Program Verification

What is a “correct program”? 

It has to terminate. 

Provably impossible to detect (for all programs) 

Assuming it terminates, its ending state should match some stated objective. 

How do we specify what a program is supposed to do? 

initial assertion: conjunction of propositions about the initial variables used in the program 

final assertion: conjunction of propositions about the final state of the program 

Note: There are many programs that are “correct” for the same criteria. The ends justify the means. 

What kind of programs will we consider? 

We will look at: 

▶ Assignment statements 
▶ Sequences of statements 
▶ Conditional statements (If B then S1 else S2) 
▶ iteration statements (while loops)
Program Verification

What is a “correct program”? It has to terminate.
Program Verification

What is a “correct program”? It has to terminate.
Assuming it terminates, its ending state should match some stated objective.
Program Verification

What is a “correct program”?
It has to terminate. **Provably impossible to detect (for all programs)**
Assuming it terminates, its ending state should match some stated objective.

How do we specify what a program is supposed to do?
Program Verification

What is a “correct program”? It has to terminate. Provably impossible to detect (for all programs) Assuming it terminates, its ending state should match some stated objective.

How do we specify what a program is supposed to do? initial assertion: conjunction of propositions about the initial variables used in the program
Program Verification

What is a “correct program”? It has to terminate. **Provably impossible to detect (for all programs)** Assuming it terminates, its ending state should match some stated objective.

How do we specify what a program is supposed to do?
initial assertion: conjunction of propositions about the initial variables used in the program
final assertion: conjunction of propositions about the final state of the program
What is a “correct program”? It has to terminate. Provably impossible to detect (for all programs) Assuming it terminates, its ending state should match some stated objective.

How do we specify what a program is supposed to do? initial assertion: conjunction of propositions about the initial variables used in the program final assertion: conjunction of propositions about the final state of the program Note: There are many programs that are “correct” for the same criteria. The ends justify the means.
Program Verification

What is a “correct program”? It has to terminate. *Provably impossible to detect (for all programs)* Assuming it terminates, its ending state should match some stated objective.

How do we specify what a program is supposed to do? initial assertion: conjunction of propositions about the initial variables used in the program final assertion: conjunction of propositions about the final state of the program

Note: There are many programs that are “correct” for the same criteria. The ends justify the means.

What kind of programs will we consider?
Program Verification

What is a “correct program”?
It has to terminate. **Provably impossible to detect (for all programs)**
Assuming it terminates, its ending state should match some stated objective.

How do we specify what a program is supposed to do?
initial assertion: conjunction of propositions about the initial variables used in the program
final assertion: conjunction of propositions about the final state of the program
Note: There are many programs that are “correct” for the same criteria. The ends justify the means.

What kind of programs will we consider?
We will look at:

- Assignment statements
- Sequences of statements
- Conditional statements (If B then S1 else S2)
- iteration statements (while loops)
Hoare Triples, Assignments, and Sequences

Hoare Triple: $p \{S\} q$:
If $p$ is true for initial state of code $S$, and $S$ terminates,
then $q$ is true about the final state.
Hoare Triples, Assignments, and Sequences

Hoare Triple: $p\{S\}q$:
If $p$ is true for initial state of code $S$, and $S$ terminates,
then $q$ is true about the final state.
$p$ is the pre-condition
$q$ is the post-condition.
Hoare Triples, Assignments, and Sequences

Hoare Triple: $p\{S\}q$:
If $p$ is true for initial state of code $S$, and $S$ terminates,
then $q$ is true about the final state.
$p$ is the pre-condition
$q$ is the post-condition.

Assignment operator:
$p(e)\{v \leftarrow e\}p(v)$
Hoare Triples, Assignments, and Sequences

Hoare Triple: $p\{S\}q$:
If $p$ is true for initial state of code $S$, and $S$ terminates,
then $q$ is true about the final state.
$p$ is the pre-condition
$q$ is the post-condition.

