Testing vs. Proof

	Testing (dynamic analysis)	**Proof (static analysis)**
Properties	observable	any
Verified execution traces	finite subset	exhaustive
Guaranteeing Absence of errors	No	Yes
Finding infeasible errors	No	Yes
Developed with	automated regression	automated proof checkers
Practicality	high	limited

Testing vs. Proof

Proofs are not practical, but we learn them because they:

- tell us how to think about program correctness.
- are important for development, inspection.
- give us intuitions about foundations of automated theorem-provers.
  - Metis[1]
  - Prover9[2]

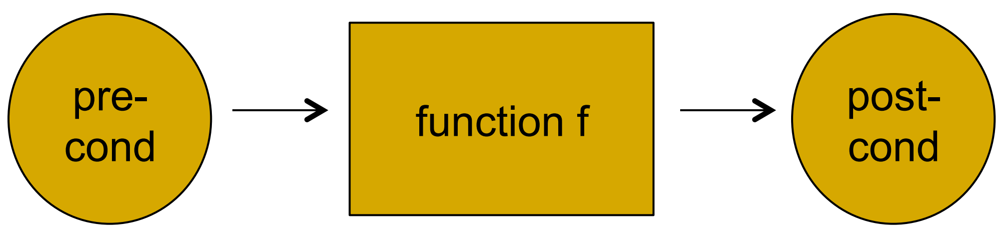
[1]http://www.gilith.com/software/metis/index.html
[2]https://www.cs.unm.edu/~mccune/mace4/

Correctness of a Function

Contract between client and implementation: starting from an *initial condition*, function f should establish a *certain condition after correctly running* (input-output claim).

- *Pre-condition & Post-condition* extracted and expressed by predicates (boolean function over program state).

- Function f is correct w.r.t the specification only if:



Correctness of a Program

- **Partial Correctness**: if pre-condition holds and the program terminates, post-condition holds.

- **Total Correctness**: if pre-condition holds, the program terminates and post-condition holds.

Hoare's Logic: Syntax & Semantics

A Hoare triple $\{\varphi_1\}P\{\varphi_2\}$ is a formula, where φ_1 and φ_2 are predicates and P is a program.

- Partial correctness $\Leftrightarrow$ starting from a state s satisfying φ_1, whenever an execution of P terminates in state s', then s' $\models \varphi_2$.
- Total correctness $\Leftrightarrow$ starting from a state s satisfying φ_1, every execution of P terminates and whenever an execution of P terminates in state s', then s' $\models \varphi_2$.
- For programs without loops, both semantics coincide.

Hoare's Logic: Syntax & Semantics

- Strongest Post-conditions:
  - If $\{\varphi_1\}P\{\varphi_2\}$ and for all φ'_2 such that $\models\{\varphi_1\}P\{\varphi'_2\}$, $\varphi_2 \Rightarrow \varphi'_2$, then φ_2 is the strongest post-condition of P with respect to φ_1.

1. `{x=5}    x:=x*2      {true}`
2. `{x=5}    x:=x*2      {x>0}`
3. `{x=5}    x:=x*2      {x=10∨x=5}`
4. `{x=5}    x:=x*2      {x=10}`

Hoare's Logic: Syntax & Semantics

- Weakest Preconditions:
  - If $\{\varphi_1\}P\{\varphi_2\}$ and for all φ'_1 such that $\{\varphi'_1\}$ P $\{\varphi_2\}$, $\varphi'_1 \Rightarrow \varphi_1$, then φ_1 is the weakest precondition of P with respect to φ_2.

1. `{x=5 && y=10}        z := x/y   {z<1}`
2. `{x<y && y>0}         z := x/y   {z<1}`
3. `{y≠0 && x/y<1}       z := x/y   {z<1}`

Hoare's Logic: Axioms

- Assignment:

  $$\{\varphi[t/y]\}\ y:=t\ \{\varphi\}$$

Hoare's Logic: Axioms

- Assignment:

  `{φ[t/y]} y:=t {φ}`

1. `{?} y:=y+5 {y=10}`

Hoare's Logic: Axioms

- Assignment:

$$\{\varphi[t/y]\}\ y{:=}t\ \{\varphi\}$$

1. $\{?\}\ y{:=}y+5\ \{y=10\}$

   $\{?\}\ y_{new}{:=}y_{old}+5\ \{y_{new}=10\}$

   $Y_{old} + 5 = 10$

Hoare's Logic: Axioms

- Assignment:

$$\{\varphi[t/y]\}\ y:=t\ \{\varphi\}$$

1. $\{y+5=10\}\ y:=y+5\ \{y=10\}$

   $\{?\}\ y_{new}:=y_{old}+5\ \{y_{new}=10\}$

   $Y_{old} + 5 = 10$

Hoare's Logic: Axioms

- Assignment:

