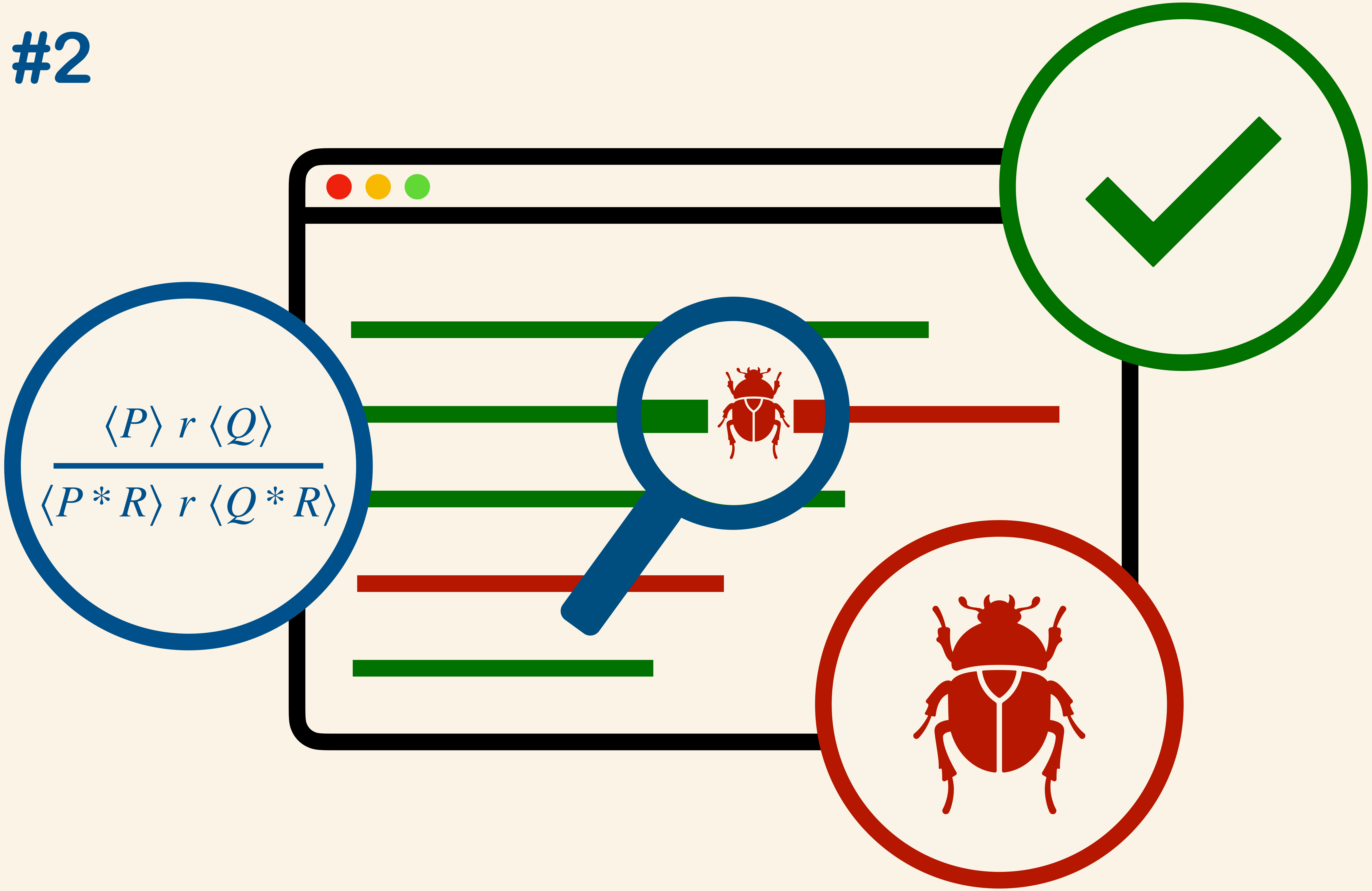
SCAN ME

Program Analysis

Lecture #2

Roberto Bruni

Università di Pisa



$$\frac{\langle P \rangle\ r\ \langle Q \rangle}{\langle P * R \rangle\ r\ \langle Q * R \rangle}$$

PhD Course
June 30 - July 4, 2025

GSSI
GRAN SASSO SCIENCE INSTITUTE
SCHOOL OF ADVANCED STUDIES
Scuola Universitaria Superiore

Consequence rule

$$\frac{P \Rightarrow P' \quad \{P'\}\ r\ \{Q'\} \quad Q' \Rightarrow Q}{\{P\}\ r\ \{Q\}}$$

strengthening

weakening

forward / backward

$$\{x - y < x \land x + y \geq 0\}$$

$$n := x - y;$$

$$\{n < x \land x + y \geq 0\}$$

Consequence rule

$$\frac{P \Rightarrow P' \quad \{P'\}\; r\; \{Q'\} \quad Q' \Rightarrow Q}{\{P\}\; r\; \{Q\}}$$

strengthening

weakening

forward / backward

$$\{-y < 0 \land x \geq 0 \land y \geq 0\} \Rightarrow$$

$$\{x - y < x \land x + y \geq 0\}$$

$$n := x - y;$$

$$\{n < x \land x + y \geq 0\}$$

Consequence rule

$$\frac{P \Rightarrow P' \quad \{P'\}\; r\; \{Q'\} \quad Q' \Rightarrow Q}{\{P\}\; r\; \{Q\}}$$

strengthening

weakening

forward / backward

$$\{x \geq 0 \land y > 0\} \Rightarrow$$

$$\{-y < 0 \land x \geq 0 \land y \geq 0\} \Rightarrow$$

$$\{x - y < x \land x + y \geq 0\}$$

$$n := x - y;$$

$$\{n < x \land x + y \geq 0\}$$

Hoare’s proof

```
  $\{\text{true}\}$
$r := x$

$q := 0;$

while $y \leq r$ do

  $r := r - y;$

  $q := q + 1$
```

Hoare's proof

```
  $\{\text{true}\} \equiv \{x = x\}$
$r := x$
  $\{x = r\}$
$q := 0;$

while $y \leq r$ do

  $r := r - y;$

  $q := q + 1$
```

$$\frac{}{\{Q[a/x]\}\ x := a\ \{Q\}}$$

Hoare's proof

```
  ${\text{true}} \equiv \{x = x\}$
$r := x$
  $\{x = r\} \equiv \{x = r + 0y\}$
$q := 0;$
  $\{x = r + qy\}$
while $y \leq r$ do

  $r := r - y;$

  $q := q + 1$
```

$$\frac{}{\{Q[a/x]\}\ x := a\ \{Q\}}$$

Hoare's proof

```
  ${\{\mathsf{true}\} \equiv \{x = x\}}$
$r := x$
  $\{x = r\} \equiv \{x = r + 0y\}$
$q := 0;$
  $\{x = r + qy\}$
while $y \leq r$ do

  $r := r - y;$

  $q := q + 1$
```

loop invariant?