Assignment operator:
$p(e)\{v \leftarrow e\}p(v)$
$ODD(y)\{x \leftarrow y + 2\}ODD(x)$
Hoare Triples, Assignments, and Sequences

Hoare Triple: \( p \{ S \} q \):
If \( p \) is true for initial state of code \( S \), and \( S \) terminates,
then \( q \) is true about the final state.
\( p \) is the pre-condition
\( q \) is the post-condition.

Assignment operator:
\( p(e) \{ v \leftarrow e \} p(v) \)
\( ODD(y) \{ x \leftarrow y + 2 \} ODD(x) \)
\( ODD(x) \{ x \leftarrow x + 1 \} Even(x) \)
Hoare Triples, Assignments, and Sequences

Hoare Triple: \( p \{ S \} q \):
If \( p \) is true for initial state of code \( S \), and \( S \) terminates,
then \( q \) is true about the final state.
\( p \) is the pre-condition
\( q \) is the post-condition.

Assignment operator:
\( p(e) \{ \nu \leftarrow e \} p(\nu) \)
\( \text{ODD}(y) \{ x \leftarrow y + 2 \} \text{ODD}(x) \)
\( \text{ODD}(x) \{ x \leftarrow x + 1 \} \text{Even}(x) \)

Sequencing of statements:
\( p \{ S_1 \} q \)
\( q \{ S_2 \} r \)
\___________
\( p \{ S_1 ; S_2 \} r \)
Examples

Consider the following simple program:

\[
y \leftarrow 3; \quad z \leftarrow x + y
\]

Claim: If we have the pre-condition \(x = 1\), we have the post-condition \(z = 4\).

\[
(x = 1) \{ y \leftarrow 3 \} \quad (x = 1 \land y = 3) \{ z \leftarrow x + y \} \quad (z = 4)
\]

Consider the following simple program:

\[
x \leftarrow x + 2; \quad y \leftarrow y + 1
\]

Claim: if we have the pre-condition \(x = 2\), \(y\), we have the post-condition \(x = 2\), \(y\).

\[
(x = 2) \{ x \leftarrow x + 2 \} \quad (x = 2(y + 1)) \{ y \leftarrow y + 1 \} \quad (x = 2(y + 1))
\]

This is called an invariant condition (or just an invariant)
Examples

Consider the following simple program:

\[
y \leftarrow 3; \quad z \leftarrow x + y
\]

Claim: If we have the pre-condition \( x = 1 \), we have the post-condition \( z = 4 \).
Examples

Consider the following simple program:
\[ y \leftarrow 3; z \leftarrow x + y \]
Claim: If we have the pre-condition \( x = 1 \), we have the post-condition \( z = 4 \).  
\[(x = 1)(y \leftarrow 3)(x = 1 \land y = 3)\]
Examples

Consider the following simple program:
\[ y \leftarrow 3; \; z \leftarrow x + y \]

Claim: If we have the pre-condition \( x = 1 \), we have the post-condition \( z = 4 \).
\[
(x = 1)\{y \leftarrow 3\}(x = 1 \land y = 3) \\
(x = 1 \land y = 3)\{z \leftarrow x + y\}(z = 4)
\]
Examples

Consider the following simple program:
\[ y \leftarrow 3; z \leftarrow x + y \]

Claim: If we have the pre-condition \( x = 1 \), we have the post-condition \( z = 4 \).
\[ (x = 1) \{ y \leftarrow 3 \} (x = 1 \land y = 3) \]
\[ (x = 1 \land y = 3) \{ z \leftarrow x + y \} (z = 4) \]

This is called an invariant condition (or just an invariant)
Examples

Consider the following simple program:
\[
y \leftarrow 3; z \leftarrow x + y
\]
Claim: If we have the pre-condition \( x = 1 \), we have the post-condition \( z = 4 \).
\[
(x = 1)\{y \leftarrow 3\}(x = 1 \land y = 3)
(x = 1 \land y = 3)\{z \leftarrow x + y\}(z = 4)
\]

\[
(x = 1)\{y \leftarrow 3; z \leftarrow x + y\}(z = 4)
\]

Consider the following simple program:
\[
x \leftarrow x + 2; y \leftarrow y + 1
\]
Examples

Consider the following simple program:
\[ y \leftarrow 3; z \leftarrow x + y \]