$$\{\varphi[t/y]\}\ y:=t\ \{\varphi\}$$

2. `{?} x:=y {x+y<z}`

Hoare's Logic: Axioms

- Assignment:

  $$\{\varphi[t/y]\}\ y{:=}t\ \{\varphi\}$$

2. $\{?\}\ x{:=}y\ \{x+y<z\}$

   $\{?\}\ x_{new}{:=}y\ \{x_{new}+y<z\}$

   $y+y<z$

Hoare's Logic: Axioms

- Assignment:

$$\{\varphi[t/y]\}\ y{:=}t\ \{\varphi\}$$

2. $\{y+y<z\}\ x{:=}y\ \{x+y<z\}$

   $\{?\}\ x_{new}{:=}y\ \{x_{new}+y<z\}$

   $y+y<z$

Hoare's Logic: Axioms

- Assignment:

  `{φ[t/y]} y:=t {φ}`

3. `{?} y:=2*(y+5) {y>20}`

Hoare's Logic: Axioms

- Assignment:

$$\{\varphi[t/y]\}\ y{:=}t\ \{\varphi\}$$

3. $\{?\}\ y{:=}2*(y+5)\ \{y>20\}$

   $\{?\}\ y_{new}{:=}2*(y_{old}+5)\ \{y_{new}>20\}$

   $2*(y_{old}+5) > 20$

Hoare's Logic: Axioms

- Assignment:

$$\{\varphi[t/y]\}\ y{:=}t\ \{\varphi\}$$

3. $\{y>5\}\ y{:=}2*(y+5)\ \{y>20\}$

   $\{?\}\ y_{new}{:=}2*(y_{old}+5)\ \{y_{new}>20\}$

   $2*(y_{old}+5) > 20$

Hoare's Logic: Axioms

- Assignment:

Axioms work backwards.

3. `{y>5} y:=2*(y+5) {?}`

   $\{y_{old}>5\}\ y_{new}:=2*(y_{old}+5)\ \{?\}$

   $y_{old}>5 \Rightarrow y_{new} > 20$

Hoare's Logic: Axioms

- Assignment:

Axioms work backwards.

3. $\{y>5\}$ $y:=2*(y+5)$ $\{y_{new}>20\}$

   $\{y_{old}>5\}$ $y_{new}:=2*(y_{old}+5)$ $\{?\}$

   $y_{old}>5 \Rightarrow y_{new} > 20$

Hoare's Logic: Axioms

- Composition rule:

$$\frac{\{\varphi\}\ \mathtt{P1}\ \{\varphi'\},\{\varphi'\}\ \mathtt{P2}\ \{\varphi''\}}{\{\varphi\}\ \mathtt{P1;P2}\ \{\varphi''\}}$$

$$\frac{\{x+1=y+2\}\ x:=x+1\ \{x=y+2\},\quad \{x=y+2\}\ y:=y+2\ \{x=y\}}{\{x+1=y+2\}\ x:=x+1;\ y:=y+2\{x=y\}}$$

Hoare's Logic: Axioms

- Condition rule:

$$\frac{\{\varphi \wedge c\}\ P1\ \{\varphi'\},\ \{\varphi \wedge \neg c\}\ P2\ \{\varphi'\}}{\{\varphi\}\ \texttt{if}\ c\ \texttt{then}\ P1\ \texttt{else}\ P2\ \texttt{fi}\ \{\varphi'\}}$$

$$\frac{\{(y>4)\wedge(z>1)\}y:=y+z\{y>3\}\ ,\ \{(y>4)\wedge\neg(z>1)\}y:=y-1\{y>3\}}{\{y>4\}\ \texttt{if}\ (z>1)\ \texttt{then}\ y:=\ y+z\ \texttt{else}\ y:=y-1\{y>3\}}$$

Hoare's Logic: Axioms

- Consequence rule:

$$\frac{\varphi \rightarrow \sigma,\ \{\sigma\}\ P\ \{\sigma'\},\ \sigma' \rightarrow \varphi'}{\{\varphi\}\ P\ \{\varphi'\}}$$

$$\frac{(y>4 \wedge z>1) \Rightarrow (y+z>5),\ \{y+z>5\}y:=y+z\{y>5\},\ (y>5) \Rightarrow (y>3)}{\{(y>4) \wedge (z>1)\}y:=y+z\{y>3\}}$$

