A + B \ \wedge \ x \geq 0\big] \ \wedge \ x > 0\Big] \Rightarrow$

$\Big[(x-1) + (y+1) = A + B \ \wedge \ (x-1) \geq 0\Big]$

```
        x = x-1;
```

$\Big[x + (y+1) = A + B \ \wedge \ x \geq 0\Big]$

```
        y = y+1;
```

$\Big[x + y = A + B \ \wedge \ x \geq 0\Big]$

```
    end
```

$\Big[\big[x + y = A + B \ \wedge \ x \geq 0\big] \ \wedge \ \neg\big[x > 0\big]\Big] \Rightarrow$

$\Big[x + y = A + B \ \wedge \ x = 0\Big] \Rightarrow \Big[y = A + B\Big]$

```
    disp (y);
```

$\Big[$If the program terminates, the value of $A + B$ gets printed.$\Big]$ $\square$

What have we done? What will we do?

What have we done? What will we do?

What have we done? What will we do?

- What, exactly, did we proof, after all? And what not?

What have we done? What will we do?

- What, exactly, did we proof, after all? And what not?
- How can we codify what we have done and will have to do next time?

What have we done? What will we do?

- What, exactly, did we proof, after all? And what not?
- How can we codify what we have done and will have to do next time?
- What are the underlying rules of reasoning?

What we did proof:

What we did proof:

- Partial Semantic Correctness of the function `iterative(x,y)` with respect to some specification.

What we did proof:

- Partial Semantic Correctness of the function `iterative(x,y)` with respect to some specification.
- The specification was:

  $\left[A \in \mathbb{N}_0 \ \wedge \ B \in \mathbb{Z} \ \wedge \ \texttt{iterative(A,B)} \text{ is called.} \right]$

  PROGRAM

  $\left[\text{If the program terminates, the value of } A + B \text{ gets printed.} \right]$

What we did proof:

- Partial Semantic Correctness of the function `iterative(x,y)` with respect to some specification.
- The specification was:
  $\left[A \in \mathbb{N}_0 \ \wedge \ B \in \mathbb{Z} \ \wedge \ \texttt{iterative(A,B)} \text{ is called.} \right]$
  PROGRAM
  $\left[\text{If the program terminates, the value of } A + B \text{ gets printed.} \right]$
- Semantic denotes that we were concerned with the meaning of the program. We are not concerned with the syntax of the program.

Partial Correctness means:

There does not happen anything contradicting the specification.

In particular, for partial correctness it is allowed that:

In particular, for partial correctness it is allowed that:

- The program never terminates.

In particular, for partial correctness it is allowed that:

- The program never terminates.
- The program does terminate, the specification is fulfilled, and something not to be found in the specification happens in addition to that.

In particular, for partial correctness it is allowed that:

- The program never terminates.
- The program does terminate, the specification is fulfilled, and something not to be found in the specification happens in addition to that.

Yet we will be content with partial correctness and will go on to concern ourselves with only that.

In particular, for partial correctness it is allowed that:

- The program never terminates.
- The program does terminate, the specification is fulfilled, and something not to be found in the specification happens in addition to that.

Yet we will be content with partial correctness and will go on to concern ourselves with only that.

- Total correctness means:

  The program terminates,
  and there does not happen anything
  contradicting the specification.

In particular, for partial correctness it is allowed that:

- The program never terminates.
- The program does terminate, the specification is fulfilled, and something not to be found in the specification happens in addition to that.

Yet we will be content with partial correctness and will go on to concern ourselves with only that.

- Total correctness means:

  The program terminates,
  and there does not happen anything
  contradicting the specification.
- Proving that a program terminates can be hard.

Codification of what we did: The Hoare Rules

- C. A. R. Hoare 1969:

Codification of what we did: The Hoare Rules

- C. A. R. Hoare 1969:

An Axiomatic Basis for Computer Programming

C. A. R. HOARE
The Queen's University of Belfast, Northern Ireland*

In this paper an attempt is made to explore the logical foundations of computer programming by use of techniques which were first applied in the study of geometry and have later been extended to other branches of mathematics. This involves the elucidation of sets of axioms and rules of inference which can be used in proofs of the properties of computer programs.

-

Predicates

A predicate is a function from some set D to the set {true, false }:

$$P : D \rightarrow \{\texttt{true}, \texttt{false}\}$$

Strong and Weak

Strong and Weak

- By Definition
  $[A] \Rightarrow [B] \quad :\Leftrightarrow \quad \neg[A] \vee [B]$

Strong and Weak

- By Definition
  $[A] \Rightarrow [B] \quad :\Leftrightarrow \quad \neg[A] \vee [B]$
- The predicate [false] is the strongest of all:
  $\forall B: \quad \text{false} \Rightarrow [B] \quad \Leftrightarrow \quad \neg\text{false} \vee [B] \quad \Leftrightarrow$
  $\text{true} \vee [B] \quad \Leftrightarrow \quad \text{true}$

Strong and Weak

- By Definition
  $[A] \Rightarrow [B] \quad :\Leftrightarrow \quad \neg[A] \vee [B]$
- The predicate $[\text{false}]$ is the strongest of all:
  $\forall B: \quad \text{false} \Rightarrow [B] \quad \Leftrightarrow \quad \neg\text{false} \vee [B] \quad \Leftrightarrow \quad \text{true} \vee [B] \quad \Leftrightarrow \quad \text{true}$
- The predicate $[\text{true}]$ is the weakest of all:
  $\forall B: \quad \text{true} \Rightarrow [B] \quad \Leftrightarrow \quad \neg\text{true} \vee [B] \quad \Leftrightarrow \quad \text{false} \vee [B] \quad \Leftrightarrow \quad [B]$

Strong and Weak

- By Definition
  $[A] \Rightarrow [B] \quad :\Leftrightarrow \quad \neg[A] \vee [B]$
- The predicate $[\text{false}]$ is the strongest of all:
  $\forall B: \quad \text{false} \Rightarrow [B] \quad \Leftrightarrow \quad \neg\text{false} \vee [B] \quad \Leftrightarrow$
  $\text{true} \vee [B] \quad \Leftrightarrow \quad \text{true}$
- The predicate $[\text{true}]$ is the weakest of all:
  $\forall B: \quad \text{true} \Rightarrow [B] \quad \Leftrightarrow \quad \neg\text{true} \vee [B] \quad \Leftrightarrow$
  $\text{false} \vee [B] \quad \Leftrightarrow \quad [B]$
- Thus:
  $\text{false} \quad \Rightarrow \quad \cdots \quad \Rightarrow \quad \text{true}$

Strong and Weak

- By Definition
  $[A] \Rightarrow [B] \quad :\Leftrightarrow \quad \neg[A] \vee [B]$
- The predicate $[\text{false}]$ is the strongest of all:
  $\forall B: \quad \text{false} \Rightarrow [B] \quad \Leftrightarrow \quad \neg\text{false} \vee [B] \quad \Leftrightarrow$
  $\text{true} \vee [B] \quad \Leftrightarrow \quad \text{true}$
- The predicate $[\text{true}]$ is the weakest of all:
  $\forall B: \quad \text{true} \Rightarrow [B] \quad \Leftrightarrow \quad \neg\text{true} \vee [B] \quad \Leftrightarrow$
  $\text{false} \vee [B] \quad \Leftrightarrow \quad [B]$
- Thus:
  $\text{false} \quad \Rightarrow \quad \cdots \quad \Rightarrow \quad \text{true}$
- We define:
  $[A]$ is stronger than $[B] \quad :\Leftrightarrow \quad [A] \Rightarrow [B]$

Strong and Weak

- By Definition
  $[A] \Rightarrow [B] \quad :\Leftrightarrow \quad \neg[A] \vee [B]$
- The predicate $[\text{false}]$ is the strongest of all:
  $\forall B : \quad \text{false} \Rightarrow [B] \quad \Leftrightarrow \quad \neg\text{false} \vee [B] \quad \Leftrightarrow$
  $\text{true} \vee [B] \quad \Leftrightarrow \quad \text{true}$
- The predicate $[\text{true}]$ is the weakest of all:
  $\forall B : \quad \text{true} \Rightarrow [B] \quad \Leftrightarrow \quad \neg\text{true} \vee [B] \quad \Leftrightarrow$
  $\text{false} \vee [B] \quad \Leftrightarrow \quad [B]$
- Thus:
  $\text{false} \quad \Rightarrow \quad \cdots \quad \Rightarrow \quad \text{true}$
- We define:
  $[A]$ is stronger than $[B] \quad :\Leftrightarrow \quad [A] \Rightarrow [B]$

The Mathematical Structure of the Hoare Rules

- Essential ingredient: Hoare Triple: $\left[\, [P] \;\mathsf{S}\; [Q] \,\right]$

The Mathematical Structure of the Hoare Rules

- Essential ingredient: Hoare Triple: $\left[[P] \ \mathsf{S} \ [Q] \right]$
- A Hoare Triple is itself a predicate
  $H : \{\mathtt{true}, \mathtt{false}\} \times \mathsf{M} \times \{\mathtt{true}, \mathtt{false}\} \longrightarrow \{\mathtt{true}, \mathtt{false}\},$

The Mathematical Structure of the Hoare Rules

- Essential ingredient: Hoare Triple: $\left[[P] \; \mathsf{S} \; [Q] \right]$
- A Hoare Triple is itself a predicate
  $H : \{\texttt{true}, \texttt{false}\} \times \mathsf{M} \times \{\texttt{true}, \texttt{false}\} \longrightarrow \{\texttt{true}, \texttt{false}\}$,
- where the predicates $[P]$ and $[Q]$ provide the first and third argument, and

The Mathematical Structure of the Hoare Rules

- Essential ingredient: Hoare Triple: $\Big[[P]\ \mathsf{S}\ [Q] \Big]$
- A Hoare Triple is itself a predicate
  $H : \{\mathtt{true}, \mathtt{false}\} \times \mathsf{M} \times \{\mathtt{true}, \mathtt{false}\} \longrightarrow \{\mathtt{true}, \mathtt{false}\}$,
- where the predicates $[P]$ and $[Q]$ provide the first and third argument, and
- the set M denotes the set of all syntactically correct programs in some programming language,

The Mathematical Structure of the Hoare Rules

- Essential ingredient: Hoare Triple: $\big[[P] \; \mathsf{S} \; [Q] \big]$
- A Hoare Triple is itself a predicate $H : \{\texttt{true}, \texttt{false}\} \times \mathsf{M} \times \{\texttt{true}, \texttt{false}\} \longrightarrow \{\texttt{true}, \texttt{false}\}$,
- where the predicates $[P]$ and $[Q]$ provide the first and third argument, and
- the set M denotes the set of all syntactically correct programs in some programming language,
- and the value of $\big[[P] \; \mathsf{S} \; [Q] \big]$ is defined as follows:

When is a Hoare Triple true, when is it false?

$$\Big[[P] \; \mathsf{S} \; [Q] \Big] = \texttt{true}$$

$:\Longleftrightarrow$

If the predicate $[P]$ is true immediately before execution of the program $\mathsf{S} \in M$, then immediately after S has terminated, the predicate $[Q]$ is true.

The rules often take the following form:

The rules often take the following form:

- $$\frac{\text{A formula involving predicates and Hoare Triples.}}{\text{A Hoare Triple whose program fragment comprises the fragments used in the Hoare Triples above.}}$$

The rules often take the following form:

- $$\frac{\text{A formula} \atop \text{involving predicates and Hoare Triples.}}{\text{A Hoare Triple whose program fragment} \atop \text{comprises the fragments used in the Hoare Triples above.}}$$
- Example:

$$\frac{\Big[[P \wedge B] \quad S \quad [P] \Big]}{\Big[[P] \quad \underline{\texttt{while}}\ B\ \underline{\texttt{do}}\ S\ \underline{\texttt{end}} \quad [P \wedge \neg B] \Big]}$$

The rules often take the following form:

- $$\frac{\text{A formula involving predicates and Hoare Triples.}}{\text{A Hoare Triple whose program fragment comprises the fragments used in the Hoare Triples above.}}$$
- Example:

$$\frac{\Big[[P \wedge B] \quad S \quad [P] \Big]}{\Big[[P] \quad \underbrace{\texttt{while } B \texttt{ do } S \texttt{ end}}_{\texttt{uses } S \texttt{ and is therefore larger than it.}} \quad [P \wedge \neg B] \Big]}$$

The Rules

Rule 0:

$$\frac{\left[\text{true}\right]}{\left[\left[P\right] \quad \underline{\%}\ \texttt{noOperation} \quad \left[P\right]\right]}$$

Rule 1: *Axiom of Assignment*

$$\frac{\left[\text{true}\right]}{\left[\left[P_{E\ \text{instead of}\ x}\right] \quad \texttt{x = E;} \quad \left[P\right]\right]}$$

Rule 1: *Axiom of Assignment*

$$\frac{\left[\text{true}\right]}{\left[\ \left[P_{E/x}\right]\quad \texttt{x = E;}\quad \left[P\right]\ \right]}$$

Rule 1: *Axiom of Assignment*

$$\frac{\left[\text{true}\right]}{\left[\left[P_{E/x}\right] \quad \texttt{x = E;} \quad \left[P\right]\right]}$$

Rule 2: *Rule of Consequence*

$$\frac{\left[\left[\left[\widetilde{P}\right] \Rightarrow \left[P\right]\right] \quad \wedge \quad \left[\left[P\right] \text{ S } \left[Q\right]\right] \quad \wedge \quad \left[\left[Q\right] \Rightarrow \left[\widetilde{Q}\right]\right]\right]}{\left[\left[\widetilde{P}\right] \text{ S } \left[\widetilde{Q}\right]\right]}$$

Rule 3: *Rule of Composition*

$$\frac{\Big[\big[[P]\ \mathtt{S}\ [Q] \big] \quad \wedge \quad \big[[Q]\ \mathtt{T}\ [R] \big] \Big]}{\big[[P] \quad \mathtt{S}\,;\mathtt{T} \quad [R] \big]}$$

Rule 4: *Rule of Iteration*

$$\frac{\Big[[P \wedge B] \quad \mathrm{S} \quad [P] \Big]}{\Big[[P] \quad \underline{\texttt{while}}\ B\ \underline{\texttt{do}}\ \mathrm{S}\ \underline{\texttt{end}} \quad [P \wedge \neg B] \Big]}$$