Claim: If we have the pre-condition \( x = 1 \), we have the post-condition \( z = 4 \).
\[
(x = 1) \{ y \leftarrow 3 \}(x = 1 \wedge y = 3) \\
(x = 1 \wedge y = 3) \{ z \leftarrow x + y \}(z = 4)
\]

\[
(x = 1) \{ y \leftarrow 3; z \leftarrow x + y \}(z = 4)
\]

Consider the following simple program:
\[ x \leftarrow x + 2; y \leftarrow y + 1 \]

Claim: if we have the pre-condition \( x = 2y \), we have the post-condition \( x = 2y \).
Examples

Consider the following simple program:
\[ y \leftarrow 3; \ z \leftarrow x + y \]
Claim: If we have the pre-condition \( x = 1 \), we have the post-condition \( z = 4 \).
\[
(x = 1) \{ y \leftarrow 3 \} (x = 1 \land y = 3) \\
(x = 1 \land y = 3) \{ z \leftarrow x + y \} (z = 4)
\]

\[
(x = 1) \{ y \leftarrow 3; \ z \leftarrow x + y \} (z = 4)
\]

Consider the following simple program:
\[ x \leftarrow x + 2; \ y \leftarrow y + 1 \]
Claim: if we have the pre-condition \( x = 2y \), we have the post-condition \( x = 2y \).
\[
(x = 2y) \{ x \leftarrow x + 2 \} (x = 2y + 2)
\]
Examples

Consider the following simple program:
\[ y \leftarrow 3; z \leftarrow x + y \]
Claim: If we have the pre-condition \( x = 1 \), we have the post-condition \( z = 4 \).
\[
(x = 1)\{ y \leftarrow 3 \}(x = 1 \land y = 3) \\
(x = 1 \land y = 3)\{ z \leftarrow x + y \}(z = 4)
\]

Consider the following simple program:
\[ x \leftarrow x + 2; y \leftarrow y + 1 \]
Claim: If we have the pre-condition \( x = 2y \), we have the post-condition \( x = 2y \).
\[
(x = 2y)\{ x \leftarrow x + 2 \}(x = 2y + 2) \\
(x = 2(y + 1))\{ y \leftarrow y + 1 \}(x = 2y)
\]
Examples

Consider the following simple program:
\( y \leftarrow 3; z \leftarrow x + y \)
Claim: If we have the pre-condition \( x = 1 \), we have the post-condition \( z = 4 \).
\[(x = 1) \{ y \leftarrow 3 \} \{ x = 1 \land y = 3 \} \]
\[(x = 1 \land y = 3) \{ z \leftarrow x + y \} (z = 4) \]
\[\overline{\text{___________}}\]
\[(x = 1) \{ y \leftarrow 3; z \leftarrow x + y \} (z = 4)\]

Consider the following simple program:
\( x \leftarrow x + 2; y \leftarrow y + 1 \)
Claim: if we have the pre-condition \( x = 2y \), we have the post-condition \( x = 2y \).
\[(x = 2y) \{ x \leftarrow x + 2 \} (x = 2y + 2)\]
\[(x = 2(y + 1)) \{ y \leftarrow y + 1 \} (x = 2y)\]
\[\overline{\text{___________}}\]
\[(x = 2y) \{ x \leftarrow x + 2; y \leftarrow y + 1 \} (x = 2y)\]
Examples

Consider the following simple program:
\[ y \leftarrow 3; \ z \leftarrow x + y \]

Claim: If we have the pre-condition \( x = 1 \), we have the post-condition \( z = 4 \).
\[
(x = 1)\{y \leftarrow 3\}(x = 1 \land y = 3)
(x = 1 \land y = 3)\{z \leftarrow x + y\}(z = 4)
\]

\[
(x = 1)\{y \leftarrow 3; \ z \leftarrow x + y\}(z = 4)
\]

Consider the following simple program:
\[ x \leftarrow x + 2; \ y \leftarrow y + 1 \]