Hoare's Logic: Axioms

- Consequence rule:

$$\frac{\varphi \rightarrow \sigma,\ \{\sigma\}\ P\ \{\sigma'\},\ \sigma' \rightarrow \varphi'}{\{\varphi\}\ P\ \{\varphi'\}}$$

$$\frac{(y>4 \land z>1) \Rightarrow (y+z>5),\ \{y+z>5\}\,y:=y+z\,\{y>5\},\ (y>5) \Rightarrow (y>3)}{\{(y>4) \land (z>1)\}\,y:=y+z\,\{y>3\}}$$

weakest pre-condition

strongest post-condition

Hoare's Logic: Axioms

- While rule:

$$\frac{\{\varphi \wedge c\}\ P\ \{\varphi\}}{\{\varphi\}\ \texttt{while}\ c\ \texttt{do}\ P\ \texttt{od}\ \{\varphi \wedge \neg c\}}$$

$$\frac{\{(y=x+z)\wedge(z!=0)\}x:=x+1;z:=z-1\{y=x+z\}}{\{y=x+z\}\texttt{while}(z!=0)x:=x+1;z:=z-1\{(y=x+z)\wedge(z=0)\}}$$

$$\frac{\{(y=x+z)\wedge \texttt{true}\}x:=x+1;z:=z-1\{y=x+z\}}{\{y=x+z\}\texttt{while}(\texttt{true})x:=x+1;z:=z-1\{(y=x+z)\wedge \texttt{false}\}}$$

Hoare's Logic: Axioms

- For correctness proof of a loop, we need an invariant (`Inv`) such that:
  - It is initially true:

    $$\varphi \Rightarrow \mathtt{Inv}$$

  - Partial correctness : each execution of the loop preserves the invariant:

    $$\{\ \mathtt{Inv} \wedge \mathtt{c}\ \}\ \mathtt{P}\ \{\ \mathtt{Inv}\ \}$$

  - Total correctness: the invariant and the loop exit condition imply the post-condition:

    $$\{\ \mathtt{Inv} \wedge \neg \mathtt{c}\ \} \Rightarrow \{\ \varphi \wedge \neg \mathtt{c}\ \}$$

Hoare's Logic: Axioms

```
for i from 1 to n
```

$\Rightarrow$ Inv $= \Sigma_{j=1\ldots i-1}\ j$

```
        temp += i
```

$\Rightarrow$ Inv $= \Sigma_{j=1\ldots i}\ j$

```
i = 1; temp = 0;
while (i<n) {
    temp+=i;
    i++;
}
```

Inv: $\sum_{j=1}^{i-1} j = temp$

Hoare's Logic: Structural rules

Conjunction

$$\frac{\{\phi_1\}P\{\psi_1\}\quad\{\phi_2\}P\{\psi_2\}}{\{\phi_1 \wedge \phi_2\}P\{\psi_1 \wedge \psi_2\}}$$

Disjunction

$$\frac{\{\phi_1\}P\{\psi_1\}\quad\{\phi_2\}P\{\psi_2\}}{\{\phi_1 \vee \phi_2\}P\{\psi_1 \vee \psi_2\}}$$

Universal
~~Existential~~ Quantification

$$\frac{\{\phi\}P\{\psi\}}{\{\forall v.\phi\}P\{\forall v.\psi\}}$$

Existential
~~Universal~~ Quantification

$$\frac{\{\phi\}P\{\psi\}}{\{\exists v.\phi\}P\{\exists v.\psi\}}$$

PARTIAL CORRECTNESS

Ex.1

- Square root of a positive integer.

```
{x > 0} =pre
if (x == 1) sqrt := 1
else
      i:=1;
      z:=1;
      while ( z <= x )
            i++;
            z:=i*i;
      sqrt := i-1;
{sqrt²≤ x < (sqrt + 1)² }=post
```

Ex.1

- Square root of a positive integer.

```
{x > 0} =pre
if (x == 1) sqrt := 1
else
      i:=1;
      z:=1;
      while ( z <= x )
            i++;
            z:=i*i;
      sqrt := i-1;
{sqrt²≤ x < (sqrt + 1)² }=post
```

Ex.1

- Square root of a positive integer.

```
{x > 0} =pre
if (x == 1) sqrt := 1
else
      i:=1;
      z:=1;
      while ( z <= x )
            i++;
            z:=i*i;
      sqrt := i-1;
{sqrt²≤ x < (sqrt + 1)² }=post
```

According to condition rule:

1. $\{x > 0 \wedge x = 1\}$ sqrt :=1 {post}
2. $\{x > 0 \wedge x != 1\}$ i:=1;… {post}

Ex.1

- Square root of a positive integer.

```
{x > 0} =pre
if (x == 1) sqrt := 1
else
      i:=1;
      z:=1;
      while ( z <= x )
            i++;
            z:=i*i;
      sqrt := i-1;
{sqrt²≤ x < (sqrt + 1)² }=post
```