$$\frac{\{P \wedge b\}\ c\ \{P\}}{\{P\}\,\text{while}\ b\ \text{do}\ c\ \{P \wedge \neg b\}}$$

Hoare's proof

$\{\text{true}\} \equiv \{x = x\}$

$r := x$

$\{x = r\} \equiv \{x = r + 0y\}$

$q := 0;$

$\{x = r + qy\}$

loop invariant?

while $y \leq r$ do

$\{x = r + qy \wedge y \leq r\}$

$r := r - y;$

$q := q + 1$

$\{x = r + qy\}$

$\{x = r + qy \wedge y > r\}$

$$\frac{\{P \wedge b\}\ c\ \{P\}}{\{P\}\ \text{while}\ b\ \text{do}\ c\ \{P \wedge \neg b\}}$$

Hoare's proof

$\{\text{true}\} \equiv \{x = x\}$

$r := x$

$\{x = r\} \equiv \{x = r + 0y\}$

$q := 0;$

loop invariant?

$\{x = r + qy\}$

while $y \leq r$ do

consequence rule

$\{x = r + qy \wedge y \leq r\} \;\Rightarrow\; \{x = r + qy\}$

$r := r - y;$

$q := q + 1$

$\{x = r + qy\}$

$\{x = r + qy \wedge y > r\}$

$$\frac{\{P \wedge b\}\; c\; \{P\}}{\{P\}\,\text{while } b \text{ do } c\; \{P \wedge \neg b\}}$$

Hoare's proof

$\{\text{true}\} \equiv \{x = x\}$

$r := x$

$\{x = r\} \equiv \{x = r + 0y\}$

$q := 0;$

loop invariant?

$\{x = r + qy\}$

while $y \leq r$ do

consequence rule

$\{x = r + qy \land y \leq r\} \;\Rightarrow\; \{x = r + qy\} \equiv \{x = (r - y) + (q + 1)y\}$

$r := r - y;$

$\{x = r + (q + 1)y\}$

$q := q + 1$

$\{x = r + qy\}$

$\{x = r + qy \land y > r\}$

$$\frac{\{P \land b\}\; c\; \{P\}}{\{P\}\,\text{while } b \text{ do } c\;\{P \land \neg b\}}$$

$$\frac{}{\{Q[a/x]\}\; x := a\; \{Q\}}$$

Wait a moment…

```
  ${true} \equiv \{x = x\}$
$r := x$
  $\{x = r\} \equiv \{x = r + 0y\}$
$q := 0;$
  $\{x = r + qy\}$
while $y \leq r$ do
    $\{x = r + qy \land y \leq r\} \Rightarrow \{x = (r - y) + (q + 1)y\}$
  $r := r - y;$
    $\{x = r + (q + 1)y\}$
  $q := q + 1$
    $\{x = r + qy\}$
  $\{x = r + qy \land y > r\}$
```

$$[\![c]\!][x \mapsto 5, y \mapsto -2] = \ldots = \emptyset$$

Wait a moment…

```
  ${true} ≡ \{x = x\}$
$r := x$
  $\{x = r\} \equiv \{x = r + 0y\}$
$q := 0;$
  $\{x = r + qy\}$
while $z = 0$ do
    $\{x = r + qy \wedge z = 0\} \Rightarrow \{x = (r - y) + (q + 1)y\}$
  $r := r - y;$
    $\{x = r + (q + 1)y\}$
  $q := q + 1$
    $\{x = r + qy\}$
  $\{x = r + qy \wedge z \neq 0\}$
```

$[\![c]\!][x \mapsto 5, y \mapsto 2, z \mapsto 0] = \ldots = \emptyset$

No guarantee of termination

$[\![c]\!][x \mapsto 5] = \ldots = \emptyset$

$\{x \geq 0\}$

while $x > 0$ do

$\{x \geq 0 \land x > 0\} \equiv \{x + 1 \geq 0\}$

$x := x + 1;$

$\{x \geq 0\}$

$\{x \geq 0 \land x \leq 0\} \equiv \{x = 0\}$

False positive

$\{x = 1\}$ while $x > 0$ do $x := x + 1$ $\{x = 0\}$

not a possible output!

complete the proof below

$$\{x = 1\} \Rightarrow \{\ ?\ \}$$

while $x > 0$ do

$$\{\ ?\ \wedge x > 0\}$$

$$x := x + 1;$$

$$\{\ ?\ \}$$

$$\{\ ?\ \wedge x \leq 0\} \Rightarrow \{x = 0\}$$

Partial vs total correctness

partial

when the precondition is met,
executing the command
establishes the postcondition

$\{P\}\ c\ \{Q\}$

total

when the precondition is met,
executing the command **terminates**
and establishes the postcondition

total correctness = partial correctness + termination

Total correctness: the idea

choose a measure
called "variant"
(e.g., an arithmetic expression)

$\{P\}$ t
while b do
$\{P \land b\}$ $t = z$
c
$\{P\}$ $t < z$
$\{P \land \neg b\}$

prove that each execution
of the body c
decreases the value of t

here z is a fresh variable
that keeps the value of t
before the execution of c

prove that
whenever the loop invariant holds
the value of t is bounded below

$t \geq 0$

Rule for total correctness

proof obligations for termination

variant = termination function

fresh variable

precedence relation

bounded decrease

$$\frac{\{P \wedge b\}\ c\ \{P\} \qquad \{P \wedge b \wedge t = z\}\ c\ \{t < z\} \qquad P \Rightarrow t \geq 0}{\{P\}\ \text{while}\ b\ \text{do}\ c\ \{P \wedge \neg b\}}$$