Claim: if we have the pre-condition \( x = 2y \), we have the post-condition \( x = 2y \).
\[
(x = 2y)\{x \leftarrow x + 2\}(x = 2y + 2)
(x = 2(y + 1))\{y \leftarrow y + 1\}(x = 2y)
\]

\[
(x = 2y)\{x \leftarrow x + 2; \ y \leftarrow y + 1\}(x = 2y)
\]

This is called an \textit{invariant condition} (or just an invariant)
Branches

If-then:
\[(p \land B) \{S\} q\]
\[(p \land \neg B) \rightarrow q\]

\[p\{\text{if } B \text{ then } S\} q\]

Example:
Let \(x = 7\), and consider code:
\[
\{\text{if } y < x \text{ then } y \leftarrow x\}
\]
Show \(y \geq 7\).
\[(x = 7) \land (y < x) \{y \leftarrow x\} (y \geq 7)\]
\[(x = 7) \land (y \geq x) \rightarrow (y \geq 7)\]
\[(x = 7)\{\text{if } y < x \text{ then } y \leftarrow x\} (y \geq 7)\]
Branches

If-then:

\[(p \land B)\{S\}q\]
\[(p \land \neg B) \rightarrow q\]

\[p\{\text{if } B \text{ then } S\}\{q\}\]

If-then-else:

\[(p \land B)\{S_1\}q\]
\[(p \land \neg B)\{S_2\}q\]

\[p\{\text{if } B \text{ then } S_1 \text{ else } S_2\}\{q\}\]

Example:

Let \(x = 7\), and consider code:

\[\{\text{if } y < x \text{ then } y \leftarrow x\}\]

Show \(y \geq 7\).
Branches

If-then:
\[(p \land B)\{S\}q\]
\[(p \land \neg B) \rightarrow q\]
\[
\]
\[p\{\text{if } B \text{ then } S\}q\]

If-then-else:
\[(p \land B)\{S_1\}q\]
\[(p \land \neg B)\{S_2\}q\]
\[
\]
\[p\{\text{if } B \text{ then } S_1 \text{ else } S_2\}q\]

Example:
Let \(x = 7\), and consider code:
\{if \(y < x\) then \(y \leftarrow x\}\}
Show \(y \geq 7\).
Branches

If-then:
\[(p \land B) \{S\} q\]
\[(p \land \neg B) \rightarrow q\]

\[p\{\text{if } B \text{ then } S\} q\]

If-then-else:
\[(p \land B) \{S_1\} q\]
\[(p \land \neg B) \{S_2\} q\]

\[p \{\text{if } B \text{ then } S_1 \text{ else } S_2\} q\]

Example:
Let \(x = 7\), and consider code:
\{if \(y < x\) then \(y \leftarrow x\)\}
Show \(y \geq 7\).

\((x = 7 \land y < x) \{y \leftarrow x\} (y \geq 7)\)
Branches

If-then:
\[(p \land B)\{S\}q\]
\[(p \land \neg B) \rightarrow q\]

\[p\{\text{if } B \text{ then } S\}q\]

If-then-else:
\[(p \land B)\{S_1\}q\]
\[(p \land \neg B)\{S_2\}q\]

\[p\{\text{if } B \text{ then } S_1 \text{ else } S_2\}q\]

Example:
Let \(x = 7\), and consider code:
\{if \(y < x\) then \(y \leftarrow x\}\}
Show \(y \geq 7\).

\[(x = 7 \land y < x)\{y \leftarrow x\}(y \geq 7)\]
\[(x = 7 \land y \geq x) \rightarrow (y \geq 7)\]
Branches

If-then:
\[(p \land B)\{S\}q\]
\[(p \land \neg B) \rightarrow q\]

\[p\{\text{if } B \text{ then } S\}q\]

If-then-else:
\[(p \land B)\{S_1\}q\]
\[(p \land \neg B)\{S_2\}q\]

\[p\{\text{if } B \text{ then } S_1 \text{ else } S_2\}q\]

Example:
Let \(x = 7\), and consider code:
\{if \(y < x\) then \(y \leftarrow x\}\}
Show \(y \geq 7\).

\[(x = 7 \land y < x)\{y \leftarrow x\}(y \geq 7)\]
\[(x = 7 \land y \geq x) \rightarrow (y \geq 7)\]
\[(x = 7)\{\text{if } y < x \text{ then } y \leftarrow x\}(y \geq 7)\]
While loops

While loop:
$(p \land B)\{S\}p$

\[
\begin{array}{c}
\hline
p\{\text{while } B \text{ do } S\}(p \land \neg B)
\end{array}
\]

We call $p$ a loop invariant.

