Warm-Up Problem

Determine whether or not the following is satisfied under partial correctness.

$(\!|\ \mathrm{true}\ |\!)$

```
x = y ;
if (y > 1){
    x = x - 1;
} else {
    x = x * x + 1;
}
```

$(\!|\ (x > 0)\ |\!)$

Is the Hoare triple still satisfied under partial correctness if we replace $x = y$ with $y = x$?

Program Verification
While Loops

Carmen Bruni

Lecture 19

Based on slides by Jonathan Buss, Lila Kari, Anna Lubiw and Steve Wolfman with thanks to B. Bonakdarpour, A. Gao, D. Maftuleac, C. Roberts, R. Trefler, and P. Van Beek

Last Time

- Understand and use implied statements as needed.
- Prove that a Hoare triple is satisfied under partial correctness for a program containing assignment and conditional statements.

Learning Goals

- Partial correctness for while loops
- Determine whether a given formula is an invariant for a while loop.
- Find an invariant for a given while loop.
- Prove that a Hoare triple is satisfied under partial correctness for a program containing while loops.

`While`-Loops and Total Correctness

Total correctness

A triple $(\!|\, P \,|\!)\ C\ (\!|\, Q \,|\!)$ is satisfied under total correctness, denoted

$$\models_{\texttt{tot}} (\!|\, P \,|\!)\ C\ (\!|\, Q \,|\!)\ ,$$

if and only if

for every state s that satisfies P,

execution of C starting from state s terminates,

and the resulting state s' satisfies Q.

Total Correctness = Partial Correctness + Termination

Examples for Partial and Total Correctness

Example 1. Total correctness satisfied:

$$\llparenthesis\, x = 1 \,\rrparenthesis$$
```
y = x ;
```
$$\llparenthesis\, y = 1 \,\rrparenthesis$$

Example 2. Neither total nor partial correctness:

$$\llparenthesis\, x = 1 \,\rrparenthesis$$
```
y = x ;
```
$$\llparenthesis\, y = 2 \,\rrparenthesis$$

Example 3. Infinite loop (partial correctness)

$(\!|\ x = 1\ |\!)$
```
while (true) {
    x = 0 ;
}
```
$(\!|\ x > 0\ |\!)$

Example 4. Total correctness

$(\!|\, x \geq 0 \,|\!)$

```
y = 1 ;
z = 0 ;
while (z != x) {
    z = z + 1 ;
    y = y * z ;
}
```

$(\!|\, y = x! \,|\!)$

What happens if we remove the precondition?

Example 5. No correctness, because input altered ("consumed")

$(\!|\ \mathrm{true}\ |\!)$

```
y = 1 ;
while (x != 0) {
    y = y * x ;
    x = x - 1 ;
}
```

$(\!|\ y = x!\ |\!)$

Proving Correctness: Recap and Overview

- Total correctness is our goal.
- We usually prove it by proving partial correctness and termination separately.
  - For partial correctness, we introduced sound inference rules.
  - For total correctness, we shall use *ad hoc* reasoning, which suffices for our examples.
    (In general, total correctness is undecidable.)

Our focus on partial correctness may seem strange. It's not the condition we want to justify.

But experience has shown it is useful to think about partial correctness separately from termination.

Inference Rule: Partial-while

"Partial while": do not (yet) require termination.

$$\frac{(\!|\,(I \wedge B)\,|\!)\ C\ (\!|\,I\,|\!)}{(\!|\,I\,|\!)\ \texttt{while}\ \texttt{(}B\texttt{)}\ C\ (\!|\,(I \wedge (\neg B))\,|\!)}\ \text{(partial-}\textit{\texttt{while}}\text{)}$$

In words:

> If the code C satisfies the triple $(\!|\,I \wedge B\,|\!)\ C\ (\!|\,I\,|\!)$,
> and I is true at the start of the `while`-loop,
> then no matter how many times we execute C,
> condition I will still be true.

Condition I is called a *loop invariant*.

After the `while`-loop terminates, $(\neg B)$ is also true.

Annotations for Partial-`while`

$(\!| P |\!)$
$(\!| I |\!)$ Implied (a)
`while (` B `) {`
  $(\!| (I \wedge B) |\!)$ partial-`while`
  C
  $(\!| I |\!)$ $\longleftarrow$ *to be justified, based on* C
`}`
$(\!| (I \wedge (\neg B)) |\!)$ partial-`while`
$(\!| Q |\!)$ Implied (b)

(a) Prove $(P \to I)$ (precondition P implies the loop invariant)

(b) Prove $((I \wedge (\neg B)) \to Q)$ (exit condition implies postcondition)

We need to determine I!!