Total correctness proof

$\{x \geq 0\}$ take $t \triangleq x$

while $x > 0$ do

$\{x \geq 0 \land x > 0\} \equiv \{x - 1 \geq 0\}$

$x := x - 1;$

$\{x \geq 0\}$

$\{x \geq 0 \land x \leq 0\} \equiv \{x = 0\}$

proof obligations

$P \Rightarrow t \geq 0$

$x \geq 0 \Rightarrow x \geq 0$

$\{P \land b \land t = z\}\ c\ \{t < z\}$

$\{x \geq 0 \land x > 0 \land x = z\} \Rightarrow$

$\{x = z\} \Rightarrow$

$\{x < z + 1\} \equiv$

$\{x - 1 < z\} \Rightarrow$

$x := x - 1$

$\{x < z\}$

Total correctness proof

```
  $\{x \geq 0 \land y > 0\} \equiv \{x \geq 0 \land y > 0 \land x = x + 0y\}$
$r := x$
  $\{x \geq 0 \land y > 0 \land x = r + 0y\} \equiv \{r \geq 0 \land y > 0 \land x = r + 0y\}$
$q := 0;$
  $\{r \geq 0 \land y > 0 \land x = r + qy\}$ take $t \triangleq r$
while $y \leq r$ do
    $\{r \geq y > 0 \land x = r + qy\} \Rightarrow \{r - y \geq 0 \land y > 0 \land x = r - y + (q + 1)y\}$
  $r := r - y;$
    $\{r \geq 0 \land y > 0 \land x = r + (q + 1)y\}$
  $q := q + 1$
    $\{r \geq 0 \land y > 0 \land x = r + qy\}$
  $\{y > r \geq 0 \land x = r + qy\}$
```

Proof obligations

$$P \Rightarrow t \geq 0$$

$$(r \geq 0 \wedge y > 0 \wedge x = r + qy) \ \Rightarrow r \geq 0$$

take $t \triangleq r$

$$\{P \wedge b \wedge t = z\}\ c\ \{t < z\}$$

$\{r \geq y > 0 \wedge \cdots \wedge r = z\} \Rightarrow \{r \geq 0 \wedge y > 0 \wedge \cdots \wedge r - y < z\}$

$r := r - y;$

$\{r \geq 0 \wedge y > 0 \wedge \cdots \wedge r < z\}$

$q := q + 1$

$\{r \geq 0 \wedge y > 0 \wedge \cdots \wedge r < z\} \Rightarrow \{r < z\}$

If rule

$$\frac{\{P \wedge b\}\ c_1\ \{Q\} \qquad \{P \wedge \neg b\}\ c_2\ \{Q\}}{\{P\}\ \text{if}\ b\ \text{then}\ c_1\ \text{else}\ c_2\ \{Q\}}$$

$\{\text{true}\}$

if $x \geq 0$ then

$\{x \geq 0\}$

skip

$\{x \geq 0\}$

else

$\{\neg(x \geq 0)\} \equiv \{-x > 0\}$

$x := -x$

$\{x > 0\} \Rightarrow \{x \geq 0\}$

$\{x \geq 0\}$

Finding invariants is difficult!

```
$\{\mathsf{true}\}$
$k := 1;$
$r := x;$
while $k > 0$ do
    if $r > 100$ then
        $r := r - 10;$
        $k := k - 1$
    else
        $r := r + 11;$
        $k := k + 1$
$\{r = f(x)\}$
```

what is f?

McCarthy's 91 function

```
{true}
$k := 1$;
$r := x$;
while $k > 0$ do
    if $r > 100$ then
        $r := r - 10$;
        $k := k - 1$
    else
        $r := r + 11$;
        $k := k + 1$
{$r = f(x)$}
```

$$f(x) \triangleq \begin{cases} f(f(x+11)) & x \leq 100 \\ x - 10 & \text{otherwise} \end{cases}$$

$$f(x) \triangleq \begin{cases} 91 & x \leq 100 \\ x - 10 & \text{otherwise} \end{cases}$$

Invariant for McCarthy’s 91 function?

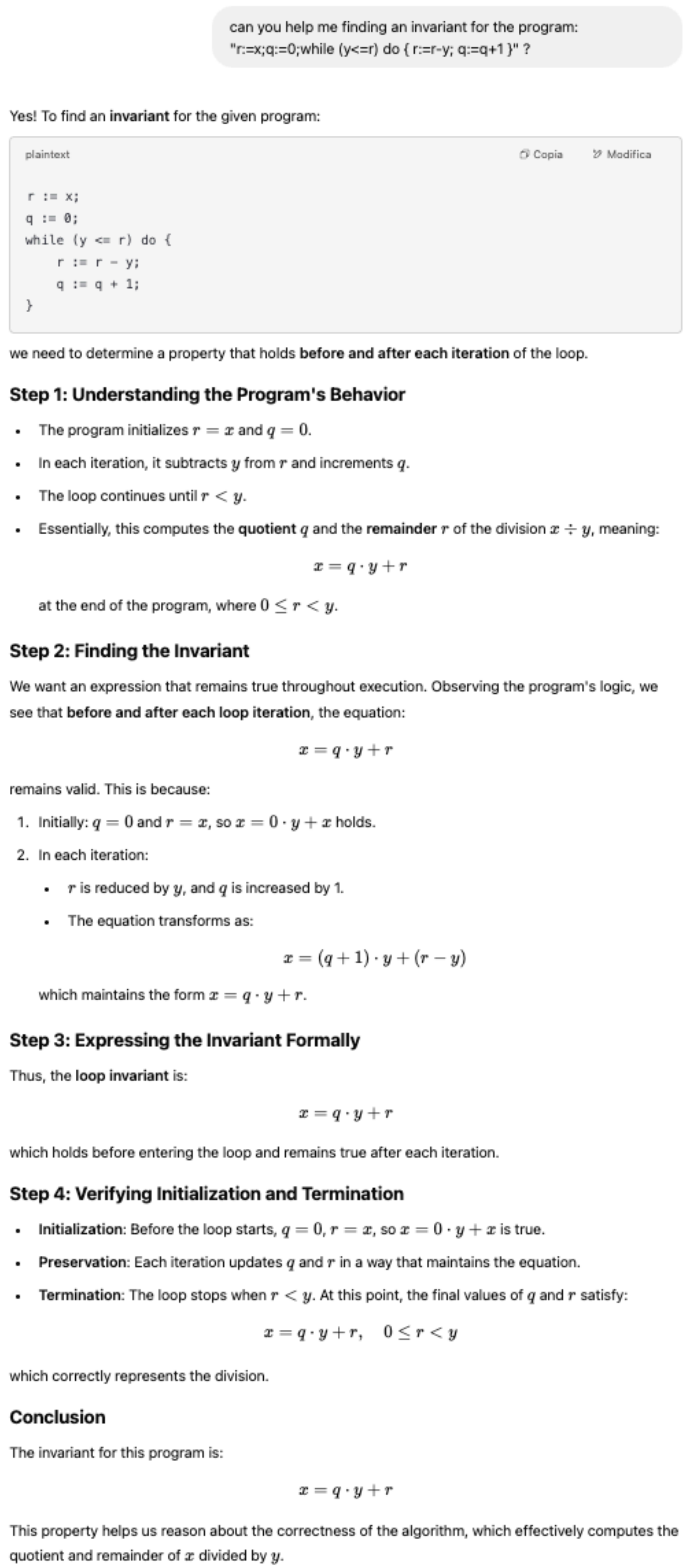
```
{true}
$k := 1$;
$r := x$;
{ ? }
while $k > 0$ do
    if $r > 100$ then
        $r := r - 10$;
        $k := k - 1$
    else
        $r := r + 11$;
        $k := k + 1$
{ ? } $\Rightarrow \{r = f(x)\}$
```

which invariant?

$$f(x) \triangleq \begin{cases} f(f(x+11)) & x \leq 100 \\ x - 10 & \text{otherwise} \end{cases}$$

$$f(x) \triangleq \begin{cases} 91 & x \leq 100 \\ x - 10 & \text{otherwise} \end{cases}$$

Ask an expert?



can you help me finding an invariant for the program: "r:=x;q:=0;while (y<=r) do { r:=r-y; q:=q+1 }" ?