According to condition rule:

1. $\{x > 0 \wedge x = 1\}$ `sqrt :=1` $\{post\}$
2. $\{x > 0 \wedge x != 1\}$ `i:=1;…` $\{post\}$

Ex.1

```
{x > 0 ∧ x =1} sqrt :=1 {post}
```

Ex.1

$\{x > 0 \wedge x = 1\}$ `sqrt :=1` $\{post\}$

$\{x > 0 \wedge x = 1\}$ `sqrt :=1` $\{sqrt^2 \leq x < (sqrt + 1)^2\}$

Ex.1

{x > 0 ∧ x =1} sqrt :=1 {post}

{x > 0 ∧ x =1} sqrt :=1 {$sqrt^2 \leq x < (sqrt + 1)^2$}

Ex.1

$\{x > 0 \wedge x = 1\}$ sqrt := 1 {post}

$\{x > 0 \wedge x = 1\}$ sqrt := 1 $\{sqrt^2 \le x < (sqrt + 1)^2\}$

$\{x = 1\} \Rightarrow \{1 \le x < (1+1)^2\}$

Ex.1

$\{x > 0 \wedge x = 1\}$ sqrt := 1 $\{post\}$

$\{x > 0 \wedge x = 1\}$ sqrt := 1 $\{sqrt^2 \le x < (sqrt + 1)^2\}$

$\{x = 1\} \quad \Rightarrow \quad \{1 \le x < (1+1)^2\}$

$\{x = 1\} \quad \Rightarrow \quad \{1 \le x < 4\}$

Ex.1

$\{x > 0 \wedge x = 1\}$ sqrt := 1 $\{post\}$

$\{x > 0 \wedge x = 1\}$ sqrt := 1 $\{sqrt^2 \leq x < (sqrt + 1)^2\}$

$\{x = 1\} \quad \Rightarrow \quad \{1 \leq x < (1+1)^2\}$

$\{x = 1\} \quad \Rightarrow \quad \{1 \leq x < 4\}$

always holds

Ex.1

- Square root of a positive integer.

```
{x > 0} =pre
if (x == 1) sqrt := 1
else
      i:=1;
      z:=1;
      while ( z <= x )
            i++;
            z:=i*i;
      sqrt := i-1;
{sqrt²≤ x < (sqrt + 1)² }=post
```

According to condition rule:

```
1. {x > 0 ∧ x =1} sqrt :=1 {post}
2. {x > 0 ∧ x!=1} i:=1;… {post}
```

Ex.1

- Square root of a positive integer.

```
{x > 0} =pre
if (x == 1) sqrt := 1
else
      i:=1;
      z:=1;
      while ( z <= x )
            i++;
            z:=i*i;
      sqrt := i-1;
{sqrt²≤ x < (sqrt + 1)² }=post
```

According to condition rule:

1. {x > 0 ∧ x =1} sqrt :=1 {post}
2. {x > 0 ∧ x!=1} i:=1;… {post}
   a. {x > 1} i:=1;z:=1; {Inv}
   b. {Inv} while … {Inv ∧ z > x }
   c. {Inv ∧ z>x} sqrt:=i–1;{post}

Ex.1

- Square root of a positive integer.

```
{x > 0} =pre
if (x == 1) sqrt := 1
else
      i:=1;
      z:=1;
      while ( z <= x )
            i++;
            z:=i*i;
      sqrt := i-1;
{sqrt²≤ x < (sqrt + 1)² }=post
```

According to condition rule:

```
1. {x > 0 ∧ x =1} sqrt :=1 {post}
2. {x > 0 ∧ x!=1} i:=1;… {post}
   a. {x > 1} i:=1;z:=1; {Inv}
   b. {Inv} while … {Inv ∧ z > x }
   c. {Inv ∧ z>x} sqrt:=i–1;{post}
```

Ex.1

```
{x > 1} i:=1;z:=1; {Inv}
```

Ex.1

`{x > 1} i:=1;z:=1; {Inv}`

Let's choose an `Inv`:

Ex.1

```
{x > 1} i:=1;z:=1; {Inv}
```

Let’s choose an `Inv`:

- what does each iteration do?

Ex.1

`{x > 1} i:=1;z:=1; {Inv}`

Let's choose an `Inv`:

- what does each iteration do? $z = i^2$

Ex.1

`{x > 1} i:=1;z:=1; {Inv}`

Let's choose an `Inv`:

- what does each iteration do? $z = i^2$
- what keeps the loop going on?