Rule 5: *Rule of Conditional Branching*

$$\frac{\Big[\big[[P \wedge B] \quad \mathtt{S} \quad [Q]\big] \;\wedge\; \big[[P \wedge \neg B] \quad \mathtt{T} \quad [Q]\big]\Big]}{\big[[P] \quad \underline{\mathtt{if}}\ B\ \underline{\mathtt{do}}\ \mathtt{S}\ \underline{\mathtt{else}}\ \underline{\mathtt{do}}\ \mathtt{T}\ \underline{\mathtt{end}} \quad [Q]\big]}$$

Applications

```
x = x + y;
y = x - y;
x = x - y;
```

$$\left[x = A \;\wedge\; y = B \right]$$

```
x = x + y;
y = x - y;
x = x - y;
```

Swapping without moving...

$$\left[x = A \ \wedge \ y = B\right]$$

```
x = x + y;
y = x - y;
x = x - y;
```

$$\left[x = B \ \wedge \ y = A\right]$$

$$\left[x = A \;\wedge\; y = B\right]$$

```
x = x + y;
y = x - y;
x = x - y;
```

$$\left[x = B \;\wedge\; y = A\right]$$

Annotating the Program with Assertions

$$\left[x = A \ \wedge \ y = B\right]$$

$$\left[\ \right]$$

```
x = x + y;
```

$$\left[\ \right]$$

```
y = x - y;
```

$$\left[\ \right]$$

```
x = x - y;
```

$$\left[x = B \ \wedge \ y = A\right]$$

$$\left[x = A \ \land \ y = B\right]$$

$$\left[\ \right]$$

```
x = x + y;
```

$$\left[\ \right]$$

```
y = x - y;
```

$$\left[\ \right]$$

```
x = x - y;
```

$$\left[x = B \ \land \ y = A\right]$$

$$\left[x = A \ \wedge \ y = B\right]$$

$$\left[\ \right]$$

```
x = x + y;
```

$$\left[\ \right]$$

```
y = x - y;
```

$$\left[\ \right]$$

$\mathtt{x}_{\mathsf{after}} = \mathtt{x}_{\mathsf{before}} - \mathtt{y}_{\mathsf{before}}$;

$$\left[x_{\mathsf{after}} = B \ \wedge \ y_{\mathsf{after}} = A\right]$$

$$\left[x = A \;\wedge\; y = B\right]$$

$$[\;]$$

```
x = x + y;
```

$$[\;]$$

```
y = x - y;
```

$$[\;]$$

$$\mathtt{x}_{\text{after}} = \mathtt{x}_{\text{before}} - \mathtt{y}_{\text{before}}\,; \qquad \mathtt{y}_{\text{after}} = \mathtt{y}_{\text{before}}\,;$$

$$\left[x_{\text{after}} = B \;\wedge\; y_{\text{after}} = A\right]$$

$$\left[x = A \;\wedge\; y = B\right]$$

$$\left[\;\right]$$

```
x = x + y;
```

$$\left[\;\right]$$

```
y = x - y;
```

$$\left[x_{\text{before}} - y_{\text{before}} = B \;\wedge\; y_{\text{before}} = A\right]$$

$$\mathtt{x}_{\text{after}} = \mathtt{x}_{\text{before}} - \mathtt{y}_{\text{before}}\,; \qquad \mathtt{y}_{\text{after}} = \mathtt{y}_{\text{before}}\,;$$

$$\left[x_{\text{after}} = B \;\wedge\; y_{\text{after}} = A\right]$$

$$\left[x = A \ \wedge \ y = B\right]$$

$$\left[\ \right]$$

```
x = x + y;
```

$$\left[\ \right]$$

```
y = x - y;
```

$$\left[x - y = B \ \wedge \ y = A\right]$$

```
x = x - y;
```

$$\left[x = B \ \wedge \ y = A\right]$$

$$\left[x = A \ \wedge \ y = B\right]$$

$$\left[\ \right]$$

```
x = x + y;
```

$$\left[\ \right]$$

```
y = x - y;
```

$$\left[x - y = B \ \wedge \ y = A\right]$$

```
x = x - y;
```

$$\left[x = B \ \wedge \ y = A\right]$$

$$\left[x = A \ \wedge \ y = B\right]$$

$$\left[\right]$$

```
x = x + y;
```

$$\left[x - (x - y) = B \ \wedge \ x - y = A\right]$$

```
y = x - y;
```

$$\left[x - y = B \ \wedge \ y = A\right]$$

```
x = x - y;
```

$$\left[x = B \ \wedge \ y = A\right]$$

$$\left[x = A \;\wedge\; y = B\right]$$

$$\left[\;\right]$$

```
x = x + y;
```

$$\left[x - (x - y) = B \;\wedge\; x - y = A\right]$$

```
y = x - y;
```

$$\left[x - y = B \;\wedge\; y = A\right]$$

```
x = x - y;
```

$$\left[x = B \;\wedge\; y = A\right]$$

$$\left[x = A \ \land \ y = B\right]$$

$$\left[\ \right]$$

```
x = x + y;
```

$$\left[x - (x - y) = B \ \land \ x - y = A\right]$$

```
y = x - y;
```

$$\left[x - y = B \ \land \ y = A\right]$$

```
x = x - y;
```

$$\left[x = B \ \land \ y = A\right]$$

$$\left[x = A \ \wedge \ y = B\right]$$

$$\left[(x+y) - ((x+y) - y) = B \ \wedge \ (x+y) - y = A\right]$$

```
x = x + y;
```

$$\left[x - (x - y) = B \ \wedge \ x - y = A\right]$$

```
y = x - y;
```

$$\left[x - y = B \ \wedge \ y = A\right]$$

```
x = x - y;
```

$$\left[x = B \ \wedge \ y = A\right]$$

$$\Big[x = A \ \wedge \ y = B\Big]$$

$$\Big[(x + y) - ((x + y) - y) = B \ \wedge \ (x + y) - y = A\Big]$$

```
x = x + y;
```

$$\Big[x - (x - y) = B \ \wedge \ x - y = A\Big]$$

```
y = x - y;
```

$$\Big[x - y = B \ \wedge \ y = A\Big]$$

```
x = x - y;
```

$$\Big[x = B \ \wedge \ y = A\Big]$$

$$\left[x = A \ \wedge \ y = B\right]$$
$$\left[(x+y) - ((x+y) - y) = B \ \wedge \ (x+y) - y = A\right]$$

```
x = x + y;
```

$$\left[x - (x - y) = B \ \wedge \ x - y = A\right]$$

```
y = x - y;
```

$$\left[x - y = B \ \wedge \ y = A\right]$$

```
x = x - y;
```

$$\left[x = B \ \wedge \ y = A\right]$$

$$\left[x = A \ \wedge \ y = B\right] \Longrightarrow$$
$$\left[(x+y) - ((x+y) - y) = B \ \wedge \ (x+y) - y = A\right]$$

```
x = x + y;
```
$$\left[x - (x - y) = B \ \wedge \ x - y = A\right]$$
```
y = x - y;
```
$$\left[x - y = B \ \wedge \ y = A\right]$$
```
x = x - y;
```

$$\left[x = B \ \wedge \ y = A\right]$$

$$\left[x = A \ \wedge \ y = B\right] \Longrightarrow$$
$$\left[(x + y) - ((x + y) - y) = B \ \wedge \ (x + y) - y = A\right]$$

```
x = x + y;
```

$$\left[x - (x - y) = B \ \wedge \ x - y = A\right]$$

```
y = x - y;
```

$$\left[x - y = B \ \wedge \ y = A\right]$$

```
x = x - y;
```

$$\left[x = B \ \wedge \ y = A\right]$$

Using the Hoare Rules

$$\left[x = A \ \wedge \ y = B\right] \implies$$
$$\left[(x+y) - ((x+y) - y) = B \ \wedge \ (x+y) - y = A\right]$$

```
x = x + y;
```
$$\left[x - (x - y) = B \ \wedge \ x - y = A\right]$$
```
y = x - y;
```
$$\left[x - y = B \ \wedge \ y = A\right]$$
```
x = x - y;
```

$$\left[x = B \ \wedge \ y = A\right]$$

Using the Hoare Rules

$$\left[x = A \wedge y = B\right] \Longrightarrow$$
$$\left[(x+y) - ((x+y) - y) = B \wedge (x+y) - y = A\right]$$

```
x = x + y;
```

$$\left[x - (x - y) = B \wedge x - y = A\right]$$

```
y = x - y;
```

$$\left[x - y = B \wedge y = A\right]$$

```
x = x - y;
```

$$\left[x = B \wedge y = A\right]$$

Using the Hoare Rules

$$\left[x = A \wedge y = B\right] \Longrightarrow$$
$$\left[(x+y) - ((x+y) - y) = B \wedge (x+y) - y = A\right]$$

```
x = x + y;
```

$$\left[x - (x - y) = B \wedge x - y = A\right]$$

```
y = x - y;
```

$$\left[x - y = B \wedge y = A\right]$$

We appeal to Rule 2: *Rule of Consequence*

```
x = x - y;
```

$$\frac{\left[[\tilde{P}] \Rightarrow [P] \wedge [P]\ \mathsf{S}\ [Q] \wedge [Q] \Rightarrow [\tilde{Q}]\right]}{\left[[\tilde{P}]\ \mathsf{S}\ [\tilde{Q}]\right]}$$

$$\left[x = B \wedge y = A\right]$$

Using the Hoare Rules

$$\left[x = A \;\wedge\; y = B\right] \Longrightarrow$$
$$\left[(x+y) - ((x+y) - y) = B \;\wedge\; (x+y) - y = A\right]$$

```
  x = x + y;
```

$$\left[x - (x - y) = B \;\wedge\; x - y = A\right]$$

```
  y = x - y;
```

$$\left[x - y = B \;\wedge\; y = A\right]$$

We analyze its constituent parts:

```
  x = x - y;
```

$$\frac{\left[[\widetilde{P}] \Rightarrow [P] \;\;\wedge\;\; [P]\,\mathsf{S}\,[Q] \;\;\wedge\;\; [Q] \Rightarrow [\widetilde{Q}]\right]}{\left[[\widetilde{P}]\,\mathsf{S}\,[\widetilde{Q}]\right]}$$

$$\left[x = B \;\wedge\; y = A\right]$$

Using the Hoare Rules

$$\left[x = A \ \wedge \ y = B\right] \Longrightarrow$$
$$\left[(x+y) - ((x+y) - y) = B \ \wedge \ (x+y) - y = A\right]$$

```
x = x + y;
```

$$\left[x - (x - y) = B \ \wedge \ x - y = A\right]$$

```
y = x - y;
```

$$\left[x - y = B \ \wedge \ y = A\right]$$

Here the rule is applicable:

```
x = x - y;
```

$$\frac{\left[[\tilde{P}] \Rightarrow [P] \quad \wedge \quad [P] \ \mathcal{S} \ [Q] \quad \wedge \quad [Q] \Rightarrow [\tilde{Q}]\right]}{\left[[\tilde{P}] \ \mathcal{S} \ [\tilde{Q}]\right]}$$

$$\left[x = B \ \wedge \ y = A\right]$$

Using the Hoare Rules

$$\left[x = A \ \wedge \ y = B\right] \Longrightarrow$$
$$\left[(x+y) - ((x+y) - y) = B \ \wedge \ (x+y) - y = A\right]$$

```
x = x + y;
```

$$\left[x - (x - y) = B \ \wedge \ x - y = A\right]$$

```
y = x - y;
```

$$\left[x - y = B \ \wedge \ y = A\right]$$

```
x = x - y;
```

We analyze its constituent parts:

$$\frac{\left[[\tilde{P}] \Rightarrow [P] \ \wedge \ [P] \, S \, [Q] \ \wedge \ [Q] \Rightarrow [\tilde{Q}]\right]}{\left[[\tilde{P}] \, S \, [\tilde{Q}]\right]}$$

$$\left[x = B \ \wedge \ y = A\right]$$

Using the Hoare Rules

$$\left[x = A \ \wedge \ y = B\right] \Longrightarrow$$
$$\left[(x+y) - ((x+y) - y) = B \ \wedge \ (x+y) - y = A\right]$$

```
x = x + y;
```

$$\left[x - (x - y) = B \ \wedge \ x - y = A\right]$$

```
y = x - y;
```

$$\left[x - y = B \ \wedge \ y = A\right]$$

We appeal to Rule 1: *Axiom of Assignment*

```
x = x - y;
```

$$\frac{\left[\left[\tilde{P}\right] \Rightarrow \left[P\right] \ \wedge \ \left[P\right] \, \mathrm{S} \, \left[Q\right] \ \wedge \ \left[Q\right] \Rightarrow \left[\tilde{Q}\right]\right]}{\left[\left[\tilde{P}\right] \, \mathrm{S} \, \left[\tilde{Q}\right]\right]}$$

$$\left[x = B \ \wedge \ y = A\right]$$

Using the Hoare Rules

$$\left[x = A \wedge y = B\right] \Longrightarrow$$
$$\left[(x+y) - ((x+y) - y) = B \wedge (x+y) - y = A\right]$$

```
x = x + y;
```

$$\left[x - (x-y) = B \wedge x - y = A\right]$$

```
y = x - y;
```

$$\left[x - y = B \wedge y = A\right]$$

Here we have to see an implication.

```
x = x - y;
```

$$\frac{\left[[\tilde{P}] \Rightarrow [P] \;\wedge\; [P]\, \mathsf{S}\, [Q] \;\wedge\; [Q] \Rightarrow [\tilde{Q}]\right]}{\left[[\tilde{P}]\, \mathsf{S}\, [\tilde{Q}]\right]}$$

$$\left[x = B \wedge y = A\right]$$

Using the Hoare Rules

$$\left[x = A \ \wedge \ y = B\right] \Longrightarrow$$
$$\left[(x+y) - ((x+y) - y) = B \ \wedge \ (x+y) - y = A\right]$$

```
x = x + y;
```