Yes! To find an **invariant** for the given program:

```
r := x;
q := 0;
while (y <= r) do {
    r := r - y;
    q := q + 1;
}
```

we need to determine a property that holds **before and after each iteration** of the loop.

Step 1: Understanding the Program's Behavior

- The program initializes $r = x$ and $q = 0$.
- In each iteration, it subtracts y from r and increments q.
- The loop continues until $r < y$.
- Essentially, this computes the **quotient** q and the **remainder** r of the division $x \div y$, meaning:

$$x = q \cdot y + r$$

at the end of the program, where $0 \leq r < y$.

Step 2: Finding the Invariant

We want an expression that remains true throughout execution. Observing the program's logic, we see that **before and after each loop iteration**, the equation:

$$x = q \cdot y + r$$

remains valid. This is because:

1. Initially: $q = 0$ and $r = x$, so $x = 0 \cdot y + x$ holds.
2. In each iteration:
   - r is reduced by y, and q is increased by 1.
   - The equation transforms as:

$$x = (q+1) \cdot y + (r - y)$$

which maintains the form $x = q \cdot y + r$.

Step 3: Expressing the Invariant Formally

Thus, the **loop invariant** is:

$$x = q \cdot y + r$$

which holds before entering the loop and remains true after each iteration.

Step 4: Verifying Initialization and Termination

- **Initialization**: Before the loop starts, $q = 0, r = x$, so $x = 0 \cdot y + x$ is true.
- **Preservation**: Each iteration updates q and r in a way that maintains the equation.
- **Termination**: The loop stops when $r < y$. At this point, the final values of q and r satisfy:

$$x = q \cdot y + r, \quad 0 \leq r < y$$

which correctly represents the division.

Conclusion

The invariant for this program is:

$$x = q \cdot y + r$$

This property helps us reason about the correctness of the algorithm, which effectively computes the quotient and remainder of x divided by y.

can you help me finding an invariant for the program: "r:=x;q:=0;while (y<=r) do { r:=r-y; q:=q+1 }" ?

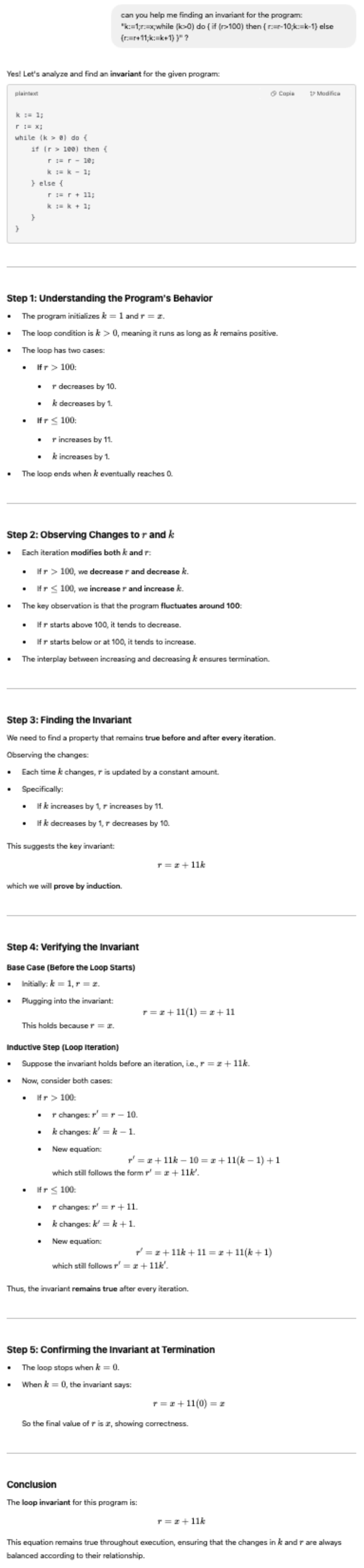
The invariant for this program is:

$$x = q \cdot y + r$$

can you help me finding an invariant for the program: "k:=1;r:=x;while (k>0) do { if (r>100) then { r:=r-10;k:=k-1} else {r:=r+11;k:=k+1} }" ?

The **loop invariant** for this program is:

$$r = x + 11k$$



can you help me finding an invariant for the program: "k:=1;r:=x;while (k>0) do { if (r>100) then { r:=r-10;k:=k-1} else {r:=r+11;k:=k+1} }" ?

Yes! Let's analyze and find an **invariant** for the given program:

```
k := 1;
r := x;
while (k > 0) do {
    if (r > 100) then {
        r := r - 10;
        k := k - 1;
    } else {
        r := r + 11;
        k := k + 1;
    }
}
```

Step 1: Understanding the Program's Behavior

- The program initializes $k = 1$ and $r = x$.
- The loop condition is $k > 0$, meaning it runs as long as k remains positive.
- The loop has two cases:
  - If $r > 100$:
    - r decreases by 10.
    - k decreases by 1.
  - If $r \leq 100$:
    - r increases by 11.
    - k increases by 1.
- The loop ends when k eventually reaches 0.

Step 2: Observing Changes to r and k

- Each iteration **modifies both k and r**:
  - If $r > 100$, we **decrease r and decrease k**.
  - If $r \leq 100$, we **increase r and increase k**.
- The key observation is that the program **fluctuates around 100**:
  - If r starts above 100, it tends to decrease.
  - If r starts below or at 100, it tends to increase.
- The interplay between increasing and decreasing k ensures termination.

Step 3: Finding the Invariant

We need to find a property that remains **true before and after every iteration.**

Observing the changes:

- Each time k changes, r is updated by a constant amount.
- Specifically:
  - If k increases by 1, r increases by 11.
  - If k decreases by 1, r decreases by 10.