Ex.1

`{x > 1} i:=1;z:=1; {Inv}`

Let's choose an `Inv`:

- what does each iteration do? $z = i^2$
- what keeps the loop going on? $(i-1)^2 \leq x$

Ex.1

$\{x > 1\}$ `i:=1;z:=1;` $\{Inv\}$

Let's choose an `Inv`:

- what does each iteration do? $z = i^2$
- what keeps the loop going on? $(i-1)^2 \le x$

Hence: $Inv = \{z = i^2 \wedge (i-1)^2 \le x\}$

Ex.1

$\{x > 1\}$ `i:=1;z:=1;` $\{Inv\}$

Let's choose an `Inv`:

- what does each iteration do? $z = i^2$
- what keeps the loop going on? $(i-1)^2 \leq x$

Hence: $Inv = \{z = i^2 \wedge (i-1)^2 \leq x\}$

$\{x > 1\}$ `i:=1;z:=1;` $\{z = i^2 \wedge (i-1)^2 \leq x\}$

$\{x > 1\} \quad \Rightarrow \quad \{1 = 1 \wedge 0 \leq x\}$

Ex.1

`{x > 1} i:=1;z:=1; {Inv}`

Let's choose an `Inv`:

- what does each iteration do? $z = i^2$
- what keeps the loop going on? $(i-1)^2 \le x$

Hence: $Inv = \{z = i^2 \wedge (i-1)^2 \le x\}$

$\{x > 1\}$ `i:=1;z:=1;` $\{z = i^2 \wedge (i-1)^2 \le x\}$

$\{x > 1\} \quad \Rightarrow \quad \{1 = 1 \wedge 0 \le x\}$

$\{x > 1\} \quad \Rightarrow \quad \{ x \ge 0\}$

Ex.1

```
{x > 1} i:=1;z:=1; {Inv}
```

Let's choose an `Inv`:

- what does each iteration do? $z = i^2$
- what keeps the loop going on? $(i-1)^2 \le x$

Hence: $Inv = \{z = i^2 \wedge (i-1)^2 \le x\}$

$\{x > 1\}$ i:=1;z:=1; $\{z = i^2 \wedge (i-1)^2 \le x\}$

$\{x > 1\} \quad \Rightarrow \quad \{1 = 1 \wedge 0 \le x\}$

$\{x > 1\} \quad \Rightarrow \quad \{ x \ge 0\}$

always holds

Ex.1

- Square root of a positive integer.

```
{x > 0} =pre
if (x == 1) sqrt := 1
else
      i:=1;
      z:=1;
      while ( z <= x )
            i++;
            z:=i*i;
      sqrt := i-1;
{sqrt²≤ x < (sqrt + 1)² }=post
```

According to condition rule:

```
1. {x > 0 ∧ x =1} sqrt :=1 {post}
2. {x > 0 ∧ x!=1} i:=1;… {post}
   a. {x > 1} i:=1;z:=1; {Inv}
   b. {Inv} while … {Inv ∧ z > x }
   c. {Inv ∧ z>x} sqrt:=i–1;{post}
```

Ex.1

```
{Inv} while … {Inv ∧ z > x }
```

Ex.1

`{Inv} while … {Inv ∧ z > x }`

According to while rule:

`{Inv ∧ z ≤ x } i++; z = i*i; {Inv}`

Ex.1

`{Inv} while … {Inv ∧ z > x }`

According to while rule:

`{Inv ∧ z ≤ x } i++; z = i*i; {Inv}`

$\{z = i^2 \wedge (i-1)^2 \leq x \wedge z \leq x\}$ `i++; z = i*i;` $\{z = i^2 \wedge (i-1)^2 \leq x\}$

Ex.1

```
{Inv} while … {Inv ∧ z > x }
```

According to while rule:

```
{Inv ∧ z ≤ x } i++; z = i*i; {Inv}
```

$\{z = i^2 \wedge (i-1)^2 \le x \wedge z \le x\}$ `i++; z = i*i;` $\{z = i^2 \wedge (i-1)^2 \le x\}$

Ex.1

$\{Inv\}$ `while` … $\{Inv \land z > x\}$

According to while rule:

$\{Inv \land z \le x\}$ `i++; z = i*i;` $\{Inv\}$

$\{z = i^2 \land (i-1)^2 \le x \land z \le x\}$ `i++; z = i*i;` $\{z = i^2 \land (i-1)^2 \le x\}$

$\{(i-1)^2 \le x \land i^2 \le x\}$ `i++;` $\{i^2 = i^2 \land (i-1)^2 \le x\}$

Ex.1

$\{Inv\}$ while … $\{Inv \land z > x\}$

According to while rule:

$\{Inv \land z \le x\}$ i++; z = i*i; $\{Inv\}$

$\{z = i^2 \land (i-1)^2 \le x \land z \le x\}$ i++; z = i*i; $\{z = i^2 \land (i-1)^2 \le x\}$