Loop Invariants

A *loop invariant* is an assertion (condition) that is true both *before* and *after* each execution of the body of a loop.

- True before the `while`-loop begins.
- True after the `while`-loop ends.
- Expresses a relationship among the variables used within the body of the loop. Some of these variables will have their values changed within the loop.
- An invariant may or may not be useful in proving termination (to discuss later).

Loop Invariant Example

What are some loop invariants for the following code?

$\llparenthesis\ \mathrm{true}\ \rrparenthesis$

```
z = 1;
while (z * z < 16){
    z = z + 1;
}
```

$\llparenthesis\ (z = 4)\ \rrparenthesis$

Loop Invariant Example

What are some loop invariants for the following code?

$(\!|\ \text{true}\ |\!)$

```
z = 1;
while (z * z < 16){
    z = z + 1;
}
```

$(\!|\ (z = 4)\ |\!)$

The code has as loop invariants $(z \geq 1)$ and $((z \cdot z) \leq 16)$ (note there are many others as well).

Loop Invariant Example

Let's attempt to annotate the code using the first invariant:

$(\!|\ \text{true}\ |\!)$

```
z = 1;
```

$(\!|\ (z \geq 1)\ |\!)$ Assignment

```
while (z * z < 16){
```

$(\!|\ ((z \geq 1) \wedge ((z \cdot z) < 16))\ |\!)$ Partial-While

```
    z = z + 1;
```

$(\!|\ (z \geq 1)\ |\!)$

```
}
```

$(\!|\ \big((z \geq 1) \wedge \big(\neg((z \cdot z) < 16)\big)\big)\ |\!)$ Partial-While

$(\!|\ (z = 4)\ |\!)$???

Loop Invariant Example

Let's attempt to annotate the code using the first invariant:

$(\!|\ \mathrm{true}\ |\!)$

```
z = 1;
```

$(\!|\ (z \geq 1)\ |\!)$ Assignment

```
while (z * z < 16){
```

$\quad(\!|\ ((z \geq 1) \wedge ((z \cdot z) < 16))\ |\!)$ Partial-While

```
    z = z + 1;
```

$\quad(\!|\ (z \geq 1)\ |\!)$

```
}
```

$(\!|\ \big((z \geq 1) \wedge (\neg((z \cdot z) < 16))\big)\ |\!)$ Partial-While

$(\!|\ (z = 4)\ |\!)$???

Notice that the first invariant $(z \geq 1)$ is not very useful as we will not be able to prove $\big((z \geq 1) \wedge (\neg((z \cdot z) < 16))\big)$ implies $(z = 4)$.

Loop Invariant Example

What about the second invariant we listed?

$(\!|\ \mathrm{true}\ |\!)$

```
z = 1;
```

$(\!|\ ((z \cdot z) \leq 16)\ |\!)$ Assignment

```
while (z * z < 16){
```

$(\!|\ (((z \cdot z) \leq 16) \wedge ((z \cdot z) < 16))\ |\!)$ Partial-While

```
    z = z + 1;
```

$(\!|\ ((z \cdot z) \leq 16)\ |\!)$

```
}
```

$(\!|\ \big(((z \cdot z) \leq 16) \wedge \big(\neg((z \cdot z) < 16)\big)\big)\ |\!)$ Partial-While

$(\!|\ (z = 4)\ |\!)$???

Loop Invariant Example

What about the second invariant we listed?

$(\!|\ \text{true}\ |\!)$

```
z = 1;
```

$(\!|\ ((z \cdot z) \leq 16)\ |\!)$ Assignment

```
while (z * z < 16){
```

$\quad (\!|\ (((z \cdot z) \leq 16) \wedge ((z \cdot z) < 16))\ |\!)$ Partial-While

```
    z = z + 1;
```

$\quad (\!|\ ((z \cdot z) \leq 16)\ |\!)$

```
}
```

$(\!|\ \big(((z \cdot z) \leq 16) \wedge \big(\neg((z \cdot z) < 16)\big)\big)\ |\!)$ Partial-While

$(\!|\ (z = 4)\ |\!)$???

The second loop invariant $((z \cdot z) \leq 16)$ on its own also isn't useful. The last two values prove that $z^2 = 16$ but they cannot alone determine if $z = 4$ or $z = -4$.