$$\left[x - (x - y) = B \ \wedge \ x - y = A\right]$$

```
y = x - y;
```

$$\left[x - y = B \ \wedge \ y = A\right]$$

And there actually is one; let $\widetilde{Q} = Q$.

```
x = x - y;
```

$$\frac{\left[[\widetilde{P}] \Rightarrow [P] \ \wedge \ [P] \ \mathsf{S} \ [Q] \ \wedge \ [Q] \Rightarrow [Q]\right]}{\left[[\widetilde{P}] \ \mathsf{S} \ [Q]\right]}$$

$$\left[x = B \ \wedge \ y = A\right]$$

Using the Hoare Rules

$$[x = A \wedge y = B] \implies$$
$$[(x + y) - ((x + y) - y) = B \wedge (x + y) - y = A]$$

```
x = x + y;
```

$$[x - (x - y) = B \wedge x - y = A]$$

```
y = x - y;
```

$$[x - y = B \wedge y = A]$$

Thus ...

```
x = x - y;
```

$$\frac{[[\widetilde{P}] \Rightarrow [P] \quad \wedge \quad [P]\ S\ [Q] \quad \wedge \quad [Q] \Rightarrow [\widetilde{Q}]]}{[[\widetilde{P}]\ S\ [\widetilde{Q}]]}$$

$$[x = B \wedge y = A]$$

Using the Hoare Rules

$$\left[x = A \wedge y = B\right] \implies$$
$$\left[(x+y) - ((x+y) - y) = B \wedge (x+y) - y = A\right]$$

```
x = x + y;
```

$$\left[x - (x - y) = B \wedge x - y = A\right]$$

```
y = x - y;
```

$$\left[x - y = B \wedge y = A\right]$$

Thus ...

```
x = x - y;
```

$$\frac{\left[[\widetilde{P}] \Rightarrow [P] \;\wedge\; [P]\,\mathbf{S}\,[Q] \;\wedge\; [Q] \Rightarrow [\widetilde{Q}]\right]}{\left[[\widetilde{P}]\,\mathbf{S}\,[\widetilde{Q}]\right]}$$

$$\left[x = B \wedge y = A\right]$$

Using the Hoare Rules

$$\left[x = A \;\wedge\; y = B\right] \;\Longrightarrow$$
$$\left[(x+y) - ((x+y) - y) = B \;\wedge\; (x+y) - y = A\right]$$

```
x = x + y;
```

$$\left[x - (x - y) = B \;\wedge\; x - y = A\right]$$

```
y = x - y;
```

$$\left[x - y = B \;\wedge\; y = A\right]$$

And one step in the program is proved.

```
x = x - y;
```

$$\frac{\left[[\tilde{P}] \Rightarrow [P] \;\wedge\; [P]\; S\; [Q] \;\wedge\; [Q] \Rightarrow [\tilde{Q}]\right]}{\left[[\tilde{P}]\; S\; [\tilde{Q}]\right]}$$

$$\left[x = B \;\wedge\; y = A\right]$$

Using the Hoare Rules

$$\left[x = A \land y = B\right] \implies$$
$$\left[(x+y) - ((x+y) - y) = B \land (x+y) - y = A\right]$$

```
x = x + y;
```

$$\left[x - (x - y) = B \land x - y = A\right]$$

```
y = x - y;
```

$$\left[x - y = B \land y = A\right]$$

We could go on like that ...

```
x = x - y;
```

$$\frac{\left[[\widetilde{P}] \Rightarrow [P] \;\land\; [P]\, \mathsf{S}\, [Q] \;\land\; [Q] \Rightarrow [\widetilde{Q}] \right]}{\left[[\widetilde{P}]\, \mathsf{S}\, [\widetilde{Q}] \right]}$$

$$\left[x = B \land y = A\right]$$

Finding suitable invariants may be not that easy.

Symmetry helps.

Symmetry helps.

```
function result = f(x,y)
```

Symmetry helps.

```
function result = f(x,y)
  while x < y || y < x
```

Symmetry helps.

```
function result = f(x,y)
  while x < y || y < x
    if x < y
```

Symmetry helps.

```
function result = f(x,y)
  while x < y || y < x
    if x < y
      y = y-x;
    else
```

Symmetry helps.

```
function result = f(x,y)
  while x < y || y < x
    if x < y
      y = y-x;
    else
      x = x-y;
    end
  end
  result =
```

Symmetry helps.

```
function result = f(x,y)
  while x < y || y < x
    if y < x
      x = x-y;
    else
      y = y-x;
    end
  end
  result =
```

What shall be our result?

```
function result = f(x,y)
  while x < y || y < x
    if x < y
      y = y-x;
    else
      x = x-y;
    end
  end
  result =
```

What shall be our result?

```
function result = f(x,y)
  while x < y || y < x
    if x < y
      y = y-x;
    else
      x = x-y;
    end
  end
  result =
```

▸ Skip proof that x=y=gcd(A,B)

Proof that $x = y = \gcd(A, B)$

```
function result = f(x,y)
  while x < y || y < x
    if x < y
      y = y-x;
    else
      x = x-y;
    end
  end
```

Proof that $x = y = \gcd(A, B)$

$$\Big[[\mathbb{N} \ni A \geq 1] \ \wedge \ [\mathbb{N} \ni B \geq 1] \ \wedge \ [\texttt{f(A,B)} \text{ is called.}] \Big]$$

```
function result = f(x,y)
  while x < y || y < x
    if x < y
      y = y-x;
    else
      x = x-y;
    end
  end
```

$$\Big[\text{If the program terminates, } \ x = y = \gcd(A, B) \Big]$$

Proof that $x = y = \gcd(A, B)$

$\Big[[\mathbb{N} \ni A \geq 1] \wedge [\mathbb{N} \ni B \geq 1] \wedge [\texttt{f(A,B)} \text{ is called.}] \Big]$

```
function result = f(x,y)
```

$\Big[[x = A] \wedge [y = B] \wedge [\mathbb{N} \ni x \geq 1] \wedge [\mathbb{N} \ni y \geq 1] \Big]$

```
    while x < y || y < x
      if x < y
        y = y-x;
      else
        x = x-y;
      end
    end
```

$\Big[\text{If the program terminates, } x = y = \gcd(A, B) \Big]$

Proof that $x = y = \gcd(A, B)$

$\Big[[\mathbb{N} \ni A \geq 1] \;\wedge\; [\mathbb{N} \ni B \geq 1] \;\wedge\; [\texttt{f(A,B)} \text{ is called.}]\Big]$

```
function result = f(x,y)
```

$\Big[[x = A] \;\wedge\; [y = B] \;\wedge\; [\mathbb{N} \ni x \geq 1] \;\wedge\; [\mathbb{N} \ni y \geq 1]\Big] \Rightarrow$

$\Big[[\gcd(x, y) = \gcd(A, B)] \;\wedge\; [\mathbb{N} \ni x \geq 1] \;\wedge\; [\mathbb{N} \ni y \geq 1]\Big]$

```
  while x < y || y < x
    if x < y
      y = y-x;
    else
      x = x-y;
    end
  end
```

$\Big[\text{If the program terminates, } \; x = y = \gcd(A, B)\Big]$

Proof that $x = y = \gcd(A, B)$

$\Big[[\mathbb{N} \ni A \geq 1] \ \wedge\ [\mathbb{N} \ni B \geq 1] \ \wedge\ [\texttt{f(A,B)} \text{ is called.}]\Big]$

```
 function result = f(x,y)
```

$\Big[[x = A] \ \wedge\ [y = B] \ \wedge\ [\mathbb{N} \ni x \geq 1] \ \wedge\ [\mathbb{N} \ni y \geq 1]\Big] \Rightarrow$

$\Big[[\gcd(x, y) = \gcd(A, B)] \ \wedge\ [\mathbb{N} \ni x \geq 1] \ \wedge\ [\mathbb{N} \ni y \geq 1]\Big]$

```
   while x < y || y < x
```

$\Big[[\gcd(x, y) = \gcd(A, B)] \ \wedge\ [x \neq y] \ \wedge\ [\mathbb{N} \ni x \geq 1] \ \wedge\ [\mathbb{N} \ni y \geq 1]\Big]$

```
     if x < y
       y = y-x;
     else
       x = x-y;
     end
```

$\Big[[\gcd(x, y) = \gcd(A, B)] \ \wedge\ [\mathbb{N} \ni x \geq 1] \ \wedge\ [\mathbb{N} \ni y \geq 1]\Big]$

```
   end
```

$\Big[[\gcd(x, y) = \gcd(A, B)] \ \wedge\ \neg[x \neq y] \ \wedge\ [\mathbb{N} \ni x \geq 1] \ \wedge\ [\mathbb{N} \ni y \geq 1]\Big]$

$\Big[\text{If the program terminates, } \ x = y = \gcd(A, B)\Big]$

Proof that $x = y = \gcd(A, B)$

$\Big[[\mathbb{N} \ni A \geq 1] \;\wedge\; [\mathbb{N} \ni B \geq 1] \;\wedge\; [\texttt{f(A,B)} \text{ is called.}]\Big]$

```
function result = f(x,y)
```

$\Big[[x = A] \;\wedge\; [y = B] \;\wedge\; [\mathbb{N} \ni x \geq 1] \;\wedge\; [\mathbb{N} \ni y \geq 1]\Big] \Rightarrow$

$\Big[[\gcd(x, y) = \gcd(A, B)] \;\wedge\; [\mathbb{N} \ni x \geq 1] \;\wedge\; [\mathbb{N} \ni y \geq 1]\Big]$

```
  while x < y || y < x
```

$\Big[[\gcd(x, y) = \gcd(A, B)] \;\wedge\; [x \neq y] \;\wedge\; [\mathbb{N} \ni x \geq 1] \;\wedge\; [\mathbb{N} \ni y \geq 1]\Big]$

```
    if x < y
      y = y-x;
    else
      x = x-y;
    end
```

$\Big[[\gcd(x, y) = \gcd(A, B)] \;\wedge\; [\mathbb{N} \ni x \geq 1] \;\wedge\; [\mathbb{N} \ni y \geq 1]\Big]$

```
  end
```

$\Big[[\gcd(x, y) = \gcd(A, B)] \;\wedge\; \neg[x \neq y] \;\wedge\; [\mathbb{N} \ni x \geq 1] \;\wedge\; [\mathbb{N} \ni y \geq 1]\Big]$

$\Big[\text{If the program terminates, } \; x = y = \gcd(A, B)\Big]$

Proof that $x = y = \gcd(A, B)$

$$\Big[[\mathbb{N} \ni A \geq 1] \;\wedge\; [\mathbb{N} \ni B \geq 1] \;\wedge\; [\texttt{f(A,B)} \text{ is called.}]\Big]$$

```
function result = f(x,y)
```

$$\Big[[x = A] \;\wedge\; [y = B] \;\wedge\; [\mathbb{N} \ni x \geq 1] \;\wedge\; [\mathbb{N} \ni y \geq 1]\Big] \Rightarrow$$

$$\Big[[\gcd(x,y) = \gcd(A,B)] \;\wedge\; [\mathbb{N} \ni x \geq 1] \;\wedge\; [\mathbb{N} \ni y \geq 1]\Big]$$

```
  while x < y || y < x
```

$$\Big[[\gcd(x,y) = \gcd(A,B)] \;\wedge\; [x \neq y] \;\wedge\; [\mathbb{N} \ni x \geq 1] \;\wedge\; [\mathbb{N} \ni y \geq 1]\Big]$$

```
    if x < y
      y = y-x;
    else
      x = x-y;
    end
```

$$\Big[[\gcd(x,y) = \gcd(A,B)] \;\wedge\; [\mathbb{N} \ni x \geq 1] \;\wedge\; [\mathbb{N} \ni y \geq 1]\Big]$$

```
  end
```

$$\Big[[\gcd(x,y) = \gcd(A,B)] \;\wedge\; \neg[x \neq y] \;\wedge\; [\mathbb{N} \ni x \geq 1] \;\wedge\; [\mathbb{N} \ni y \geq 1]\Big]$$

$$\Big[\text{If the program terminates, } \; x = y = \gcd(A,B)\Big]$$

Proof that $x = y = \gcd(A, B)$

$$\Big[[\mathbb{N} \ni A \geq 1] \wedge [\mathbb{N} \ni B \geq 1] \wedge [\texttt{f(A,B)} \text{ is called.}]\Big]$$

```
function result = f(x,y)
```

$$\Big[[x = A] \wedge [y = B] \wedge [\mathbb{N} \ni x \geq 1] \wedge [\mathbb{N} \ni y \geq 1]\Big] \Rightarrow$$

$$\Big[[\gcd(x, y) = \gcd(A, B)] \wedge [\mathbb{N} \ni x \geq 1] \wedge [\mathbb{N} \ni y \geq 1]\Big]$$

```
  while x < y || y < x
```

$$\Big[[\gcd(x, y) = \gcd(A, B)] \wedge [x \neq y] \wedge [\mathbb{N} \ni x \geq 1] \wedge [\mathbb{N} \ni y \geq 1]\Big]$$

```
    if x < y
      y = y-x;
    else
      x = x-y;
    end
```

$$\Big[[\gcd(x, y) = \gcd(A, B)] \wedge [\mathbb{N} \ni x \geq 1] \wedge [\mathbb{N} \ni y \geq 1]\Big]$$

```
  end
```

$$\Big[[\gcd(x, y) = \gcd(A, B)] \wedge \neg[x \neq y] \wedge [\mathbb{N} \ni x \geq 1] \wedge [\mathbb{N} \ni y \geq 1]\Big]$$

$$\Big[\text{If the program terminates, } x = y = \gcd(A, B)\Big]$$

Proof that $x = y = \gcd(A, B)$

$\Big[[\mathbb{N} \ni A \geq 1] \;\wedge\; [\mathbb{N} \ni B \geq 1] \;\wedge\; [\texttt{f(A,B)} \text{ is called.}]\Big]$

```
function result = f(x,y)
```

$\Big[[x = A] \;\wedge\; [y = B] \;\wedge\; [\mathbb{N} \ni x \geq 1] \;\wedge\; [\mathbb{N} \ni y \geq 1]\Big] \Rightarrow$

$\Big[[\gcd(x, y) = \gcd(A, B)] \;\wedge\; [\mathbb{N} \ni x \geq 1] \;\wedge\; [\mathbb{N} \ni y \geq 1]\Big]$

```
  while x < y || y < x
```