$\{(i-1)^2 \le x \land i^2 \le x\}$ i++;$\{i^2 = i^2 \land (i-1)^2 \le x\}$

$\{i^2 \le x\}$ i++; $\{(i-1)^2 \le x\}$

Ex.1

$\{Inv\}$ while … $\{Inv \land z > x\}$

According to while rule:

$\{Inv \land z \le x\}$ i++; z = i*i; $\{Inv\}$

$\{z = i^2 \land (i-1)^2 \le x \land z \le x\}$ i++; z = i*i; $\{z = i^2 \land (i-1)^2 \le x\}$

$\{(i-1)^2 \le x \land i^2 \le x\}$ i++;$\{i^2 = i^2 \land (i-1)^2 \le x\}$

$\{i^2 \le x\}$ i++; $\{(i-1)^2 \le x\}$

$\{i^2 \le x\} \Rightarrow \{i^2 \le x\}$

equal sides

Ex.1

- Square root of a positive integer.

```
{x > 0} =pre
if (x == 1) sqrt := 1
else
      i:=1;
      z:=1;
      while ( z <= x )
            i++;
            z:=i*i;
      sqrt := i-1;
{sqrt²≤ x < (sqrt + 1)² }=post
```

According to condition rule:

```
1. {x > 0 ∧ x =1} sqrt :=1 {post}
2. {x > 0 ∧ x!=1} i:=1;… {post}
   a. {x > 1} i:=1;z:=1; {Inv}
   b. {Inv} while … {Inv ∧ z > x }
   c. {Inv ∧ z>x} sqrt:=i–1;{post}
```

Ex.1

```
{Inv ∧ z>x} sqrt:=i–1;{post}
```

Ex.1

$\{Inv \wedge z>x\}$ `sqrt:=i–1;` $\{post\}$

$\{z = i^2 \wedge (i-1)^2 \leq x \wedge z>x\}$ `sqrt:=i–1;` $\{post\}$

Ex.1

```
{Inv ∧ z>x} sqrt:=i–1;{post}
{z = i² ∧ (i-1)²≤ x ∧ z>x} sqrt:=i–1;{post}
{(i-1)²≤ x < i² } sqrt:=i–1;{post}
```

$\{sqrt^2 \leq x < (sqrt+1)^2\}$

$\{(i-1)^2 \leq x < i^2\}$

equivalentes

Ex.2

- Least common multiple of two positive integers.

```
{x,y > 0} = pre
Z := 1
While (z % x != 0 ∨ z % y != 0 )
      z ++;
{z%x=0 ∧ z%y=0 ∧ ∀i(1≤i<z =>i%x!=0 ∨ i%y!=0)} = post
```

Ex.2

- Least common multiple of two positive integers.

```
{x,y > 0} = pre
Z := 1
While (z % x != 0 ∨ z % y != 0 )
      z ++;
{z%x=0 ∧ z%y=0 ∧ ∀i(1≤i<z =>i%x!=0 ∨ i%y!=0)} = post
```

According to while rule:

```
1. {x,y > 0} Z := 1 {Inv}
2. {Inv} while …{Inv ∧ !(z % x != 0 ∨ z % y != 0)}
3. {Inv ∧ !(z % x != 0 ∨ z % y != 0)} => {post}
```

Ex.2

- Least common multiple of two positive integers.

```
{x,y > 0} = pre
Z := 1
While (z % x != 0 ∨ z % y != 0 )
      z ++;
{z%x=0 ∧ z%y=0 ∧ ∀i(1≤i<z =>i%x!=0 ∨ i%y!=0)} = post
```

According to while rule:

```
1. {x,y > 0} Z := 1 {Inv}
2. {Inv} while …{Inv ∧ !(z % x != 0 ∨ z % y != 0)}
3. {Inv ∧ !(z % x != 0 ∨ z % y != 0)} => {post}
```

Ex.2

```
{x,y > 0} Z := 1 {Inv}
```

Ex.2

`{x,y > 0} Z := 1 {Inv}`

What could be a proper `Inv`?

Ex.2

`{x,y > 0} Z := 1 {Inv}`

What could be a proper `Inv?`

- what does each iteration do? Nothing really

Ex.2

```
{x,y > 0} Z := 1 {Inv}
```

What could be a proper `Inv?`

- what does each iteration do? Nothing really
- what keeps the loop going on?

Ex.2

`{x,y > 0} Z := 1 {Inv}`

What could be a proper `Inv?`

- what does each iteration do? Nothing really
- what keeps the loop going on? `∀i(1≤i<z =>i%x!=0 ∨ i%y!=0)`

Ex.2

`{x,y > 0} Z := 1 {Inv}`

What could be a proper `Inv`?