Loop Invariant Example

What about the second invariant we listed?

$(\!|\ \mathrm{true}\ |\!)$

```
z = 1;
```

$(\!|\ ((z \cdot z) \leq 16)\ |\!)$ Assignment

```
while (z * z < 16){
```

$(\!|\ (((z \cdot z) \leq 16) \wedge ((z \cdot z) < 16))\ |\!)$ Partial-While

```
    z = z + 1;
```

$(\!|\ ((z \cdot z) \leq 16)\ |\!)$

```
}
```

$(\!|\ \big(((z \cdot z) \leq 16) \wedge \big(\neg((z \cdot z) < 16)\big)\big)\ |\!)$ Partial-While

$(\!|\ (z = 4)\ |\!)$???

The second loop invariant $((z \cdot z) \leq 16)$ on its own also isn't useful. The last two values prove that $z^2 = 16$ but they cannot alone determine if $z = 4$ or $z = -4$.

How can we fix this problem?

Loop Invariant Example

Let's combine the two!

$(\!|\ \mathrm{true}\ |\!)$

$(\!|\ ((1 \geq 1) \land ((1 \cdot 1) \leq 16))\ |\!)$ Implied(a)

`z = 1;`

$(\!|\ ((z \geq 1) \land ((z \cdot z) \leq 16))\ |\!)$ Assignment

`while (z * z < 16){`

$\quad (\!|\ \big(((z \geq 1) \land ((z \cdot z) \leq 16)) \land ((z \cdot z) < 16)\big)\ |\!)$ Partial-While

$\quad (\!|\ (((z+1) \geq 1) \land (((z+1) \cdot (z+1)) \leq 16))\ |\!)$ Implied (b)

`    z = z + 1;`

$\quad (\!|\ ((z \geq 1) \land ((z \cdot z) \leq 16))\ |\!)$ Assignment

`}`

$(\!|\ \big(((z \geq 1) \land ((z \cdot z) \leq 16)) \land \big(\neg((z \cdot z) < 16)\big)\big)\ |\!)$ Partial-While

$(\!|\ (z = 4)\ |\!)$ Implied (c)

Loop Invariant Example

Let's combine the two!

$(\!|\ \mathrm{true}\ |\!)$

$(\!|\ ((1 \geq 1) \wedge ((1 \cdot 1) \leq 16))\ |\!)$ Implied(a)

```
z = 1;
```

$(\!|\ ((z \geq 1) \wedge ((z \cdot z) \leq 16))\ |\!)$ Assignment

```
while (z * z < 16){
```

$(\!|\ \big(((z \geq 1) \wedge ((z \cdot z) \leq 16)) \wedge ((z \cdot z) < 16)\big)\ |\!)$ Partial-While

$(\!|\ (((z+1) \geq 1) \wedge (((z+1) \cdot (z+1)) \leq 16))\ |\!)$ Implied (b)

```
    z = z + 1;
```

$(\!|\ ((z \geq 1) \wedge ((z \cdot z) \leq 16))\ |\!)$ Assignment

```
}
```

$(\!|\ \big(((z \geq 1) \wedge ((z \cdot z) \leq 16)) \wedge \big(\neg((z \cdot z) < 16)\big)\big)\ |\!)$ Partial-While

$(\!|\ (z = 4)\ |\!)$ Implied (c)

This works. Implied (c) now follows since $z^2 = 16$ and $z \geq 1$ gives the result and Implied (b) is true since $z^2 < 16$ implies that $z < 4$ and with $z \geq 1$ gives $(z+1)^2 \leq 16$.

Example: Finding a loop invariant

$(\!|\ (x \geq 0)\ |\!)$

```
y = 1 ;
z = 0 ;
while (z != x) {
    z = z + 1 ;
    y = y * z ;
}
```

$(\!|\ (y = x!)\ |\!)$

Let's construct a table of states for this loop.

Example: Finding a loop invariant

$(\!| (x \geq 0) |\!)$

```
y = 1 ;
z = 0 ;
while (z != x) {
    z = z + 1 ;
    y = y * z ;
}
```

$(\!| (y = x!) |\!)$

At the `while` statement:

x	y	z	$z \neq x$
5	1	0	true
5	1	1	true
5	2	2	true
5	6	3	true
5	24	4	true
5	120	5	false

Example: Finding a loop invariant

$(\!|\ (x \geq 0)\ |\!)$

```
y = 1 ;
z = 0 ;
while (z != x) {
    z = z + 1 ;
    y = y * z ;
}
```

$(\!|\ (y = x!)\ |\!)$

At the `while` statement:

x	y	z	$z \neq x$
5	1	0	true
5	1	1	true
5	2	2	true
5	6	3	true
5	24	4	true
5	120	5	false

The loop invariant we will attempt to use is $(y = z!)$

Why are $y \geq z$ or $x \geq 0$ not useful?