Note that the correctness of this inference rule technically requires a mathematical induction.
While loops

While loop:
\[(p \land B)\{S\}p\]

\[
\frac{
}{p\{\text{while } B \text{ do } S\}(p \land \neg B)}
\]

We call \(p\) a loop invariant.
While loops

While loop:
\[(p \land B)\{S\}p\]

\[
\]

\[p\{\text{while } B \text{ do } S\}(p \land \neg B)\]

We call \(p\) a loop invariant.
Note that the correctness of this inference rule technically requires a mathematical induction.
Example: $n!$

\[
(p \land B)\{S\}p
\]

\[
\frac{(p \land B)\{S\}p}{p\{\text{while } B \text{ do } S\}(p \land \lnot B)}
\]
Example: \( n! \)

\[
(p \land B) \{ S \} p \\
\overline{\phantom{(p \land B) \{ S \} p}} \\
p \{ \text{while } B \text{ do } S \} (p \land \neg B)
\]

Find a loop invariant for the following program, and prove the program computes \( n! \)

\[
i \leftarrow 1; \\
f \leftarrow 1; \\
\text{while } i < n \text{ do} \\
\quad i \leftarrow i + 1 \\
\quad f \leftarrow f \cdot i
\]
Example: $n!$

$$(p \land B)\{S\}p$$

$$(p \land B)\{S\}(p \land \neg B)$$

Find a loop invariant for the following program, and prove the program computes $n!$

$i \leftarrow 1;$
$f \leftarrow 1;$
while $i < n$ do
    $i \leftarrow i + 1$
    $f \leftarrow f \cdot i$

$p = (f = i! \land i \leq n)$  (Loop invariant)
Example: $n!$

$$(p \land B)\{S\}p$$

$$(p \land B)\{\text{while } B \text{ do } S\}(p \land \neg B)$$

Find a loop invariant for the following program, and prove the program computes $n!$

$i \leftarrow 1$;
$f \leftarrow 1$;
while $i < n$ do
    $i \leftarrow i + 1$
    $f \leftarrow f \cdot i$

$p = (f = i! \land i \leq n)$
$B = (i < n)$

(Loop invariant)
(Loop invariant)
Example: \( n! \)

\[
(p \land B)\{S\}p
\]

\[
p\{\text{while } B \text{ do } S\}(p \land \neg B)
\]

Find a loop invariant for the following program, and prove the program computes \( n! \):

\[
i \leftarrow 1;
\]

\[
f \leftarrow 1;
\]

while \( i < n \) do

\[
i \leftarrow i + 1
\]

\[
f \leftarrow f \cdot i
\]

\[
p = (f = i! \land i \leq n)
\]

(Loop invariant)

\[
B = (i < n)
\]

(Loop condition)

\[
(f = i! \land i \leq n) \land (i \geq n)
\]
Example: \( n! \)

\[(p \land B)\{S\}p\]

\[p\{\text{while } B \text{ do } S\}(p \land \neg B)\]

Find a loop invariant for the following program, and prove the program computes \( n! \):

\[i \leftarrow 1;\]
\[f \leftarrow 1;\]
while \( i < n \) do
\[i \leftarrow i + 1\]
\[f \leftarrow f \cdot i\]

\[p = (f = i! \land i \leq n) \quad \text{(Loop invariant)}\]
\[B = (i < n) \quad \text{(Branch condition)}\]

\[(f = i! \land i \leq n) \quad \{\text{while } (i < n) \text{ do } S\} \quad (f = i! \land i \leq n) \land (i \geq n)\]

\[\equiv (f = i! \land i = n)\]
Example: \( n! \)

\[
(p \land B) \{S\} p \\
\underline{p\{\text{while } B \text{ do } S\}(p \land \neg B)}
\]