$\Big[[\gcd(x, y) = \gcd(A, B)] \;\wedge\; [x \neq y] \;\wedge\; [\mathbb{N} \ni x \geq 1] \;\wedge\; [\mathbb{N} \ni y \geq 1]\Big]$

```
    if x < y
      y = y-x;
    else
      x = x-y;
    end
```

$\Big[[\gcd(x, y) = \gcd(A, B)] \;\wedge\; [\mathbb{N} \ni x \geq 1] \;\wedge\; [\mathbb{N} \ni y \geq 1]\Big]$

```
  end
```

$\Big[[\gcd(x, y) = \gcd(A, B)] \;\wedge\; \neg[x \neq y] \;\wedge\; [\mathbb{N} \ni x \geq 1] \;\wedge\; [\mathbb{N} \ni y \geq 1]\Big]$

$\Big[\text{If the program terminates, } x = y = \gcd(A, B)\Big]$

Proof that $x = y = \gcd(A, B)$

$\Big[[\mathbb{N} \ni A \geq 1] \;\wedge\; [\mathbb{N} \ni B \geq 1] \;\wedge\; [\texttt{f(A,B)} \text{ is called.}]\Big]$

```
 function result = f(x,y)
```

$\Big[[x = A] \;\wedge\; [y = B] \;\wedge\; [\mathbb{N} \ni x \geq 1] \;\wedge\; [\mathbb{N} \ni y \geq 1]\Big] \Rightarrow$

$\Big[[\gcd(x, y) = \gcd(A, B)] \;\wedge\; [\mathbb{N} \ni x \geq 1] \;\wedge\; [\mathbb{N} \ni y \geq 1]\Big]$

```
   while x < y || y < x
```

$\Big[[\gcd(x, y) = \gcd(A, B)] \;\wedge\; [x \neq y] \;\wedge\; [\mathbb{N} \ni x \geq 1] \;\wedge\; [\mathbb{N} \ni y \geq 1]\Big]$

```
     if x < y
       y = y-x;
     else
       x = x-y;
     end
```

$\Big[[\gcd(x, y) = \gcd(A, B)] \;\wedge\; [\mathbb{N} \ni x \geq 1] \;\wedge\; [\mathbb{N} \ni y \geq 1]\Big]$

```
   end
```

$\Big[[\gcd(x, y) = \gcd(A, B)] \;\wedge\; \neg[x \neq y] \;\wedge\; [\mathbb{N} \ni x \geq 1] \;\wedge\; [\mathbb{N} \ni y \geq 1]\Big]$

$\Big[\text{If the program terminates, } \; x = y = \gcd(A, B)\Big]$

Proof that $x = y = \gcd(A, B)$

$$\Big[[\mathbb{N} \ni A \geq 1] \wedge [\mathbb{N} \ni B \geq 1] \wedge [\texttt{f(A,B)} \text{ is called.}]\Big]$$

```
function result = f(x,y)
```

$$\Big[[x = A] \wedge [y = B] \wedge [\mathbb{N} \ni x \geq 1] \wedge [\mathbb{N} \ni y \geq 1]\Big] \Rightarrow$$

$$\Big[[\gcd(x,y) = \gcd(A,B)] \wedge [\mathbb{N} \ni x \geq 1] \wedge [\mathbb{N} \ni y \geq 1]\Big]$$

```
   while x < y || y < x
```

$$\Big[[\gcd(x,y) = \gcd(A,B)] \wedge [x \neq y] \wedge [\mathbb{N} \ni x \geq 1] \wedge [\mathbb{N} \ni y \geq 1]\Big]$$

$[P]$

```
      if x < y
        y = y-x;
      else
        x = x-y;
      end
```

$[Q]$

$$\Big[[\gcd(x,y) = \gcd(A,B)] \wedge [\mathbb{N} \ni x \geq 1] \wedge [\mathbb{N} \ni y \geq 1]\Big]$$

```
   end
```

$$\Big[[\gcd(x,y) = \gcd(A,B)] \wedge \neg[x \neq y] \wedge [\mathbb{N} \ni x \geq 1] \wedge [\mathbb{N} \ni y \geq 1]\Big]$$

$$\Big[\text{If the program terminates, } x = y = \gcd(A,B)\Big]$$

Proof that $x = y = \gcd(A, B)$

$\Big[[\mathbb{N} \ni A \geq 1] \ \wedge\ [\mathbb{N} \ni B \geq 1] \ \wedge\ [\texttt{f(A,B)} \text{ is called.}]\Big]$

```
function result = f(x,y)
```

$\Big[[x = A] \ \wedge\ [y = B] \ \wedge\ [\mathbb{N} \ni x \geq 1] \ \wedge\ [\mathbb{N} \ni y \geq 1]\Big] \Rightarrow$

$\Big[[\gcd(x, y) = \gcd(A, B)] \ \wedge\ [\mathbb{N} \ni x \geq 1] \ \wedge\ [\mathbb{N} \ni y \geq 1]\Big]$

```
  while x < y || y < x
```

$\Big[[\gcd(x, y) = \gcd(A, B)] \ \wedge\ [x \neq y] \ \wedge\ [\mathbb{N} \ni x \geq 1] \ \wedge\ [\mathbb{N} \ni y \geq 1]\Big]$

Let $[P] := \Big[[\gcd(x, y) = \gcd(A, B)] \ \wedge\ [x \neq y] \ \wedge\ [\mathbb{N} \ni x \geq 1] \ \wedge\ [\mathbb{N} \ni y \geq 1]\Big]$

```
    if x < y
      y = y-x;
    else
      x = x-y;
    end
```

Let $[Q] := \Big[[\gcd(x, y) = \gcd(A, B)] \ \wedge\ [\mathbb{N} \ni x \geq 1] \ \wedge\ [\mathbb{N} \ni y \geq 1]\Big]$

$\Big[[\gcd(x, y) = \gcd(A, B)] \ \wedge\ [\mathbb{N} \ni x \geq 1] \ \wedge\ [\mathbb{N} \ni y \geq 1]\Big]$

```
  end
```

$\Big[[\gcd(x, y) = \gcd(A, B)] \ \wedge\ \neg[x \neq y] \ \wedge\ [\mathbb{N} \ni x \geq 1] \ \wedge\ [\mathbb{N} \ni y \geq 1]\Big]$

$\Big[\text{If the program terminates, } \ x = y = \gcd(A, B)\Big]$

Proof that $x = y = \gcd(A, B)$

$$\Big[[\mathbb{N} \ni A \geq 1] \ \wedge\ [\mathbb{N} \ni B \geq 1] \ \wedge\ [\texttt{f(A,B)} \text{ is called.}]\Big]$$

```
function result = f(x,y)
```

$$\Big[[x = A] \ \wedge\ [y = B] \ \wedge\ [\mathbb{N} \ni x \geq 1] \ \wedge\ [\mathbb{N} \ni y \geq 1]\Big] \Rightarrow$$

$$\Big[[\gcd(x, y) = \gcd(A, B)] \ \wedge\ [\mathbb{N} \ni x \geq 1] \ \wedge\ [\mathbb{N} \ni y \geq 1]\Big]$$

```
    while x < y || y < x
```

$$\Big[[\gcd(x, y) = \gcd(A, B)] \ \wedge\ [x \neq y] \ \wedge\ [\mathbb{N} \ni x \geq 1] \ \wedge\ [\mathbb{N} \ni y \geq 1]\Big]$$

Let $[P] := \Big[[\gcd(x, y) = \gcd(A, B)] \ \wedge\ [x \neq y] \ \wedge\ [\mathbb{N} \ni x \geq 1] \ \wedge\ [\mathbb{N} \ni y \geq 1]\Big]$

```
        if x < y
          y = y-x;
        else
          x = x-y;
        end
```

Let $[Q] := \Big[[\gcd(x, y) = \gcd(A, B)] \ \wedge\ [\mathbb{N} \ni x \geq 1] \ \wedge\ [\mathbb{N} \ni y \geq 1]\Big]$

$$\Big[[\gcd(x, y) = \gcd(A, B)] \ \wedge\ [\mathbb{N} \ni x \geq 1] \ \wedge\ [\mathbb{N} \ni y \geq 1]\Big]$$

```
    end
```

$$\Big[[\gcd(x, y) = \gcd(A, B)] \ \wedge\ \neg[x \neq y] \ \wedge\ [\mathbb{N} \ni x \geq 1] \ \wedge\ [\mathbb{N} \ni y \geq 1]\Big]$$

$$\Big[\text{If the program terminates, } \ x = y = \gcd(A, B)\Big]$$

Proof that $x = y = \gcd(A, B)$

$$\Big[[\mathbb{N} \ni A \geq 1] \wedge [\mathbb{N} \ni B \geq 1] \wedge [\texttt{f(A,B)} \text{ is called.}]\Big]$$

```
function result = f(x,y)
```

$$\Big[[x = A] \wedge [y = B] \wedge [\mathbb{N} \ni x \geq 1] \wedge [\mathbb{N} \ni y \geq 1]\Big] \Rightarrow$$

$$\Big[[\gcd(x, y) = \gcd(A, B)] \wedge [\mathbb{N} \ni x \geq 1] \wedge [\mathbb{N} \ni y \geq 1]\Big]$$

```
   while x < y || y < x
```

$$\Big[[\gcd(x, y) = \gcd(A, B)] \wedge [x \neq y] \wedge [\mathbb{N} \ni x \geq 1] \wedge [\mathbb{N} \ni y \geq 1]\Big]$$

$$[P] = \Big[[\gcd(x, y) = \gcd(A, B)] \wedge [x \neq y] \wedge [\mathbb{N} \ni x \geq 1] \wedge [\mathbb{N} \ni y \geq 1]\Big]$$

```
      if x < y
        y = y-x;
      else
        x = x-y;
      end
```

$$[Q] = \Big[[\gcd(x, y) = \gcd(A, B)] \wedge [\mathbb{N} \ni x \geq 1] \wedge [\mathbb{N} \ni y \geq 1]\Big]$$

$$\Big[[\gcd(x, y) = \gcd(A, B)] \wedge [\mathbb{N} \ni x \geq 1] \wedge [\mathbb{N} \ni y \geq 1]\Big]$$

```
   end
```

$$\Big[[\gcd(x, y) = \gcd(A, B)] \wedge \neg[x \neq y] \wedge [\mathbb{N} \ni x \geq 1] \wedge [\mathbb{N} \ni y \geq 1]\Big]$$

$$\Big[\text{If the program terminates, } \; x = y = \gcd(A, B)\Big]$$

The inner block

$$[P] = \Big[[\text{gcd}(x,y) = \text{gcd}(A,B)] \ \wedge \ [x \neq y] \ \wedge \ [\mathbb{N} \ni x \geq 1] \ \wedge \ [\mathbb{N} \ni y \geq 1]\Big]$$

```
if x < y

  y = y-x;
else

  x = x-y;
end
```

$$[Q] = \Big[[\text{gcd}(x,y) = \text{gcd}(A,B)] \ \wedge \ [\mathbb{N} \ni x \geq 1] \ \wedge \ [\mathbb{N} \ni y \geq 1]\Big]$$

The Rule of Conditional Branching demands:

$$[P] = \Big[[\mathrm{gcd}(x,y) = \mathrm{gcd}(A,B)] \ \wedge \ [x \neq y] \ \wedge \ [\mathbb{N} \ni x \geq 1] \ \wedge \ [\mathbb{N} \ni y \geq 1]\Big]$$

```
if x < y
```

$$\Big[[\mathrm{gcd}(x,y) = \mathrm{gcd}(A,B)] \ \wedge \ [x \neq y] \ \wedge \ [\mathbb{N} \ni x \geq 1] \ \wedge \ [\mathbb{N} \ni y \geq 1] \ \wedge \ [x < y]\Big]$$

```
    y = y-x;
```

$$\Big[[\mathrm{gcd}(x,y) = \mathrm{gcd}(A,B)] \ \wedge \ [\mathbb{N} \ni x \geq 1] \ \wedge \ [\mathbb{N} \ni y \geq 1]\Big]$$

```
else
```

$$\Big[[\mathrm{gcd}(x,y) = \mathrm{gcd}(A,B)] \ \wedge \ [x \neq y] \ \wedge \ [\mathbb{N} \ni x \geq 1] \ \wedge \ [\mathbb{N} \ni y \geq 1] \ \wedge \ \neg[x < y]\Big]$$

```
    x = x-y;
```

$$\Big[[\mathrm{gcd}(x,y) = \mathrm{gcd}(A,B)] \ \wedge \ [\mathbb{N} \ni x \geq 1] \ \wedge \ [\mathbb{N} \ni y \geq 1]\Big]$$

```
end
```

$$[Q] = \Big[[\mathrm{gcd}(x,y) = \mathrm{gcd}(A,B)] \ \wedge \ [\mathbb{N} \ni x \geq 1] \ \wedge \ [\mathbb{N} \ni y \geq 1]\Big]$$

By the Axiom of Assignment:

$$[P] = \Big[[\mathrm{gcd}(x,y) = \mathrm{gcd}(A,B)] \ \wedge\ [x \neq y] \ \wedge\ [\mathbb{N} \ni x \geq 1] \ \wedge\ [\mathbb{N} \ni y \geq 1]\Big]$$

```
if x < y
```

$$\Big[[\mathrm{gcd}(x,y) = \mathrm{gcd}(A,B)] \ \wedge\ [x \neq y] \ \wedge\ [\mathbb{N} \ni x \geq 1] \ \wedge\ [\mathbb{N} \ni y \geq 1] \ \wedge\ [x < y]\Big]$$

$$\Big[[\mathrm{gcd}(x,y-x) = \mathrm{gcd}(A,B)] \ \wedge\ [\mathbb{N} \ni x \geq 1] \ \wedge\ [\mathbb{N} \ni y-x \geq 1]\Big]$$

```
    y = y-x;
```

$$\Big[[\mathrm{gcd}(x,y) = \mathrm{gcd}(A,B)] \ \wedge\ [\mathbb{N} \ni x \geq 1] \ \wedge\ [\mathbb{N} \ni y \geq 1]\Big]$$

```
else
```

$$\Big[[\mathrm{gcd}(x,y) = \mathrm{gcd}(A,B)] \ \wedge\ [x \neq y] \ \wedge\ [\mathbb{N} \ni x \geq 1] \ \wedge\ [\mathbb{N} \ni y \geq 1] \ \wedge\ \neg[x < y]\Big]$$

$$\Big[[\mathrm{gcd}(x-y,y) = \mathrm{gcd}(A,B)] \ \wedge\ [\mathbb{N} \ni x-y \geq 1] \ \wedge\ [\mathbb{N} \ni y \geq 1]\Big]$$

```
    x = x-y;
```

$$\Big[[\mathrm{gcd}(x,y) = \mathrm{gcd}(A,B)] \ \wedge\ [\mathbb{N} \ni x \geq 1] \ \wedge\ [\mathbb{N} \ni y \geq 1]\Big]$$

```
end
```

$$[Q] = \Big[[\mathrm{gcd}(x,y) = \mathrm{gcd}(A,B)] \ \wedge\ [\mathbb{N} \ni x \geq 1] \ \wedge\ [\mathbb{N} \ni y \geq 1]\Big]$$