- what does each iteration do? Nothing really
- what keeps the loop going on? `∀i(1≤i<z =>i%x!=0 ∨ i%y!=0)`

`Hence, Inv = {∀i(1≤i<z =>i%x!=0 ∨ i%y!=0)}.`

Ex.2

`{x,y > 0} Z := 1 {Inv}`

What could be a proper `Inv`?

- what does each iteration do? Nothing really
- what keeps the loop going on? `∀i(1≤i<z =>i%x!=0 ∨ i%y!=0)`

`Hence, Inv = {∀i(1≤i<z =>i%x!=0 ∨ i%y!=0)}.`

`{x,y > 0} Z := 1 {∀i(1≤i<z =>i%x!=0 ∨ i%y!=0)}`

Ex.2

`{x,y > 0} Z := 1 {Inv}`

What could be a proper `Inv`?

- what does each iteration do? Nothing really
- what keeps the loop going on? `∀i(1≤i<z =>i%x!=0 ∨ i%y!=0)`

`Hence, Inv = {∀i(1≤i<z =>i%x!=0 ∨ i%y!=0)}.`

`{x,y > 0} Z := 1 {∀i(1≤i<z =>i%x!=0 ∨ i%y!=0)}`

Ex.2

`{x,y > 0} Z := 1 {Inv}`

What could be a proper `Inv?`

- what does each iteration do? Nothing really
- what keeps the loop going on? $\forall i(1 \le i < z \Rightarrow i\%x != 0 \lor i\%y != 0)$

Hence, Inv = $\{\forall i(1 \le i < z \Rightarrow i\%x != 0 \lor i\%y != 0)\}$.

$\{x,y > 0\}\ Z := 1\ \{\forall i(1 \le i < z \Rightarrow i\%x != 0 \lor i\%y != 0)\}$

$\{x,y > 0\} \Rightarrow \{\forall i(1 \le i < 1 \Rightarrow i\%x != 0 \lor i\%y != 0)\}$

Ex.2

`{x,y > 0} Z := 1 {Inv}`

What could be a proper `Inv?`

- what does each iteration do? Nothing really
- what keeps the loop going on? $\forall$`i(1≤i<z =>i%x!=0 ∨ i%y!=0)`

Hence, `Inv = {`$\forall$`i(1≤i<z =>i%x!=0 ∨ i%y!=0)}.`

`{x,y > 0} Z := 1 {`$\forall$`i(1≤i<z =>i%x!=0 ∨ i%y!=0)}`

`{x,y > 0} => {`$\forall$`i(1≤i<1 =>i%x!=0 ∨ i%y!=0)}`

false

Ex.2

`{x,y > 0} Z := 1 {Inv}`

What could be a proper `Inv`?

- what does each iteration do? Nothing really
- what keeps the loop going on? `∀i(1≤i<z =>i%x!=0 ∨ i%y!=0)`

`Hence, Inv = {∀i(1≤i<z =>i%x!=0 ∨ i%y!=0)}.`

`{x,y > 0} Z := 1 {∀i(1≤i<z =>i%x!=0 ∨ i%y!=0)}`

`{x,y > 0} => {∀i(1≤i<1 =>i%x!=0 ∨ i%y!=0)}`

true

Ex.2

`{x,y > 0} Z := 1 {Inv}`

What could be a proper `Inv`?

- what does each iteration do? Nothing really
- what keeps the loop going on? $\forall i(1 \le i < z \Rightarrow i\%x!=0 \lor i\%y!=0)$

Hence, `Inv` = $\{\forall i(1 \le i < z \Rightarrow i\%x!=0 \lor i\%y!=0)\}$.

$\{x,y > 0\}\ Z := 1\ \{\forall i(1 \le i < z \Rightarrow i\%x!=0 \lor i\%y!=0)\}$

$\{x,y > 0\} \Rightarrow \{\forall i(\underline{1 \le i < 1 \Rightarrow i\%x!=0 \lor i\%y!=0})\}$

If precondition holds, `Inv` is always true before the loop.