> These conjoined with $(\neg(z \neq x))$ won't be able to prove that $(y = x!)$ is true.

Annotations Inside a `while`-Loop

1. First annotate code using the while-loop inference rule, and any other control rules, such as `if-then`.
2. Then work bottom-up ("push up") through program code.
   - Apply inference rule appropriate for the specific line of code, or
   - Note a new assertion ("implied") to be proven separately.
3. Prove the implied assertions using the inference rules of ordinary logic.

Example: annotations for partial-while

Annotate by partial-while, with chosen invariant $(y = z!)$.

$(\!|\ (x \geq 0)\ |\!)$

```
y = 1 ;

z = 0 ;
```

$(\!|\ (y = z!)\ |\!)$ *[justification required]*

```
while ((z != x)) {
```

$(\!|\ \big((y = z!) \wedge (\neg(z = x))\big)\ |\!)$ partial-while $((\!|\ (I \wedge B)\ |\!))$

```
    z = z + 1 ;

    y = y * z ;
```

$(\!|\ (y = z!)\ |\!)$ *[justification required]*

```
}
```

$(\!|\ \Big((y = z!) \wedge (\neg(\neg(z = x)))\Big)\ |\!)$ partial-while $((\!|\ (I \wedge (\neg B))\ |\!))$

$(\!|\ (y = x!)\ |\!)$

Example: annotations for partial-while

Annotate assignment statements (bottom-up).

$(\!|\ (x \geq 0)\ |\!)$

$(\!|\ (1 = 0!)\ |\!)$

`y = 1 ;`

$(\!|\ (y = 0!)\ |\!)$ assignment

`z = 0 ;`

$(\!|\ (y = z!)\ |\!)$ assignment

`while ((z != x)) {`

  $(\!|\ ((y = z!) \wedge (\neg(z = x)))\ |\!)$ partial-while

  $(\!|\ ((y \cdot (z+1)) = (z+1)!)\ |\!)$

  `z = z + 1 ;`

  $(\!|\ ((y \cdot z) = z!)\ |\!)$ assignment

  `y = y * z ;`

  $(\!|\ (y = z!)\ |\!)$ assignment

`}`

$(\!|\ \big((y = z!) \wedge (\neg(\neg(z = x)))\big)\ |\!)$ partial-while

$(\!|\ (y = x!)\ |\!)$

Example: annotations for partial-while

Note the required implied conditions.

$(\!\| (x \geq 0) \|\!)$	
$(\!\| (1 = 0!) \|\!)$	implied (a)
`y = 1 ;`	
$(\!\| (y = 0!) \|\!)$	assignment
`z = 0 ;`	
$(\!\| (y = z!) \|\!)$	assignment
`while ((z != x)) {`	
$(\!\| ((y = z!) \wedge (\neg(z = x))) \|\!)$	partial-while
$(\!\| ((y \cdot (z + 1)) = (z + 1)!) \|\!)$	implied (b)
`z = z + 1 ;`	
$(\!\| ((y \cdot z) = z!) \|\!)$	assignment
`y = y * z ;`	
$(\!\| (y = z!) \|\!)$	assignment
`}`	
$(\!\| \big((y = z!) \wedge (\neg(\neg(z = x)))\big) \|\!)$	partial-while
$(\!\| (y = x!) \|\!)$	implied (c)

Proofs

Proof of implied (a): $(x \geq 0) \vdash (1 = 0!)$.

By definition of factorial.

Proof of implied (b): $\big((y = z!) \wedge \neg(z = x)\big) \vdash ((y \cdot (z+1)) = (z+1)!)$

Since $y = z!$, multiplying both sides by $(z+1)$ and by the definition of factorial, we see that $y(z+1) = (z+1)!$

Proof of implied (c): $\big((y = z!) \wedge (z = x)\big) \vdash (y = x!)$.

Since $y = z!$ and $z = x$, a simple substitution completes the proof.

Total Correctness (Termination)

Total Correctness = Partial Correctness + Termination

Only `while`-loops can be responsible for non-termination in our programming language.