There are the implications:

$$[P] = \Big[[\mathrm{gcd}(x,y) = \mathrm{gcd}(A,B)] \;\wedge\; [x \neq y] \;\wedge\; [\mathbb{N} \ni x \geq 1] \;\wedge\; [\mathbb{N} \ni y \geq 1]\Big]$$

```
    if x < y
```

$$\Big[[\mathrm{gcd}(x,y) = \mathrm{gcd}(A,B)] \;\wedge\; [x \neq y] \;\wedge\; [\mathbb{N} \ni x \geq 1] \;\wedge\; [\mathbb{N} \ni y \geq 1] \;\wedge\; [x < y]\Big] \Rightarrow$$

$$\Big[[\mathrm{gcd}(x,y-x) = \mathrm{gcd}(A,B)] \;\wedge\; [\mathbb{N} \ni x \geq 1] \;\wedge\; [\mathbb{N} \ni y - x \geq 1]\Big]$$

```
      y = y-x;
```

$$\Big[[\mathrm{gcd}(x,y) = \mathrm{gcd}(A,B)] \;\wedge\; [\mathbb{N} \ni x \geq 1] \;\wedge\; [\mathbb{N} \ni y \geq 1]\Big]$$

```
    else
```

$$\Big[[\mathrm{gcd}(x,y) = \mathrm{gcd}(A,B)] \;\wedge\; [x \neq y] \;\wedge\; [\mathbb{N} \ni x \geq 1] \;\wedge\; [\mathbb{N} \ni y \geq 1] \;\wedge\; \neg[x < y]\Big] \Rightarrow$$

$$\Big[[\mathrm{gcd}(x-y,y) = \mathrm{gcd}(A,B)] \;\wedge\; [\mathbb{N} \ni x - y \geq 1] \;\wedge\; [\mathbb{N} \ni y \geq 1]\Big]$$

```
      x = x-y;
```

$$\Big[[\mathrm{gcd}(x,y) = \mathrm{gcd}(A,B)] \;\wedge\; [\mathbb{N} \ni x \geq 1] \;\wedge\; [\mathbb{N} \ni y \geq 1]\Big]$$

```
    end
```

$$[Q] = \Big[[\mathrm{gcd}(x,y) = \mathrm{gcd}(A,B)] \;\wedge\; [\mathbb{N} \ni x \geq 1] \;\wedge\; [\mathbb{N} \ni y \geq 1]\Big]$$

We invoke the rule: $$\frac{\Big[\big[[P \wedge B] \quad \mathtt{S} \quad [Q]\big] \quad \wedge \quad \big[[P \wedge \neg B] \quad \mathtt{T} \quad [Q]\big]\Big]}{\big[[P] \quad \underline{\mathtt{if}}\ B\ \underline{\mathtt{do}}\ \mathtt{S}\ \underline{\mathtt{else}}\ \underline{\mathtt{do}}\ \mathtt{T}\ \underline{\mathtt{end}} \quad [Q]\big]}$$

$$[P] = \big[[\gcd(x,y) = \gcd(A,B)] \ \wedge\ [x \neq y] \ \wedge\ [\mathbb{N} \ni x \geq 1] \ \wedge\ [\mathbb{N} \ni y \geq 1]\big]$$

```
    if x < y
```

$$\big[[\gcd(x,y) = \gcd(A,B)] \ \wedge\ [x \neq y] \ \wedge\ [\mathbb{N} \ni x \geq 1] \ \wedge\ [\mathbb{N} \ni y \geq 1] \ \wedge\ [x < y]\big] \Rightarrow$$

$$\big[[\gcd(x,y-x) = \gcd(A,B)] \ \wedge\ [\mathbb{N} \ni x \geq 1] \ \wedge\ [\mathbb{N} \ni y - x \geq 1]\big]$$

```
      y = y-x;
```

$$\big[[\gcd(x,y) = \gcd(A,B)] \ \wedge\ [\mathbb{N} \ni x \geq 1] \ \wedge\ [\mathbb{N} \ni y \geq 1]\big]$$

```
    else
```

$$\big[[\gcd(x,y) = \gcd(A,B)] \ \wedge\ [x \neq y] \ \wedge\ [\mathbb{N} \ni x \geq 1] \ \wedge\ [\mathbb{N} \ni y \geq 1] \ \wedge\ \neg[x < y]\big] \Rightarrow$$

$$\big[[\gcd(x-y,y) = \gcd(A,B)] \ \wedge\ [\mathbb{N} \ni x - y \geq 1] \ \wedge\ [\mathbb{N} \ni y \geq 1]\big]$$

```
      x = x-y;
```

$$\big[[\gcd(x,y) = \gcd(A,B)] \ \wedge\ [\mathbb{N} \ni x \geq 1] \ \wedge\ [\mathbb{N} \ni y \geq 1]\big]$$

```
    end
```

$$[Q] = \big[[\gcd(x,y) = \gcd(A,B)] \ \wedge\ [\mathbb{N} \ni x \geq 1] \ \wedge\ [\mathbb{N} \ni y \geq 1]\big]$$

Where: $[P]$ if B do S else do T end $[Q]$

$$[P] = \Big[[\gcd(x,y)=\gcd(A,B)] \wedge [x \neq y] \wedge [\mathbb{N} \ni x \geq 1] \wedge [\mathbb{N} \ni y \geq 1]\Big]$$

```
if x < y
```

$$\Big[[\gcd(x,y)=\gcd(A,B)] \wedge [x \neq y] \wedge [\mathbb{N} \ni x \geq 1] \wedge [\mathbb{N} \ni y \geq 1] \wedge [x < y]\Big] \Rightarrow$$

$$\Big[[\gcd(x,y-x)=\gcd(A,B)] \wedge [\mathbb{N} \ni x \geq 1] \wedge [\mathbb{N} \ni y-x \geq 1]\Big]$$

```
y = y-x;
```

$$\Big[[\gcd(x,y)=\gcd(A,B)] \wedge [\mathbb{N} \ni x \geq 1] \wedge [\mathbb{N} \ni y \geq 1]\Big]$$

```
else
```

$$\Big[[\gcd(x,y)=\gcd(A,B)] \wedge [x \neq y] \wedge [\mathbb{N} \ni x \geq 1] \wedge [\mathbb{N} \ni y \geq 1] \wedge \neg[x < y]\Big] \Rightarrow$$

$$\Big[[\gcd(x-y,y)=\gcd(A,B)] \wedge [\mathbb{N} \ni x-y \geq 1] \wedge [\mathbb{N} \ni y \geq 1]\Big]$$

```
x = x-y;
```

$$\Big[[\gcd(x,y)=\gcd(A,B)] \wedge [\mathbb{N} \ni x \geq 1] \wedge [\mathbb{N} \ni y \geq 1]\Big]$$

```
end
```

$$[Q] = \Big[[\gcd(x,y)=\gcd(A,B)] \wedge [\mathbb{N} \ni x \geq 1] \wedge [\mathbb{N} \ni y \geq 1]\Big]$$

Where: $[P]$ `if` B `do` S `else do` T `end` $[Q]$

$$[P] = \Big[[\gcd(x,y)=\gcd(A,B)] \wedge [x \neq y] \wedge [\mathbb{N} \ni x \geq 1] \wedge [\mathbb{N} \ni y \geq 1]\Big]$$

```
if x < y
```

$$\Big[[\gcd(x,y)=\gcd(A,B)] \wedge [x \neq y] \wedge [\mathbb{N} \ni x \geq 1] \wedge [\mathbb{N} \ni y \geq 1] \wedge [x < y]\Big] \Rightarrow$$

$$\Big[[\gcd(x,y-x)=\gcd(A,B)] \wedge [\mathbb{N} \ni x \geq 1] \wedge [\mathbb{N} \ni y-x \geq 1]\Big]$$

```
    y = y-x;
```

$$\Big[[\gcd(x,y)=\gcd(A,B)] \wedge [\mathbb{N} \ni x \geq 1] \wedge [\mathbb{N} \ni y \geq 1]\Big]$$

```
else
```

$$\Big[[\gcd(x,y)=\gcd(A,B)] \wedge [x \neq y] \wedge [\mathbb{N} \ni x \geq 1] \wedge [\mathbb{N} \ni y \geq 1] \wedge \neg[x < y]\Big] \Rightarrow$$

$$\Big[[\gcd(x-y,y)=\gcd(A,B)] \wedge [\mathbb{N} \ni x-y \geq 1] \wedge [\mathbb{N} \ni y \geq 1]\Big]$$

```
    x = x-y;
```

$$\Big[[\gcd(x,y)=\gcd(A,B)] \wedge [\mathbb{N} \ni x \geq 1] \wedge [\mathbb{N} \ni y \geq 1]\Big]$$

```
end
```

$$[Q] = \Big[[\gcd(x,y)=\gcd(A,B)] \wedge [\mathbb{N} \ni x \geq 1] \wedge [\mathbb{N} \ni y \geq 1]\Big]$$

Where: $[P]$ if B do S else do T end $[Q]$

$$[P] = \Big[[\gcd(x,y)=\gcd(A,B)] \wedge [x \neq y] \wedge [\mathbb{N} \ni x \geq 1] \wedge [\mathbb{N} \ni y \geq 1]\Big]$$

```
if x < y
```

$$\Big[[\gcd(x,y)=\gcd(A,B)] \wedge [x \neq y] \wedge [\mathbb{N} \ni x \geq 1] \wedge [\mathbb{N} \ni y \geq 1] \wedge [x<y]\Big] \Rightarrow$$

$$\Big[[\gcd(x,y-x)=\gcd(A,B)] \wedge [\mathbb{N} \ni x \geq 1] \wedge [\mathbb{N} \ni y-x \geq 1]\Big]$$

```
    y = y-x;
```

$$\Big[[\gcd(x,y)=\gcd(A,B)] \wedge [\mathbb{N} \ni x \geq 1] \wedge [\mathbb{N} \ni y \geq 1]\Big]$$

```
else
```

$$\Big[[\gcd(x,y)=\gcd(A,B)] \wedge [x \neq y] \wedge [\mathbb{N} \ni x \geq 1] \wedge [\mathbb{N} \ni y \geq 1] \wedge \neg[x<y]\Big] \Rightarrow$$

$$\Big[[\gcd(x-y,y)=\gcd(A,B)] \wedge [\mathbb{N} \ni x-y \geq 1] \wedge [\mathbb{N} \ni y \geq 1]\Big]$$

```
    x = x-y;
```

$$\Big[[\gcd(x,y)=\gcd(A,B)] \wedge [\mathbb{N} \ni x \geq 1] \wedge [\mathbb{N} \ni y \geq 1]\Big]$$

```
end
```

$$[Q] = \Big[[\gcd(x,y)=\gcd(A,B)] \wedge [\mathbb{N} \ni x \geq 1] \wedge [\mathbb{N} \ni y \geq 1]\Big]$$

Where: $[P]$ `if` B `do` S `else` `do` T `end` $[Q]$

$$[P] = \Big[[\mathrm{gcd}(x,y) = \mathrm{gcd}(A,B)] \ \wedge \ [x \neq y] \ \wedge \ [\mathbb{N} \ni x \geq 1] \ \wedge \ [\mathbb{N} \ni y \geq 1]\Big]$$

```
if x < y
```

$$\Big[[\mathrm{gcd}(x,y) = \mathrm{gcd}(A,B)] \ \wedge \ [x \neq y] \ \wedge \ [\mathbb{N} \ni x \geq 1] \ \wedge \ [\mathbb{N} \ni y \geq 1] \ \wedge \ [x < y]\Big] \Rightarrow$$

$$\Big[[\mathrm{gcd}(x,y-x) = \mathrm{gcd}(A,B)] \ \wedge \ [\mathbb{N} \ni x \geq 1] \ \wedge \ [\mathbb{N} \ni y - x \geq 1]\Big]$$

```
    y = y-x;
```

$$\Big[[\mathrm{gcd}(x,y) = \mathrm{gcd}(A,B)] \ \wedge \ [\mathbb{N} \ni x \geq 1] \ \wedge \ [\mathbb{N} \ni y \geq 1]\Big]$$

```
else
```

$$\Big[[\mathrm{gcd}(x,y) = \mathrm{gcd}(A,B)] \ \wedge \ [x \neq y] \ \wedge \ [\mathbb{N} \ni x \geq 1] \ \wedge \ [\mathbb{N} \ni y \geq 1] \ \wedge \ \neg[x < y]\Big] \Rightarrow$$

$$\Big[[\mathrm{gcd}(x-y,y) = \mathrm{gcd}(A,B)] \ \wedge \ [\mathbb{N} \ni x - y \geq 1] \ \wedge \ [\mathbb{N} \ni y \geq 1]\Big]$$

```
    x = x-y;
```

$$\Big[[\mathrm{gcd}(x,y) = \mathrm{gcd}(A,B)] \ \wedge \ [\mathbb{N} \ni x \geq 1] \ \wedge \ [\mathbb{N} \ni y \geq 1]\Big]$$

```
end
```

$$[Q] = \Big[[\mathrm{gcd}(x,y) = \mathrm{gcd}(A,B)] \ \wedge \ [\mathbb{N} \ni x \geq 1] \ \wedge \ [\mathbb{N} \ni y \geq 1]\Big]$$