Ex.2

- Least common multiple of two positive integers.

```
{x,y > 0} = pre
Z := 1
While (z % x != 0 ∨ z % y != 0 )
      z ++;
{z%x=0 ∧ z%y=0 ∧ ∀i(1≤i<z =>i%x!=0 ∨ i%y!=0)} = post
```

According to while rule:

```
1. {x,y > 0} Z := 1 {Inv}
2. {Inv} while …{Inv ∧ !(z % x != 0 ∨ z % y != 0)}
3. {Inv ∧ !(z % x != 0 ∨ z % y != 0)} => {post}
```

Ex.2

```
{Inv} while …{Inv ∧ !(z % x != 0 ∨ z % y != 0)}
```

Ex.2

`{Inv} while …{Inv ∧ !(z % x != 0 ∨ z % y != 0)}`

According to while rule:

`{Inv ∧ (z % x != 0 ∨ z % y != 0)} z ++;{Inv}`

Ex.2

`{Inv} while …{Inv ∧ !(z % x != 0 ∨ z % y != 0)}`

According to while rule:

`{Inv ∧ (z % x != 0 ∨ z % y != 0)} z ++;{Inv}`

And we know that `Inv = {∀i(1≤i<z =>i%x!=0 ∨ i%y!=0)}`

Ex.2

`{Inv} while …{Inv ∧ !(z % x != 0 ∨ z % y != 0)}`

According to while rule:

`{Inv ∧ (z % x != 0 ∨ z % y != 0)} z ++;{Inv}`

And we know that `Inv = {∀i(1≤i<z =>i%x!=0 ∨ i%y!=0)}`

`{∀i(1≤i≤z =>i%x!=0 ∨ i%y!=0)} z ++;{Inv}`

Ex.2

`{Inv} while …{Inv ∧ !(z % x != 0 ∨ z % y != 0)}`

According to while rule:

`{Inv ∧ (z % x != 0 ∨ z % y != 0)} z ++;{Inv}`

And we know that `Inv = {∀i(1≤i<z =>i%x!=0 ∨ i%y!=0)}`

`{∀i(1≤i≤z =>i%x!=0 ∨ i%y!=0)} z ++;{Inv}`

Ex.2

`{Inv} while …{Inv ∧ !(z % x != 0 ∨ z % y != 0)}`

According to while rule:

`{Inv ∧ (z % x != 0 ∨ z % y != 0)} z ++;{Inv}`

And we know that `Inv = {∀i(1≤i<z =>i%x!=0 ∨ i%y!=0)}`

`{∀i(1≤i≤z =>i%x!=0 ∨ i%y!=0)} z ++;{Inv}`

`{∀i(1≤i≤z =>i%x!=0 ∨ i%y!=0)} => {∀i(1≤i<z+1 =>i%x!=0 ∨ i%y!=0)}`

Ex.2

`{Inv} while …{Inv ∧ !(z % x != 0 ∨ z % y != 0)}`

According to while rule:

`{Inv ∧ (z % x != 0 ∨ z % y != 0)} z ++;{Inv}`

And we know that `Inv = {∀i(1≤i<z =>i%x!=0 ∨ i%y!=0)}`

`{∀i(1≤i≤z =>i%x!=0 ∨ i%y!=0)} z ++;{Inv}`

`{∀i(1≤i≤z =>i%x!=0 ∨ i%y!=0)} => {∀i(1≤i<z+1 =>i%x!=0 ∨ i%y!=0)}`

`{∀i(1≤i≤z =>i%x!=0 ∨ i%y!=0)} => {∀i(1≤i≤z =>i%x!=0 ∨ i%y!=0)}`

equal sides

Ex.2

- Least common multiple of two positive integers.

```
{x,y > 0} = pre
Z := 1
While (z % x != 0 ∨ z % y != 0 )
      z ++;
{z%x=0 ∧ z%y=0 ∧ ∀i(1≤i<z =>i%x!=0 ∨ i%y!=0)} = post
```

According to while rule:

```
1. {x,y > 0} Z := 1 {Inv}
2. {Inv} while …{Inv ∧ !(z % x != 0 ∨ z % y != 0)}
3. {Inv ∧ !(z % x != 0 ∨ z % y != 0)} => {post}
```

Ex.2

```
{Inv ∧ !(z % x != 0 ∨ z % y != 0)} => {post}
```

Ex.2

```
{Inv ∧ !(z % x != 0 ∨ z % y != 0)} => {post}
{Inv ∧ (z % x = 0 ∧ z % y = 0)} => {post}
```

Ex.2

$\{Inv \wedge !(z \% x != 0 \vee z \% y != 0)\} \Rightarrow \{post\}$

$\{Inv \wedge (z \% x = 0 \wedge z \% y = 0)\} \Rightarrow \{post\}$

$\{\forall i(1 \le i < z \Rightarrow i\%x!=0 \vee i\%y!=0) \wedge (z \% x = 0 \wedge z \% y = 0)\} \Rightarrow \{post\}$

Ex.2

```
{Inv ∧ !(z % x != 0 ∨ z % y != 0)} => {post}
{Inv ∧ (z % x = 0 ∧ z % y = 0)} => {post}
{∀i(1≤i<z =>i%x!=0 ∨ i%y!=0) ∧ (z % x = 0 ∧ z % y = 0)} => {post}
```

The left-hand side formula is basically the post-condition.