This suggests the key invariant:

$$r = x + 11k$$

which we will **prove by induction.**

Step 4: Verifying the Invariant

Base Case (Before the Loop Starts)

- Initially: $k = 1, r = x$.
- Plugging into the invariant:

$$r = x + 11(1) = x + 11$$

This holds because $r = x$.

Inductive Step (Loop Iteration)

- Suppose the invariant holds before an iteration, i.e., $r = x + 11k$.
- Now, consider both cases:
  - If $r > 100$:
    - r changes: $r' = r - 10$.
    - k changes: $k' = k - 1$.
    - New equation:

$$r' = x + 11k - 10 = x + 11(k-1) + 1$$

which still follows the form $r' = x + 11k'$.

  - If $r \leq 100$:
    - r changes: $r' = r + 11$.
    - k changes: $k' = k + 1$.
    - New equation:

$$r' = x + 11k + 11 = x + 11(k+1)$$

which still follows $r' = x + 11k'$.

Thus, the invariant **remains true** after every iteration.

Step 5: Confirming the Invariant at Termination

- The loop stops when $k = 0$.
- When $k = 0$, the invariant says:

$$r = x + 11(0) = x$$

So the final value of r is x, showing correctness.

Conclusion

The **loop invariant** for this program is:

$$r = x + 11k$$

This equation remains true throughout execution, ensuring that the changes in k and r are always balanced according to their relationship.

Invariant for McCarthy’s 91 function?

```
$\{\text{true}\}$
$k := 1;$
$r := x;$
$\{\ r = x + 11k\ \}$
while $k > 0$ do
    if $r > 100$ then
        $r := r - 10;$
        $k := k - 1$
    else
        $r := r + 11;$
        $k := k + 1$
$\{\ ?\ \} \Rightarrow \{r = f(x)\}$
```

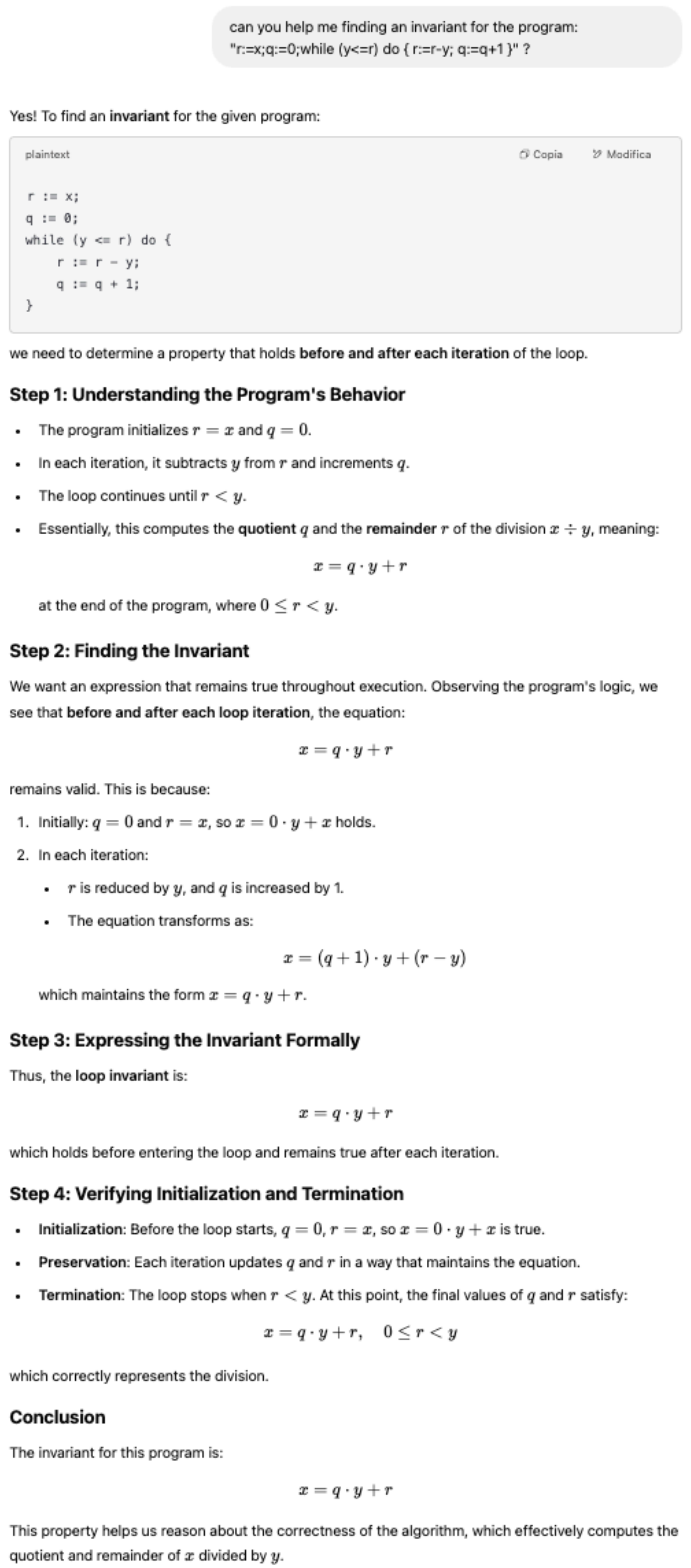
don't think so!

when $(k = 1, r = x)$ then $r = x + 11k$ does not hold!

$$f(x) \triangleq \begin{cases} f(f(x+11)) & x \leq 100 \\ x - 10 & \text{otherwise} \end{cases}$$

$$f(x) \triangleq \begin{cases} 91 & x \leq 100 \\ x - 10 & \text{otherwise} \end{cases}$$

Ask an expert?



can you help me finding an invariant for the program: "r:=x;q:=0;while (y<=r) do { r:=r-y; q:=q+1 }" ?

Yes! To find an **invariant** for the given program:

```
r := x;
q := 0;
while (y <= r) do {
    r := r - y;
    q := q + 1;
}
```

we need to determine a property that holds **before and after each iteration** of the loop.

Step 1: Understanding the Program's Behavior

- The program initializes $r = x$ and $q = 0$.
- In each iteration, it subtracts y from r and increments q.
- The loop continues until $r < y$.
- Essentially, this computes the **quotient** q and the **remainder** r of the division $x \div y$, meaning:

  $$x = q \cdot y + r$$

  at the end of the program, where $0 \leq r < y$.