Where: $[P]$ if B do$S$ else do T end $[Q]$

$$[P] = \Big[[\gcd(x,y) = \gcd(A,B)] \wedge [x \neq y] \wedge [\mathbb{N} \ni x \geq 1] \wedge [\mathbb{N} \ni y \geq 1]\Big]$$

```
if x < y
```

$$\Big[[\gcd(x,y) = \gcd(A,B)] \wedge [x \neq y] \wedge [\mathbb{N} \ni x \geq 1] \wedge [\mathbb{N} \ni y \geq 1] \wedge [x < y]\Big] \Rightarrow$$

$$\Big[[\gcd(x,y-x) = \gcd(A,B)] \wedge [\mathbb{N} \ni x \geq 1] \wedge [\mathbb{N} \ni y - x \geq 1]\Big]$$

```
    y = y-x;
```

$$\Big[[\gcd(x,y) = \gcd(A,B)] \wedge [\mathbb{N} \ni x \geq 1] \wedge [\mathbb{N} \ni y \geq 1]\Big]$$

```
else
```

$$\Big[[\gcd(x,y) = \gcd(A,B)] \wedge [x \neq y] \wedge [\mathbb{N} \ni x \geq 1] \wedge [\mathbb{N} \ni y \geq 1] \wedge \neg[x < y]\Big] \Rightarrow$$

$$\Big[[\gcd(x-y,y) = \gcd(A,B)] \wedge [\mathbb{N} \ni x - y \geq 1] \wedge [\mathbb{N} \ni y \geq 1]\Big]$$

```
    x = x-y;
```

$$\Big[[\gcd(x,y) = \gcd(A,B)] \wedge [\mathbb{N} \ni x \geq 1] \wedge [\mathbb{N} \ni y \geq 1]\Big]$$

```
end
```

$$[Q] = \Big[[\gcd(x,y) = \gcd(A,B)] \wedge [\mathbb{N} \ni x \geq 1] \wedge [\mathbb{N} \ni y \geq 1]\Big]$$

Where: $[P]$ if B do S else do T end $[Q]$

$$[P] = \Big[[\gcd(x,y) = \gcd(A,B)] \;\wedge\; [x \neq y] \;\wedge\; [\mathbb{N} \ni x \geq 1] \;\wedge\; [\mathbb{N} \ni y \geq 1]\Big]$$

```
if x < y
```

$$\Big[[\gcd(x,y) = \gcd(A,B)] \;\wedge\; [x \neq y] \;\wedge\; [\mathbb{N} \ni x \geq 1] \;\wedge\; [\mathbb{N} \ni y \geq 1] \;\wedge\; [x < y]\Big] \Rightarrow$$

$$\Big[[\gcd(x,y-x) = \gcd(A,B)] \;\wedge\; [\mathbb{N} \ni x \geq 1] \;\wedge\; [\mathbb{N} \ni y - x \geq 1]\Big]$$

```
y = y-x;
```

$$\Big[[\gcd(x,y) = \gcd(A,B)] \;\wedge\; [\mathbb{N} \ni x \geq 1] \;\wedge\; [\mathbb{N} \ni y \geq 1]\Big]$$

```
else
```

$$\Big[[\gcd(x,y) = \gcd(A,B)] \;\wedge\; [x \neq y] \;\wedge\; [\mathbb{N} \ni x \geq 1] \;\wedge\; [\mathbb{N} \ni y \geq 1] \;\wedge\; \neg[x < y]\Big] \Rightarrow$$

$$\Big[[\gcd(x-y,y) = \gcd(A,B)] \;\wedge\; [\mathbb{N} \ni x - y \geq 1] \;\wedge\; [\mathbb{N} \ni y \geq 1]\Big]$$

```
x = x-y;
```

$$\Big[[\gcd(x,y) = \gcd(A,B)] \;\wedge\; [\mathbb{N} \ni x \geq 1] \;\wedge\; [\mathbb{N} \ni y \geq 1]\Big]$$

```
end
```

$$[Q] = \Big[[\gcd(x,y) = \gcd(A,B)] \;\wedge\; [\mathbb{N} \ni x \geq 1] \;\wedge\; [\mathbb{N} \ni y \geq 1]\Big]$$

So the inner block is proved.

$$[P] = \Big[[\gcd(x,y) = \gcd(A,B)] \wedge [x \neq y] \wedge [\mathbb{N} \ni x \geq 1] \wedge [\mathbb{N} \ni y \geq 1]\Big]$$

```
if x < y
```

$$\Big[[\gcd(x,y) = \gcd(A,B)] \wedge [x \neq y] \wedge [\mathbb{N} \ni x \geq 1] \wedge [\mathbb{N} \ni y \geq 1] \wedge [x < y]\Big] \Rightarrow$$

$$\Big[[\gcd(x,y-x) = \gcd(A,B)] \wedge [\mathbb{N} \ni x \geq 1] \wedge [\mathbb{N} \ni y - x \geq 1]\Big]$$

```
   y = y-x;
```

$$\Big[[\gcd(x,y) = \gcd(A,B)] \wedge [\mathbb{N} \ni x \geq 1] \wedge [\mathbb{N} \ni y \geq 1]\Big]$$

```
else
```

$$\Big[[\gcd(x,y) = \gcd(A,B)] \wedge [x \neq y] \wedge [\mathbb{N} \ni x \geq 1] \wedge [\mathbb{N} \ni y \geq 1] \wedge \neg[x < y]\Big] \Rightarrow$$

$$\Big[[\gcd(x-y,y) = \gcd(A,B)] \wedge [\mathbb{N} \ni x - y \geq 1] \wedge [\mathbb{N} \ni y \geq 1]\Big]$$

```
   x = x-y;
```

$$\Big[[\gcd(x,y) = \gcd(A,B)] \wedge [\mathbb{N} \ni x \geq 1] \wedge [\mathbb{N} \ni y \geq 1]\Big]$$

```
end
```

$$[Q] = \Big[[\gcd(x,y) = \gcd(A,B)] \wedge [\mathbb{N} \ni x \geq 1] \wedge [\mathbb{N} \ni y \geq 1]\Big]$$

$$\Big[[\mathbb{N} \ni A \geq 1] \ \wedge \ [\mathbb{N} \ni B \geq 1] \ \wedge \ [\texttt{f(A,B)} \text{ is called.}]\Big]$$

```
function result = f(x,y)
```

$$\Big[[x = A] \ \wedge \ [y = B] \ \wedge \ [\mathbb{N} \ni x \geq 1] \ \wedge \ [\mathbb{N} \ni y \geq 1]\Big] \Rightarrow$$

$$\Big[[\gcd(x, y) = \gcd(A, B)] \ \wedge \ [\mathbb{N} \ni x \geq 1] \ \wedge \ [\mathbb{N} \ni y \geq 1]\Big]$$

```
  while x < y || y < x
```

$$\Big[[\gcd(x, y) = \gcd(A, B)] \ \wedge \ [x \neq y] \ \wedge \ [\mathbb{N} \ni x \geq 1] \ \wedge \ [\mathbb{N} \ni y \geq 1]\Big]$$

$$[P] = \Big[[\gcd(x, y) = \gcd(A, B)] \ \wedge \ [x \neq y] \ \wedge \ [\mathbb{N} \ni x \geq 1] \ \wedge \ [\mathbb{N} \ni y \geq 1]\Big]$$

```
    if x < y
      y = y-x;
    else
      x = x-y;
    end
```

$$[Q] = \Big[[\gcd(x, y) = \gcd(A, B)] \ \wedge \ [\mathbb{N} \ni x \geq 1] \ \wedge \ [\mathbb{N} \ni y \geq 1]\Big]$$

$$\Big[[\gcd(x, y) = \gcd(A, B)] \ \wedge \ [\mathbb{N} \ni x \geq 1] \ \wedge \ [\mathbb{N} \ni y \geq 1]\Big]$$

```
  end
```

$$\Big[[\gcd(x, y) = \gcd(A, B)] \ \wedge \ \neg[x \neq y] \ \wedge \ [\mathbb{N} \ni x \geq 1] \ \wedge \ [\mathbb{N} \ni y \geq 1]\Big]$$

$$\Big[\text{If the program terminates, } \ x = y = \gcd(A, B)\Big]$$

Thus, in our main proof:

$$\Big[[\mathbb{N} \ni A \geq 1] \;\wedge\; [\mathbb{N} \ni B \geq 1] \;\wedge\; [\texttt{f(A,B)} \text{ is called.}]\Big]$$

```
function result = f(x,y)
```

$$\Big[[x = A] \;\wedge\; [y = B] \;\wedge\; [\mathbb{N} \ni x \geq 1] \;\wedge\; [\mathbb{N} \ni y \geq 1]\Big] \Rightarrow$$

$$\Big[[\gcd(x,y) = \gcd(A,B)] \;\wedge\; [\mathbb{N} \ni x \geq 1] \;\wedge\; [\mathbb{N} \ni y \geq 1]\Big]$$

```
   while x < y || y < x
```

$$\Big[[\gcd(x,y) = \gcd(A,B)] \;\wedge\; [x \neq y] \;\wedge\; [\mathbb{N} \ni x \geq 1] \;\wedge\; [\mathbb{N} \ni y \geq 1]\Big]$$

$$[P] = \Big[[\gcd(x,y) = \gcd(A,B)] \;\wedge\; [x \neq y] \;\wedge\; [\mathbb{N} \ni x \geq 1] \;\wedge\; [\mathbb{N} \ni y \geq 1]\Big]$$

```
      if x < y
        y = y-x;
      else
        x = x-y;
      end
```

$$[Q] = \Big[[\gcd(x,y) = \gcd(A,B)] \;\wedge\; [\mathbb{N} \ni x \geq 1] \;\wedge\; [\mathbb{N} \ni y \geq 1]\Big]$$

$$\Big[[\gcd(x,y) = \gcd(A,B)] \;\wedge\; [\mathbb{N} \ni x \geq 1] \;\wedge\; [\mathbb{N} \ni y \geq 1]\Big]$$

```
   end
```

$$\Big[[\gcd(x,y) = \gcd(A,B)] \;\wedge\; \neg[x \neq y] \;\wedge\; [\mathbb{N} \ni x \geq 1] \;\wedge\; [\mathbb{N} \ni y \geq 1]\Big]$$

$$\Big[\text{If the program terminates, } \; x = y = \gcd(A,B)\Big]$$

There are three collisions left; trivially:

$\Big[[\mathbb{N} \ni A \geq 1] \ \wedge \ [\mathbb{N} \ni B \geq 1] \ \wedge \ [\texttt{f(A,B)} \text{ is called.}]\Big]$

```
  function result = f(x,y)
```

$\Big[[x = A] \ \wedge \ [y = B] \ \wedge \ [\mathbb{N} \ni x \geq 1] \ \wedge \ [\mathbb{N} \ni y \geq 1]\Big] \Rightarrow$

$\Big[[\gcd(x, y) = \gcd(A, B)] \ \wedge \ [\mathbb{N} \ni x \geq 1] \ \wedge \ [\mathbb{N} \ni y \geq 1]\Big]$

```
    while x < y || y < x
```

$\Big[[\gcd(x, y) = \gcd(A, B)] \ \wedge \ [x \neq y] \ \wedge \ [\mathbb{N} \ni x \geq 1] \ \wedge \ [\mathbb{N} \ni y \geq 1]\Big] \Rightarrow$

$[P] = \Big[[\gcd(x, y) = \gcd(A, B)] \ \wedge \ [x \neq y] \ \wedge \ [\mathbb{N} \ni x \geq 1] \ \wedge \ [\mathbb{N} \ni y \geq 1]\Big]$

```
      if x < y
        y = y-x;
      else
        x = x-y;
      end
```

$[Q] = \Big[[\gcd(x, y) = \gcd(A, B)] \ \wedge \ [\mathbb{N} \ni x \geq 1] \ \wedge \ [\mathbb{N} \ni y \geq 1]\Big] \Rightarrow$

$\Big[[\gcd(x, y) = \gcd(A, B)] \ \wedge \ [\mathbb{N} \ni x \geq 1] \ \wedge \ [\mathbb{N} \ni y \geq 1]\Big]$

```
    end
```

$\Big[[\gcd(x, y) = \gcd(A, B)] \ \wedge \ \neg[x \neq y] \ \wedge \ [\mathbb{N} \ni x \geq 1] \ \wedge \ [\mathbb{N} \ni y \geq 1]\Big]$

$\Big[\text{If the program terminates, } \ x = y = \gcd(A, B)\Big]$

And, clearly:

$$\Big[[\mathbb{N} \ni A \geq 1] \ \wedge\ [\mathbb{N} \ni B \geq 1] \ \wedge\ [\texttt{f(A,B)} \text{ is called.}] \Big]$$

```
function result = f(x,y)
```

$$\Big[[x = A] \ \wedge\ [y = B] \ \wedge\ [\mathbb{N} \ni x \geq 1] \ \wedge\ [\mathbb{N} \ni y \geq 1] \Big] \Rightarrow$$

$$\Big[[\gcd(x,y) = \gcd(A,B)] \ \wedge\ [\mathbb{N} \ni x \geq 1] \ \wedge\ [\mathbb{N} \ni y \geq 1] \Big]$$

```
  while x < y || y < x
```

$$\Big[[\gcd(x,y) = \gcd(A,B)] \ \wedge\ [x \neq y] \ \wedge\ [\mathbb{N} \ni x \geq 1] \ \wedge\ [\mathbb{N} \ni y \geq 1] \Big] \Rightarrow$$

$$[P] = \Big[[\gcd(x,y) = \gcd(A,B)] \ \wedge\ [x \neq y] \ \wedge\ [\mathbb{N} \ni x \geq 1] \ \wedge\ [\mathbb{N} \ni y \geq 1] \Big]$$

```
    if x < y
      y = y-x;
    else
      x = x-y;
    end
```

$$[Q] = \Big[[\gcd(x,y) = \gcd(A,B)] \ \wedge\ [\mathbb{N} \ni x \geq 1] \ \wedge\ [\mathbb{N} \ni y \geq 1] \Big] \Rightarrow$$

$$\Big[[\gcd(x,y) = \gcd(A,B)] \ \wedge\ [\mathbb{N} \ni x \geq 1] \ \wedge\ [\mathbb{N} \ni y \geq 1] \Big]$$

```
  end
```

$$\Big[[\gcd(x,y) = \gcd(A,B)] \ \wedge\ \neg[x \neq y] \ \wedge\ [\mathbb{N} \ni x \geq 1] \ \wedge\ [\mathbb{N} \ni y \geq 1] \Big] \Rightarrow$$

$$\Big[\text{If the program terminates, } \ x = y = \gcd(A,B) \Big]$$

$\square$

What shall be our result?

```
function result = f(x,y)
  while x < y || y < x
    if x < y
      y = y-x;
    else
      x = x-y;
    end
  end
```