Find a loop invariant for the following program, and prove the program computes \( n! \):

\[
i \leftarrow 1; \\
f \leftarrow 1; \\
\text{while } i < n \text{ do} \\
\quad i \leftarrow i + 1 \\
\quad f \leftarrow f \cdot i
\]

\[
p = (f = i! \land i \leq n) \quad \text{(Loop invariant)}
\]

\[
B = (i < n) \quad \text{(Branch condition)}
\]

\[
(f = i! \land i \leq n) \land (i \geq n) \\
\equiv (f = i! \land i = n) \equiv f = n!
\]
Example: $n!$

\[(p \land B)\{S\}p\]

\[p\{\text{while } B \text{ do } S\}(p \land \neg B)\]

Find a loop invariant for the following program, and prove the program computes $n!$

\[i \leftarrow 1;\]
\[f \leftarrow 1;\]
while $i < n$ do
  \[i \leftarrow i + 1\]
  \[f \leftarrow f \cdot i\]

$p = (f = i! \land i \leq n)$ \hspace{1cm} (Loop invariant)

$B = (i < n)$ \hspace{1cm} (Branch condition)

(f = i! \land i \leq n \land i < n) \quad \{i \leftarrow i + 1; f \leftarrow f \cdot i\} \quad (f = i! \land i \leq n)$

(f = i! \land i \leq n) \quad \{\text{while } (i < n) \text{ do } S\} \quad (f = i! \land i \leq n) \land (i \geq n)

(\equiv (f = i! \land i = n) \equiv f = n!)}
Example: $n!$

$$(p \land B)\{S\}p$$

$$p\{\text{while } B \text{ do } S\}(p \land \neg B)$$

Find a loop invariant for the following program, and prove the program computes $n!$

$i \leftarrow 1$;
$f \leftarrow 1$;

while $i < n$ do

$i \leftarrow i + 1$
$f \leftarrow f \cdot i$

$p = (f = i! \land i \leq n)$ (Loop invariant)
$B = (i < n)$ (Branch condition)

$$\begin{align*}
(f = i! \land i \leq n \land i < n) & \quad \{i \leftarrow i + 1\} \\
(f = i! \land i \leq n) & \quad (f = (i - 1)! \land i \leq n)
\end{align*}$$

$$\begin{align*}
(f = i! \land i \leq n \land i < n) & \quad \{i \leftarrow i + 1; f \leftarrow f \cdot i\} \\
(f = i! \land i \leq n) & \quad (f = i! \land i \leq n) \\
(f = i! \land i \leq n \land i < n) & \quad \{\text{while } (i < n) \text{ do } S\} \\
(f = i! \land i \leq n) & \quad (f = i! \land i \leq n) \land (i \geq n) \\
& \equiv (f = i! \land i = n) \equiv f = n!
\end{align*}$$
Example: $n!$

$(p \land B)\{S\}p$

\[\overline{p\{\text{while } B \text{ do } S\}(p \land \neg B)}\]

Find a loop invariant for the following program, and prove the program computes $n!$

\[i \leftarrow 1;\]
\[f \leftarrow 1;\]
while $i < n$ do
  \[i \leftarrow i + 1\]
  \[f \leftarrow f \cdot i\]

\[p = (f = i! \land i \leq n)\] (Loop invariant)
\[B = (i < n)\] (Branch condition)

\[
(f = i! \land i \leq n \land i < n) \quad \{i \leftarrow i + 1\} \quad (f = (i - 1)! \land i \leq n)
\]
\[
(f = (i - 1)! \land i \leq n) \quad \{f \leftarrow f \cdot i\} \quad (f = i! \land i \leq n)
\]
\[
(f = i! \land i \leq n \land i < n) \quad \{i \leftarrow i + 1; f \leftarrow f \cdot i\} \quad (f = i! \land i \leq n)
\]
\[
(f = i! \land i \leq n) \quad \{\text{while } (i < n) \text{ do } S\} \quad (f = i! \land i \leq n) \land (i \geq n)
\]
\[
\equiv (f = i! \land i = n) \equiv f = n!
\]