Step 2: Finding the Invariant

We want an expression that remains true throughout execution. Observing the program's logic, we see that **before and after each loop iteration**, the equation:

$$x = q \cdot y + r$$

remains valid. This is because:

1. Initially: $q = 0$ and $r = x$, so $x = 0 \cdot y + x$ holds.
2. In each iteration:
   - r is reduced by y, and q is increased by 1.
   - The equation transforms as:

     $$x = (q + 1) \cdot y + (r - y)$$

   which maintains the form $x = q \cdot y + r$.

Step 3: Expressing the Invariant Formally

Thus, the **loop invariant** is:

$$x = q \cdot y + r$$

which holds before entering the loop and remains true after each iteration.

Step 4: Verifying Initialization and Termination

- **Initialization**: Before the loop starts, $q = 0, r = x$, so $x = 0 \cdot y + x$ is true.
- **Preservation**: Each iteration updates q and r in a way that maintains the equation.
- **Termination**: The loop stops when $r < y$. At this point, the final values of q and r satisfy:

  $$x = q \cdot y + r, \quad 0 \leq r < y$$

which correctly represents the division.

Conclusion

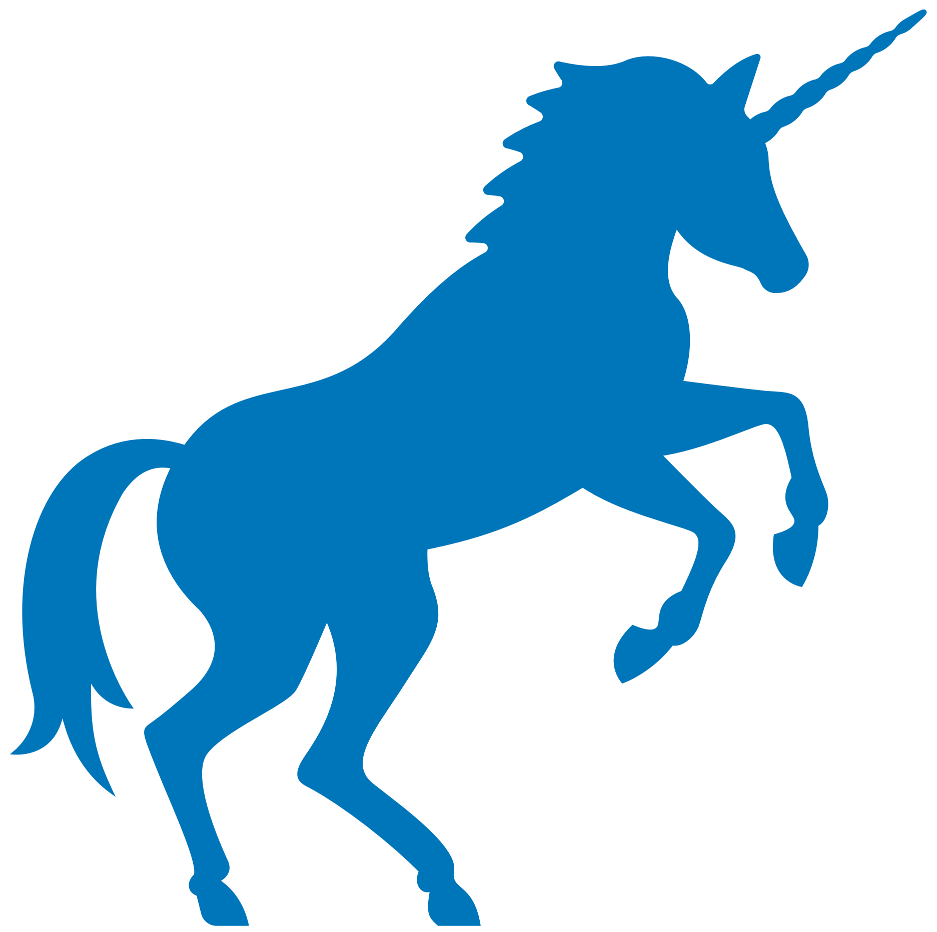
The invariant for this program is:

$$x = q \cdot y + r$$

This property helps us reason about the correctness of the algorithm, which effectively computes the quotient and remainder of x divided by y.

can you help me finding an invariant for the program: "r:=x;q:=0;while (y<=r) do { r:=r-y; q:=q+1 }" ?

The invariant for this program is:

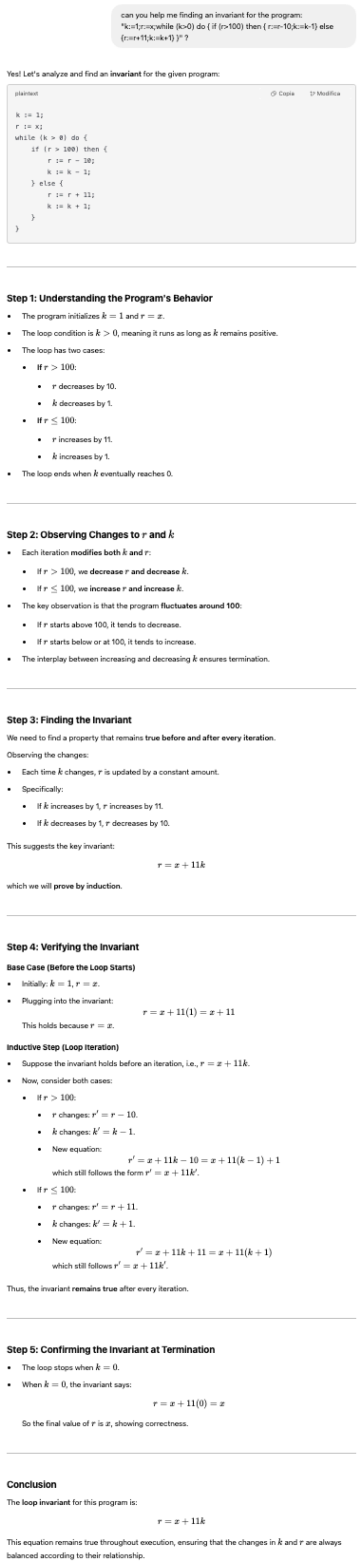


$$x = q \cdot y + r$$

can you help me finding an invariant for the program: "k:=1;r:=x;while (k>0) do { if (r>100) then { r:=r-10;k:=k-1} else {r:=r+11;k:=k+1} }" ?

The **loop invariant** for this program is:

$$r = x + 11k$$



can you help me finding an invariant for the program: "k:=1;r:=x;while (k>0) do { if (r>100) then { r:=r-10;k:=k-1} else {r:=r+11;k:=k+1} }" ?