What shall be our result?

```
function result = f(x,y)
  while x < y || y < x
    if x < y
      y = y-x;
    else
      x = x-y;
    end
  end
  result = (x+y)/2;
```

What we know:

$\left[\mathbb{N} \ni A \geq 1 \ \wedge \ \mathbb{N} \ni B \geq 1 \ \wedge \ \texttt{f(A,B)} \text{ is called.}\right]$

```
function result = f(x,y)
  while x < y || y < x
    if x < y
      y = y-x;
    else
      x = x-y;
    end
  end
  result = (x+y)/2;
```

$\left[\text{If the program terminates, } \ \texttt{result} = \gcd(A, B)\right]$

Let us make it more symmetric.

```
function result = f(x,y)
  while x < y || y < x
    if x < y
      y = y-x;
    else
      x = x-y;
    end
  end
  result = (x+y)/2;
```

Only one result is rather asymmetric...

```
function [fR, sR] = f(x,y)
  while x < y || y < x
    if x < y
      y = y-x;
    else
      x = x-y;
    end
  end
  result = (x+y)/2;
```

Only one result is rather asymmetric...

```
function [fR, sR] = f(x,y)
  while x < y || y < x
    if x < y
      y = y-x;
    else
      x = x-y;
    end
  end
  fR = (x+y)/2;
  sR = ?
```

More results may need more variables...

```
function [fR, sR] = f(x,y)
  u = x;
  while x < y || y < x
    if x < y
      y = y-x;
    else
      x = x-y;
    end
  end
  fR = (x+y)/2;
  sR = ?
```

More results may need more variables...

```
function [fR, sR] = f(x,y)
  u = x;
  v = y;
  while x < y || y < x
    if x < y
      y = y-x;
    else
      x = x-y;
    end
  end
  fR = (x+y)/2;
  sR = ?
```

We had better balance this surplus of minuses...

```
function [fR, sR] = f(x,y)
  u = x;
  v = y;
  while x < y || y < x
    if x < y
      y = y-x;
      v = v+u;
    else
      x = x-y;
    end
  end
  fR = (x+y)/2;
  sR = ?
```

We had better balance this surplus of minuses...

```
function [fR, sR] = f(x,y)
  u = x;
  v = y;
  while x < y || y < x
    if x < y
      y = y-x;
      v = v+u;
    else
      x = x-y;
      u = u+v;
    end
  end
  fR = (x+y)/2;
  sR = ?
```

What shall our second result be?

```
function [fR, sR] = f(x,y)
  u = x;
  v = y;
  while x < y || y < x
    if x < y
      y = y-x;
      v = v+u;
    else
      x = x-y;
      u = u+v;
    end
  end
  fR = (x+y)/2;
  sR = ?
```

A symmetric one, of course...

```
function [fR, sR] = f(x,y)
  u = x;
  v = y;
  while x < y || y < x
    if x < y
      y = y-x;
      v = v+u;
    else
      x = x-y;
      u = u+v;
    end
  end
  fR = (x+y)/2;
  sR = (u+v)/2;
```

So, what is sR?

```
function [fR, sR] = f(x,y)
  u = x;
  v = y;
  while x < y || y < x
    if x < y
      y = y-x;
      v = v+u;
    else
      x = x-y;
      u = u+v;
    end
  end
  fR = (x+y)/2;
  sR = (u+v)/2;
```

Well, what is the counterpart to gcd?

$\left[\mathbb{N} \ni A \geq 1 \ \wedge \ \mathbb{N} \ni B \geq 1 \ \wedge \ \texttt{f(A,B)} \text{ is called.}\right]$

```
function [fR, sR] = f(x,y)
  u = x;
  v = y;
  while x < y || y < x
    if x < y
      y = y-x;
      v = v+u;
    else
      x = x-y;
      u = u+v;
    end
  end
  fR = (x+y)/2;
  sR = (u+v)/2;
```

$\left[\text{If the program terminates, } \ sR = \mathrm{scm}(A, B).\right]$

How could we find a good invariant?

$\left[\mathbb{N} \ni A \geq 1 \ \wedge \ \mathbb{N} \ni B \geq 1 \ \wedge \ \texttt{f(A,B)} \text{ is called.}\right]$

```
function [fR, sR] = f(x,y)
  u = x;
  v = y;
  while x < y || y < x
    if x < y
      y = y-x;
      v = v+u;
    else
      x = x-y;
      u = u+v;
    end
  end
  fR = (x+y)/2;
  sR = (u+v)/2;
```

$\left[\text{If the program terminates, } \ sR = \text{scm}(A, B).\right]$

We use what we know...

$\left[\mathbb{N} \ni A \geq 1 \ \land \ \mathbb{N} \ni B \geq 1 \ \land\ \texttt{f(A,B)} \text{ is called.}\right]$

```
function [fR, sR] = f(x,y)
  u = x;
  v = y;
  while x < y || y < x
    if x < y
      y = y-x;
      v = v+u;
    else
      x = x-y;
      u = u+v;
    end
  end
  fR = (x+y)/2;
  sR = (u+v)/2;
```

$\left[\text{If the program terminates, } \ sR = \text{scm}(A, B).\right]$

$\left[\text{If the program terminates, } \ \texttt{fR} = \gcd(A, B)\right]$

...and try to invent a loop invariant.

$\left[\mathbb{N} \ni A \geq 1 \;\wedge\; \mathbb{N} \ni B \geq 1 \;\wedge\; \texttt{f(A,B)} \text{ is called.} \right]$

```
function [fR, sR] = f(x,y)
  u = x;
  v = y;
  while x < y || y < x
    if x < y
      y = y-x;
      v = v+u;
    else
      x = x-y;
      u = u+v;
    end
  end
  fR = (x+y)/2;
  sR = (u+v)/2;
```

Well, suppose we are right in both assertions.

$\left[\text{If the program terminates,}\;\; sR = \text{scm}(A, B). \right]$

$\left[\text{If the program terminates,}\;\; \texttt{fR} = \text{gcd}(A, B) \right]$

...and try to invent a loop invariant.

$\left[\mathbb{N} \ni A \geq 1 \ \wedge \ \mathbb{N} \ni B \geq 1 \ \wedge \ \texttt{f(A,B)} \text{ is called.}\right]$

```
function [fR, sR] = f(x,y)
  u = x;
  v = y;
  while x < y || y < x
    if x < y
      y = y-x;
      v = v+u;
    else
      x = x-y;
      u = u+v;
    end
  end
  fR = (x+y)/2;
  sR = (u+v)/2;
```

$\left[\text{If the program terminates, } \ sR = \text{scm}(A, B).\right]$

$\left[\text{If the program terminates, } \ \texttt{fR} = \text{gcd}(A, B)\right]$

Well, suppose we are right in both assertions.
Then, upon termination,

...and try to invent a loop invariant.

$\left[\mathbb{N} \ni A \geq 1 \ \wedge \ \mathbb{N} \ni B \geq 1 \ \wedge \ \texttt{f(A,B)} \text{ is called.}\right]$

```
function [fR, sR] = f(x,y)
  u = x;
  v = y;
  while x < y || y < x
    if x < y
      y = y-x;
      v = v+u;
    else
      x = x-y;
      u = u+v;
    end
  end
  fR = (x+y)/2;
  sR = (u+v)/2;
```

$\left[\text{If the program terminates, } \ sR = \text{scm}(A, B).\right]$

$\left[\text{If the program terminates, } \ \texttt{fR} = \text{gcd}(A, B)\right]$

Well, suppose we are right in both assertions.
Then, upon termination,

$$\text{gcd}(A, B) \cdot \text{scm}(A, B) = fR \cdot sR$$

...and try to invent a loop invariant.

$\left[\mathbb{N} \ni A \geq 1 \ \wedge \ \mathbb{N} \ni B \geq 1 \ \wedge \ \texttt{f(A,B)} \text{ is called.}\right]$

```
function [fR, sR] = f(x,y)
  u = x;
  v = y;
  while x < y || y < x
    if x < y
      y = y-x;
      v = v+u;
    else
      x = x-y;
      u = u+v;
    end
  end
  fR = (x+y)/2;
  sR = (u+v)/2;
```

$\left[\text{If the program terminates, } \ sR = \text{scm}(A, B).\right]$

$\left[\text{If the program terminates, } \ \texttt{fR} = \text{gcd}(A, B)\right]$

Well, suppose we are right in both assertions.
Then, upon termination,

$$A \cdot B = \text{gcd}(A, B) \cdot \text{scm}(A, B) = fR \cdot sR$$

...and try to invent a loop invariant.

$\left[\mathbb{N} \ni A \geq 1 \ \wedge\ \mathbb{N} \ni B \geq 1 \ \wedge\ \texttt{f(A,B)} \text{ is called.}\right]$

```
function [fR, sR] = f(x,y)
  u = x;
  v = y;
  while x < y || y < x
    if x < y
      y = y-x;
      v = v+u;
    else
      x = x-y;
      u = u+v;
    end
  end
  fR = (x+y)/2;
  sR = (u+v)/2;
```

$\left[\text{If the program terminates, } \ sR = \text{scm}(A, B).\right]$

$\left[\text{If the program terminates, } \ \texttt{fR} = \text{gcd}(A, B)\right]$

Well, suppose we are right in both assertions. Then, upon termination,

$$A \cdot B = \text{gcd}(A, B) \cdot \text{scm}(A, B) = fR \cdot sR$$

$$\text{But: } \frac{x+y}{2} \cdot \frac{u+v}{2} = fR \cdot sR$$

...and try to invent a loop invariant.

$\left[\mathbb{N} \ni A \geq 1 \ \wedge\ \mathbb{N} \ni B \geq 1 \ \wedge\ \texttt{f(A,B)} \text{ is called.}\right]$

```
function [fR, sR] = f(x,y)
  u = x;
  v = y;
  while x < y || y < x
    if x < y
      y = y-x;
      v = v+u;
    else
      x = x-y;
      u = u+v;
    end
  end
  fR = (x+y)/2;
  sR = (u+v)/2;
```

$\left[\text{If the program terminates, } \ sR = \operatorname{scm}(A, B).\right]$

$\left[\text{If the program terminates, } \ \texttt{fR} = \gcd(A, B)\right]$

Well, suppose we are right in both assertions.
Then, upon termination,

$$A \cdot B = \gcd(A, B) \cdot \operatorname{scm}(A, B) = fR \cdot sR$$

Upon termination: $\frac{x+x}{2} \cdot \frac{u+v}{2} = fR \cdot sR$

...and try to invent a loop invariant.

$\left[\mathbb{N} \ni A \geq 1 \ \wedge\ \mathbb{N} \ni B \geq 1 \ \wedge\ \texttt{f(A,B)} \text{ is called.}\right]$

```
function [fR, sR] = f(x,y)
  u = x;
  v = y;
  while x < y || y < x
    if x < y
      y = y-x;
      v = v+u;
    else
      x = x-y;
      u = u+v;
    end
  end
  fR = (x+y)/2;
  sR = (u+v)/2;
```

$\left[\text{If the program terminates, } sR = \mathrm{scm}(A, B).\right]$

$\left[\text{If the program terminates, } \texttt{fR} = \mathrm{gcd}(A, B)\right]$

Well, suppose we are right in both assertions.
Then, upon termination,

$$A \cdot B = \mathrm{gcd}(A, B) \cdot \mathrm{scm}(A, B) = fR \cdot sR$$

$$\text{Upon termination: } \frac{x \cdot u + x \cdot v}{2} = fR \cdot sR$$

...and try to invent a loop invariant.

$\left[\mathbb{N} \ni A \geq 1 \;\wedge\; \mathbb{N} \ni B \geq 1 \;\wedge\; \texttt{f(A,B)} \text{ is called.}\right]$

```
function [fR, sR] = f(x,y)
  u = x;
  v = y;
  while x < y || y < x
    if x < y
      y = y-x;
      v = v+u;
    else
      x = x-y;
      u = u+v;
    end
  end
  fR = (x+y)/2;
  sR = (u+v)/2;
```

$\left[\text{If the program terminates, } sR = \text{scm}(A, B).\right]$

$\left[\text{If the program terminates, } \texttt{fR} = \gcd(A, B)\right]$

Well, suppose we are right in both assertions.
Then, upon termination,

$$A \cdot B = \gcd(A, B) \cdot \text{scm}(A, B) = fR \cdot sR$$

$$\text{Upon termination: } \frac{y \cdot u + x \cdot v}{2} = fR \cdot sR$$

…and try to invent a loop invariant.

$\left[\mathbb{N} \ni A \geq 1 \;\wedge\; \mathbb{N} \ni B \geq 1 \;\wedge\; \texttt{f(A,B)} \text{ is called.}\right]$

```
function [fR, sR] = f(x,y)
  u = x;
  v = y;
  while x < y || y < x
    if x < y
      y = y-x;
      v = v+u;
    else
      x = x-y;
      u = u+v;
    end
  end
  fR = (x+y)/2;
  sR = (u+v)/2;
```

$\left[\text{If the program terminates, } \; sR = \mathrm{scm}(A,B).\right]$

$\left[\text{If the program terminates, } \; \texttt{fR} = \gcd(A,B)\right]$

Well, suppose we are right in both assertions.
Then, upon termination,

$$A \cdot B = \gcd(A,B) \cdot \mathrm{scm}(A,B) = fR \cdot sR$$

Upon termination: $\frac{y \cdot u + x \cdot v}{2} = fR \cdot sR$

Together: $A \cdot B = \frac{y \cdot u + x \cdot v}{2}$

Or: $2 \cdot A \cdot B = y \cdot u + x \cdot v$

Might $2 \cdot A \cdot B = y \cdot u + x \cdot v$ also be true during execution?