Yes! Let's analyze and find an **invariant** for the given program:

```
k := 1;
r := x;
while (k > 0) do {
    if (r > 100) then {
        r := r - 10;
        k := k - 1;
    } else {
        r := r + 11;
        k := k + 1;
    }
}
```

Step 1: Understanding the Program's Behavior

- The program initializes $k = 1$ and $r = x$.
- The loop condition is $k > 0$, meaning it runs as long as k remains positive.
- The loop has two cases:
  - If $r > 100$:
    - r decreases by 10.
    - k decreases by 1.
  - If $r \leq 100$:
    - r increases by 11.
    - k increases by 1.
- The loop ends when k eventually reaches 0.

Step 2: Observing Changes to r and k

- Each iteration **modifies both k and r**:
  - If $r > 100$, we **decrease r and decrease k**.
  - If $r \leq 100$, we **increase r and increase k**.
- The key observation is that the program **fluctuates around 100**:
  - If r starts above 100, it tends to decrease.
  - If r starts below or at 100, it tends to increase.
- The interplay between increasing and decreasing k ensures termination.

Step 3: Finding the Invariant

We need to find a property that remains **true before and after every iteration**.

Observing the changes:

- Each time k changes, r is updated by a constant amount.
- Specifically:
  - If k increases by 1, r increases by 11.
  - If k decreases by 1, r decreases by 10.

This suggests the key invariant:

$$r = x + 11k$$

which we will **prove by induction**.

Step 4: Verifying the Invariant

Base Case (Before the Loop Starts)

- Initially: $k = 1, r = x$.
- Plugging into the invariant:

  $$r = x + 11(1) = x + 11$$

  This holds because $r = x$.

Inductive Step (Loop Iteration)

- Suppose the invariant holds before an iteration, i.e., $r = x + 11k$.
- Now, consider both cases:
  - If $r > 100$:
    - r changes: $r' = r - 10$.
    - k changes: $k' = k - 1$.
    - New equation:

      $$r' = x + 11k - 10 = x + 11(k - 1) + 1$$

      which still follows the form $r' = x + 11k'$.
  - If $r \leq 100$:
    - r changes: $r' = r + 11$.
    - k changes: $k' = k + 1$.
    - New equation:

      $$r' = x + 11k + 11 = x + 11(k + 1)$$

      which still follows $r' = x + 11k'$.

Thus, the invariant **remains true** after every iteration.

Step 5: Confirming the Invariant at Termination

- The loop stops when $k = 0$.
- When $k = 0$, the invariant says:

  $$r = x + 11(0) = x$$

  So the final value of r is x, showing correctness.

Conclusion

The **loop invariant** for this program is:

$$r = x + 11k$$

This equation remains true throughout execution, ensuring that the changes in k and r are always balanced according to their relationship.

Invariant for McCarthy's 91 function

```
 $\{\text{true}\}$
$k := 1;$
$r := x;$
 $\{k \geq 0 \wedge f^k(r) = f(x)\}$
while $k > 0$ do
    if $r > 100$ then
        $r := r - 10;$
        $k := k - 1$
    else
        $r := r + 11;$
        $k := k + 1$
$\{k = 0 \wedge f^k(r) = f(x)\} \Rightarrow \{r = f(x)\}$
```

when $(k = 1, r = x)$ then both $k \geq 0$ and $f^k(r) = f^1(x) = f(x)$ hold

$$f(x) \triangleq \begin{cases} f(f(x+11)) & x \leq 100 \\ x - 10 & \text{otherwise} \end{cases}$$

$$f(x) \triangleq \begin{cases} 91 & x \leq 100 \\ x - 10 & \text{otherwise} \end{cases}$$

Variant for McCarthy's 91 function?

```
{true}
$k := 1;$
$r := x;$
{$k \geq 0 \wedge f^k(r) = f(x)$}    $t \triangleq ?$
while $k > 0$ do
    if $r > 100$ then
        $r := r - 10;$
        $k := k - 1$
    else
        $r := r + 11;$
        $k := k + 1$
{$k = 0 \wedge f^k(r) = f(x)$} $\Rightarrow$ {$r = f(x)$}
```

which variant (for termination)?

$$f(x) \triangleq \begin{cases} f(f(x+11)) & x \leq 100 \\ x - 10 & \text{otherwise} \end{cases}$$

$$f(x) \triangleq \begin{cases} 91 & x \leq 100 \\ x - 10 & \text{otherwise} \end{cases}$$

Finding invariants (McCarthy91)

```
  {true}
$k := 1;$
$r := x;$
  $\{k \geq 0 \wedge f^k(r) = f(x)\}$
while $k > 0$ do
    if $r > 100$ then
        $r := r - 10;$
        $k := k - 1$
    else
        $r := r + 11;$
        $k := k + 1$
  $\{k = 0 \wedge f^k(r) = f(x)\} \Rightarrow \{r = f(x)\}$
```

$t = (|101 - r + 10k|, k)$

lexicographic order

$$f(x) \triangleq \begin{cases} f(f(x+11)) & x \leq 100 \\ x - 10 & \text{otherwise} \end{cases}$$

$$f(x) \triangleq \begin{cases} 91 & x \leq 100 \\ x - 10 & \text{otherwise} \end{cases}$$

Validity, soundness, completeness

Validity

A HL triple $\{P\}\ c\ \{Q\}$ is **valid** if $[\![c]\!]P \subseteq Q$

Is $\{x > 0\}\ x := 10x\ \{x > 10\}$ valid? ✗

Is $\{x > 0, y > 0\}\ x := yx\ \{x \geq 0\}$ valid? ✓

Is $\{\mathsf{false}\}\ c\ \{Q\}$ valid? ✓

Is $\{P\}\ c\ \{\mathsf{true}\}$ valid? ✓

Correctness

Th. Any derivable HL triple is valid

Proof. By induction on the derivation tree, e.g.

$$\frac{\{P\}\ c_1\ \{R\} \quad \{R\}\ c_2\ \{Q\}}{\{P\}\ c_1; c_2\ \{Q\}}$$

We prove the conclusion is valid assuming the premises are valid

$$[\![c_1; c_2]\!]P\ = [\![c_2]\!]([\![c_1]\!]P) \subseteq [\![c_2]\!]R\ \subseteq Q$$

Incompleteness I

Conjecture Any valid HL triple is derivable

Counterexample:
$\{\mathsf{true}\}\ c\ \{\mathsf{false}\}$ is valid only when c diverges
but halting problem is not r.e.
while the set of derivable HL triples is r.e.