$\left[\mathbb{N} \ni A \geq 1 \ \wedge \ \mathbb{N} \ni B \geq 1 \ \wedge\ \texttt{f(A,B)} \text{ is called.}\right]$

```
function [fR, sR] = f(x,y)
  u = x;
  v = y;
  while x < y || y < x
    if x < y
      y = y-x;
      v = v+u;
    else
      x = x-y;
      u = u+v;
    end
  end
  fR = (x+y)/2;
  sR = (u+v)/2;
```

$\left[\text{If the program terminates, } \ sR = \text{scm}(A, B).\right]$

$\left[\text{If the program terminates, } \ \texttt{fR} = \gcd(A, B)\right]$

We conjecture that this is a loop invariant.

$A \cdot B = \gcd(A, B) \cdot \text{scm}(A, B) = fR \cdot sR$

Upon termination: $\frac{y \cdot u + x \cdot v}{2} = fR \cdot sR$

Together: $A \cdot B = \frac{y \cdot u + x \cdot v}{2}$

Or: $2 \cdot A \cdot B = y \cdot u + x \cdot v$

Yes.

$\left[\mathbb{N} \ni A \geq 1 \ \wedge\ \mathbb{N} \ni B \geq 1 \ \wedge\ \texttt{f(A,B)} \text{ is called.}\right]$

```
function [fR, sR] = f(x,y)
  u = x;
  v = y;
  while x < y || y < x
    if x < y
      y = y-x;
      v = v+u;
    else
      x = x-y;
      u = u+v;
    end
  end
  fR = (x+y)/2;
  sR = (u+v)/2;
```

$\left[\text{If the program terminates,}\ \ sR = \text{scm}(A, B).\right]$

$\left[\text{If the program terminates,}\ \ \texttt{fR} = \text{gcd}(A, B)\right]$

We conjecture that this is a loop invariant.

$A \cdot B = \text{gcd}(A, B) \cdot \text{scm}(A, B) = fR \cdot sR$

Upon termination: $\frac{y \cdot u + x \cdot v}{2} = fR \cdot sR$

Together: $A \cdot B = \frac{y \cdot u + x \cdot v}{2}$

Or: $2 \cdot A \cdot B = y \cdot u + x \cdot v$

Yes.

$[\mathbb{N} \ni A \geq 1 \ \wedge \ \mathbb{N} \ni B \geq 1 \ \wedge$ `f(A,B)` is called.$]$

```
function [fR, sR] = f(x,y)
  u = x;
  v = y;
  while x < y || y < x
    if x < y
      y = y-x;
      v = v+u;
    else
      x = x-y;
      u = u+v;
    end
  end
  fR = (x+y)/2;
  sR = (u+v)/2;
```

$[$If the program terminates, $sR = \mathrm{scm}(A, B)$.$]$

$[$If the program terminates, `fR` $= \gcd(A, B)]$

We conjecture that this is a loop invariant.

$A \cdot B = \gcd(A, B) \cdot \mathrm{scm}(A, B) = fR \cdot sR$

Upon termination: $\frac{y \cdot u + x \cdot v}{2} = fR \cdot sR$

Together: $A \cdot B = \frac{y \cdot u + x \cdot v}{2}$

Or: $2 \cdot A \cdot B = y \cdot u + x \cdot v$

Upon initialization,

$x = u = A$ and $y = v = B$

Yes.

$\left[\mathbb{N} \ni A \geq 1 \ \wedge\ \mathbb{N} \ni B \geq 1 \ \wedge\ \texttt{f(A,B)} \text{ is called.}\right]$

```
function [fR, sR] = f(x,y)
  u = x;
  v = y;
  while x < y || y < x
    if x < y
      y = y-x;
      v = v+u;
    else
      x = x-y;
      u = u+v;
    end
  end
  fR = (x+y)/2;
  sR = (u+v)/2;
```

$\left[\text{If the program terminates, } \ sR = \mathrm{scm}(A, B).\right]$

$\left[\text{If the program terminates, } \ \texttt{fR} = \mathrm{gcd}(A, B)\right]$

We conjecture that this is a loop invariant.

$$A \cdot B = \mathrm{gcd}(A, B) \cdot \mathrm{scm}(A, B) = fR \cdot sR$$

Upon termination: $\frac{y \cdot u + x \cdot v}{2} = fR \cdot sR$

Together: $A \cdot B = \frac{y \cdot u + x \cdot v}{2}$

Or: $2 \cdot A \cdot B = y \cdot u + x \cdot v$

So,

$$2 \cdot A \cdot B = y \cdot u + x \cdot v \Leftrightarrow 2 \cdot A \cdot B = 2 \cdot A \cdot B$$

Yes.

$\left[\mathbb{N} \ni A \geq 1 \ \wedge\ \mathbb{N} \ni B \geq 1 \ \wedge\ \texttt{f(A,B)} \text{ is called.}\right]$

```
function [fR, sR] = f(x,y)
  u = x;
  v = y;
  while x < y || y < x
    if x < y
      y = y-x;
      v = v+u;
    else
      x = x-y;
      u = u+v;
    end
  end
  fR = (x+y)/2;
  sR = (u+v)/2;
```

$\left[\text{If the program terminates, } sR = \text{scm}(A, B).\right]$

$\left[\text{If the program terminates, } \texttt{fR} = \text{gcd}(A, B)\right]$

We conjecture that this is a loop invariant.

$A \cdot B = \text{gcd}(A, B) \cdot \text{scm}(A, B) = fR \cdot sR$

Upon termination: $\frac{y \cdot u + x \cdot v}{2} = fR \cdot sR$

Together: $A \cdot B = \frac{y \cdot u + x \cdot v}{2}$

Or: $2 \cdot A \cdot B = y \cdot u + x \cdot v$

In the one branch,

$2 \cdot A \cdot B = y \cdot u + x \cdot v$ becomes

Yes.

$\left[\mathbb{N} \ni A \geq 1 \ \wedge \ \mathbb{N} \ni B \geq 1 \ \wedge \ \texttt{f(A,B)} \text{ is called.}\right]$

```
function [fR, sR] = f(x,y)
  u = x;
  v = y;
  while x < y || y < x
    if x < y
      y = y-x;
      v = v+u;
    else
      x = x-y;
      u = u+v;
    end
  end
  fR = (x+y)/2;
  sR = (u+v)/2;
```

$\left[\text{If the program terminates, } sR = \text{scm}(A, B).\right]$

$\left[\text{If the program terminates, } \texttt{fR} = \text{gcd}(A, B)\right]$

We conjecture that this is a loop invariant.

$A \cdot B = \text{gcd}(A, B) \cdot \text{scm}(A, B) = fR \cdot sR$

Upon termination: $\frac{y \cdot u + x \cdot v}{2} = fR \cdot sR$

Together: $A \cdot B = \frac{y \cdot u + x \cdot v}{2}$

Or: $2 \cdot A \cdot B = y \cdot u + x \cdot v$

In the one branch,

$2 \cdot A \cdot B = y \cdot u + x \cdot v$ becomes

$2 \cdot A \cdot B = (y - x) \cdot u + x \cdot (v + u)$

Yes.

[$\mathbb{N} \ni A \geq 1 \ \wedge\ \mathbb{N} \ni B \geq 1 \ \wedge$ `f(A,B)` is called.]

```
function [fR, sR] = f(x,y)
  u = x;
  v = y;
  while x < y || y < x
    if x < y
      y = y-x;
      v = v+u;
    else
      x = x-y;
      u = u+v;
    end
  end
  fR = (x+y)/2;
  sR = (u+v)/2;
```

[If the program terminates, $sR = \mathrm{scm}(A, B)$.]

[If the program terminates, `fR` $= \gcd(A, B)$]

We conjecture that this is a loop invariant.

$$A \cdot B = \gcd(A, B) \cdot \mathrm{scm}(A, B) = fR \cdot sR$$

Upon termination: $\frac{y \cdot u + x \cdot v}{2} = fR \cdot sR$

Together: $A \cdot B = \frac{y \cdot u + x \cdot v}{2}$

Or: $2 \cdot A \cdot B = y \cdot u + x \cdot v$

In the other branch,

$2 \cdot A \cdot B = y \cdot u + x \cdot v$ becomes

Yes.

$\left[\mathbb{N} \ni A \geq 1 \ \wedge \ \mathbb{N} \ni B \geq 1 \ \wedge \ \texttt{f(A,B)} \text{ is called.}\right]$

```
function [fR, sR] = f(x,y)
  u = x;
  v = y;
  while x < y || y < x
    if x < y
      y = y-x;
      v = v+u;
    else
      x = x-y;
      u = u+v;
    end
  end
  fR = (x+y)/2;
  sR = (u+v)/2;
```

$\left[\text{If the program terminates, } \ sR = \text{scm}(A, B).\right]$

$\left[\text{If the program terminates, } \ \texttt{fR} = \text{gcd}(A, B)\right]$

We conjecture that this is a loop invariant.

$A \cdot B = \text{gcd}(A, B) \cdot \text{scm}(A, B) = fR \cdot sR$

Upon termination: $\frac{y \cdot u + x \cdot v}{2} = fR \cdot sR$

Together: $A \cdot B = \frac{y \cdot u + x \cdot v}{2}$

Or: $2 \cdot A \cdot B = y \cdot u + x \cdot v$

In the other branch,

$2 \cdot A \cdot B = y \cdot u + x \cdot v$ becomes

$2 \cdot A \cdot B = y \cdot (u + v) + (x - y) \cdot v$

Yes.

$\left[\mathbb{N} \ni A \geq 1 \ \wedge \ \mathbb{N} \ni B \geq 1 \ \wedge\ \texttt{f(A,B)} \text{ is called.}\right]$

```
function [fR, sR] = f(x,y)
  u = x;
  v = y;
  while x < y || y < x
    if x < y
      y = y-x;
      v = v+u;
    else
      x = x-y;
      u = u+v;
    end
  end
  fR = (x+y)/2;
  sR = (u+v)/2;
```

$\left[\text{If the program terminates, } \ sR = \text{scm}(A, B).\right]$

$\left[\text{If the program terminates, } \ \texttt{fR} = \text{gcd}(A, B)\right]$

We conjecture that this is a loop invariant.

$A \cdot B = \text{gcd}(A, B) \cdot \text{scm}(A, B) = fR \cdot sR$

Upon termination: $\frac{y \cdot u + x \cdot v}{2} = fR \cdot sR$

Together: $A \cdot B = \frac{y \cdot u + x \cdot v}{2}$

Or: $2 \cdot A \cdot B = y \cdot u + x \cdot v$

Thus, the equality is maintained.

Yes.

[$\mathbb{N} \ni A \geq 1 \ \wedge\ \mathbb{N} \ni B \geq 1 \ \wedge$ `f(A,B)` is called.]

```
function [fR, sR] = f(x,y)
  u = x;
  v = y;
  while x < y || y < x
    if x < y
      y = y-x;
      v = v+u;
    else
      x = x-y;
      u = u+v;
    end
  end
  fR = (x+y)/2;
  sR = (u+v)/2;
```

[If the program terminates, $sR = \text{scm}(A, B)$.]

[If the program terminates, $\texttt{fR} = \gcd(A, B)$]

We conjecture that this is a loop invariant.

$$A \cdot B = \gcd(A, B) \cdot \text{scm}(A, B) = fR \cdot sR$$

Upon termination: $\frac{y \cdot u + x \cdot v}{2} = fR \cdot sR$

Together: $A \cdot B = \frac{y \cdot u + x \cdot v}{2}$

Or: $2 \cdot A \cdot B = y \cdot u + x \cdot v$

Upon completion,

$x = y = \gcd(A, B)$, thus

Yes.

$[\mathbb{N} \ni A \geq 1 \ \wedge\ \mathbb{N} \ni B \geq 1 \ \wedge$ `f(A,B)` is called.$]$

```
function [fR, sR] = f(x,y)
  u = x;
  v = y;
  while x < y || y < x
    if x < y
      y = y-x;
      v = v+u;
    else
      x = x-y;
      u = u+v;
    end
  end
  fR = (x+y)/2;
  sR = (u+v)/2;
```

$[$If the program terminates, $sR = \mathrm{scm}(A,B)$.$]$

$[$If the program terminates, `fR` $= \mathrm{gcd}(A,B)]$

We conjecture that this is a loop invariant.

$A \cdot B = \mathrm{gcd}(A,B) \cdot \mathrm{scm}(A,B) = fR \cdot sR$

Upon termination: $\frac{y \cdot u + x \cdot v}{2} = fR \cdot sR$

Together: $A \cdot B = \frac{y \cdot u + x \cdot v}{2}$

Or: $2 \cdot A \cdot B = y \cdot u + x \cdot v$

Upon completion,

$x = y = \mathrm{gcd}(A,B)$, thus

$2 \cdot A \cdot B = y \cdot u + x \cdot v = \mathrm{gcd}(A,B) \cdot (u+v)$

Yes.

$[\mathbb{N} \ni A \geq 1 \ \wedge \ \mathbb{N} \ni B \geq 1 \ \wedge$ `f(A,B)` is called.$]$

```
function [fR, sR] = f(x,y)
  u = x;
  v = y;
  while x < y || y < x
    if x < y
      y = y-x;
      v = v+u;
    else
      x = x-y;
      u = u+v;
    end
  end
  fR = (x+y)/2;
  sR = (u+v)/2;
```

$[$If the program terminates, $sR = \text{scm}(A, B)$.$]$

$[$If the program terminates, fR $= \gcd(A, B)$.$]$

We conjecture that this is a loop invariant.

$A \cdot B = \gcd(A, B) \cdot \text{scm}(A, B) = fR \cdot sR$

Upon termination: $\frac{y \cdot u + x \cdot v}{2} = fR \cdot sR$

Together: $A \cdot B = \frac{y \cdot u + x \cdot v}{2}$

Or: $2 \cdot A \cdot B = y \cdot u + x \cdot v$

Therefore

$\text{sR} = \frac{u+v}{2} = \frac{A \cdot B}{\gcd(A,B)} = \text{scm}(A, B)$ □