Incompleteness II

Conjecture Any valid HL triple is derivable

Counterexample:

$\{\mathsf{true}\}$ skip $\{Q\}$ is valid when Q is a tautology
but Godel’s Incompleteness Theorem (1939) tells us that there is no *effective* proof system such that its theorems coincide with all valid arithmetic assertions

Relative completeness I

Relative completeness: suppose we can consult an oracle to check if an assertion $P \Rightarrow P'$ is valid or not, then HL is complete

In other words, we separate concerns about programs and reasoning about them from concerns to do with arithmetic and the incompleteness of any proof system for it

Dijkstra’s weakest precondition

Given a command c and a postcondition Q a **weakest liberal precondition** is a predicate P such that for any precondition R

$$\{R\}\ c\ \{Q\} \text{ iff } R \Rightarrow P$$

i.e., P is the least restrictive requirement that guarantees that Q holds after executing c (if it terminates)

Typically, it is denoted by $wlp(c, Q) \triangleq \{\sigma \in \Sigma \mid [\![c]\!]\{\sigma\} \subseteq Q\}$

Adjoints

$$P \Rightarrow wlp(c, Q)$$

iff

strongest postcondition

$$[\![c]\!]P \subseteq Q$$

(Relative) Completeness

for any postcondition Q expressible in the logic and for any command c, the precondition $wlp(c, Q)$ is also expressible in the logic

Th. If the logic language is *expressive enough*, then any valid HL triple can be derived.

Proof. Suppose $\{P\}\ c\ \{Q\}$ is valid (with P and Q expressible). By structural induction on c we can build an assertion R that is equivalent to $wlp(c, Q)$ and such that $\{R\}\ c\ \{Q\}$ is derivable. By applying the consequence rule we derive $\{P\}\ c\ \{Q\}$.

Adding nondeterminism

Regular commands

regular command

atomic command

$$
\begin{aligned}
r ::=\ & e \\
\mid\ & r_1 ; r_2 \\
\mid\ & r_1 + r_2 \\
\mid\ & r^{\star}
\end{aligned}
$$

choice

Kleene star

$$e ::= \text{skip} \mid x := a \mid b? \mid \ldots$$

$$[\![b?]\!] P \triangleq [\![b]\!] P$$

$$[\![r_1 + r_2]\!] P \triangleq [\![r_1]\!] P \cup [\![r_2]\!] P$$

$$[\![r^{\star}]\!] P \triangleq \bigcup_{k=0}^{\infty} [\![r]\!]^k P$$

Encoding while commands

if b then c_1 else c_2 $\triangleq (b?; c_1) + (\neg b?; c_2)$

while b do c $\triangleq (b?; c)^\star; \neg b?$

Minimal set of rules

$$\frac{}{\{P\}\ e\ \{[\![e]\!]P\}}\ \text{\{atom\}}$$

$$\frac{\{P\}\ r_1\ \{R\} \quad \{R\}\ r_2\ \{Q\}}{\{P\}\ r_1; r_2\ \{Q\}}\ \text{\{seq\}}$$

$$\frac{\forall i \in \{1,2\} \quad \{P\}\ r_i\ \{Q\}}{\{P\}\ r_1 + r_2\ \{Q\}}\ \text{\{choice\}}$$

$$\frac{\{P\}\ r\ \{P\}}{\{P\}\ r^\star\ \{P\}}\ \text{\{iter\}}$$

$$\frac{P \Rightarrow P' \quad \{P'\}\ r\ \{Q'\} \quad Q' \Rightarrow Q}{\{P\}\ r\ \{Q\}}\ \text{\{cons\}}$$

Auxiliary rules

$$\frac{\{P_1\}\ r\ \{Q_1\} \quad \{P_2\}\ r\ \{Q_2\}}{\{P_1 \lor P_2\}\ r\ \{Q_1 \lor Q_2\}}\ \{\text{disj}\}$$

$$\frac{\{P_1\}\ r\ \{Q_1\} \quad \{P_2\}\ r\ \{Q_2\}}{\{P_1 \land P_2\}\ r\ \{Q_1 \land Q_2\}}\ \{\text{conj}\}$$

assigned variables in r
are disjoint from
free variables in R

$$\frac{\{P\}\ r\ \{Q\}}{\{P \land R\}\ r\ \{Q \land R\}}\ \{\text{frame}\}$$

$$\frac{P \Rightarrow P' \quad \{P'\}\ r\ \{Q\}}{\{P\}\ r\ \{Q\}}\ \{\text{stren}\}$$

$$\frac{\{P\}\ r\ \{Q'\} \quad Q' \Rightarrow Q}{\{P\}\ r\ \{Q\}}\ \{\text{weak}\}$$

Questions

Question 1

Can we take $P = \neg b$ an invariant?

$$\frac{\{P \land b\}\ c\ \{P\}}{\{P\}\ \text{while}\ b\ \text{do}\ c\ \{P \land \neg b\}}$$

$$\frac{\{\text{false}\}\ c\ \{\neg b\}}{\{\neg b\}\ \text{while}\ b\ \text{do}\ c\ \{\neg b\}}$$ ✓

Question 2

Find a derivation for the HL triple

$$\{\mathsf{true}\}\ \text{if } x \geq y \text{ then } z := x \text{ else } z := y\ \{z = \max(x, y)\}$$

```
  $\{\mathsf{true}\}$
if $x \geq y$ then
      $\{x \geq y\}$
   $z := x$
      $\{z = x \geq y\} \Rightarrow \{z = \max(x, y)\}$
else
     $\{x < y\}$
   $z := y$
     $\{z = y > x\} \Rightarrow \{z = \max(x, y)\}$
  $\{z = \max(x, y)\}$
```

Question 3

Prove that rule {conj} is sound

$$\frac{\{P_1\}\ r\ \{Q_1\} \quad \{P_2\}\ r\ \{Q_2\}}{\{P_1 \wedge P_2\}\ r\ \{Q_1 \wedge Q_2\}}\ \{\text{conj}\}$$

Assume $[\![r]\!]P_1 \subseteq Q_1$ and $[\![r]\!]P_2 \subseteq Q_2$

By monotonicity of $[\![r]\!]$ we have:

$[\![r]\!](P_1 \wedge P_2) \subseteq [\![r]\!]P_1 \subseteq Q_1$ and

$[\![r]\!](P_1 \wedge P_2) \subseteq [\![r]\!]P_2 \subseteq Q_2$

Therefore $[\![r]\!](P_1 \wedge P_2) \subseteq Q_1 \wedge Q_2$

Question 4

Show that the following rule for assignment is not sound

$$\frac{}{\{P\}\ x := a\ \{P[a/x]\}}$$

syntax replacement

Consider the instance $\{x = y\}\ x := 0\ \{y = 0\}$

then $\llbracket x := 0 \rrbracket [x \mapsto 1, y \mapsto 1] = [x \mapsto 0, y \mapsto 1] \nvDash \{y = 0\}$