(In general, recursion can also cause it).

Proving termination:
For each `while`-loop in the program,

Identify an integer expression which is

- *always* non-negative
- such that the value *decreases* every time through the while-loop.

This expression is called a **variant**. This expression usually corresponds with the while loop condition called the **loop guard**.

Example For Total Correctness

The code below has a "*loop guard*" of $z \neq x$,
which is equivalent to $x - z \neq 0$.

What happens to the value of $x - z$ during execution?

$(\!|\ (x \geq 0)\ |\!)$

```
y = 1 ;
z = 0 ;
```

At start of loop: $x - z = x$ and $x \geq 0$
hence $x - z \geq 0$.

```
while ( z != x ) {
    z = z + 1 ;
    y = y * z ;
}
```

$(\!|\ (y = x!)\ |\!)$

Example For Total Correctness

The code below has a "*loop guard*" of $z \neq x$,
which is equivalent to $x - z \neq 0$.

What happens to the value of $x - z$ during execution?

$\llparenthesis\ (x \geq 0)\ \rrparenthesis$

```
y = 1 ;
z = 0 ;
```

At start of loop: $x - z = x$ and $x \geq 0$
hence $x - z \geq 0$.

```
while ( z != x ) {
    z = z + 1 ;
    y = y * z ;
}
```

$x - z$ decreases by 1

$\llparenthesis\ (y = x!)\ \rrparenthesis$

Example For Total Correctness

The code below has a "*loop guard*" of $z \neq x$,
which is equivalent to $x - z \neq 0$.

What happens to the value of $x - z$ during execution?

$(\!| \, (x \geq 0) \, |\!)$

```
y = 1 ;
z = 0 ;
```

At start of loop: $x - z = x$ and $x \geq 0$
hence $x - z \geq 0$.

```
while ( z != x ) {
    z = z + 1 ;
    y = y * z ;
}
```

$x - z$ decreases by 1
$x - z$ unchanged

$(\!| \, (y = x!) \, |\!)$

Example For Total Correctness

The code below has a "*loop guard*" of $z \neq x$,
which is equivalent to $x - z \neq 0$.

What happens to the value of $x - z$ during execution?

$(\!| \, (x \geq 0) \, |\!)$

```
y = 1 ;
z = 0 ;
```

At start of loop: $x - z = x$ and $x \geq 0$
hence $x - z \geq 0$.

```
while ( z != x ) {
    z = z + 1 ;
    y = y * z ;
}
```

$x - z$ decreases by 1
$x - z$ unchanged

$(\!| \, (y = x!) \, |\!)$

Thus the value of $x - z$ will eventually reach 0.
The loop then exits and the program terminates.

Proof of Total Correctness

We choose the **variant** $x - z$.

At the start of the loop, $x - z \geq 0$:

- Precondition: $x \geq 0$.
- Assignment $z \leftarrow 0$.

Each time through the loop:

- x doesn't change: no assignment to it.
- z increases by 1, by assignment.
- Thus $x - z$ decreases by 1.

Thus the value of $x - z$ will eventually reach 0.

When $x - z = 0$, the guard $z \mathrel{!}= x$ ends the loop.

Example 2 (Back to Partial-while)

Prove the following is satisfied under partial correctness. Draw the trace of the following loop to help find an invariant.

$$\llparenthesis\ ((n \geq 0) \wedge (a \geq 0))\ \rrparenthesis$$

```
s = 1 ;
i = 0 ;
while (i < n) {
    s = s * a ;
    i = i + 1 ;
}
```

$$\llparenthesis\ (s = a^n)\ \rrparenthesis$$

Example 2 (Back to Partial-`while`)

Prove the following is satisfied under partial correctness. Draw the trace of the following loop to help find an invariant.

$(\!|\ ((n \geq 0) \wedge (a \geq 0))\ |\!)$

```
s = 1 ;
i = 0 ;
while (i < n) {
    s = s * a ;
    i = i + 1 ;
}
```

$(\!|\ (s = a^n)\ |\!)$

Trace of the loop:

a	n	i	s
2	3	0	1
2	3	1	1∗2
2	3	2	1∗2∗2
2	3	3	1∗2∗2∗2

Invariant

For our example

$\llparenthesis\, ((n \geq 0) \wedge (a \geq 0))\, \rrparenthesis$

```
s = 1 ;
i = 0 ;
while (i < n) {
    s = s * a ;
    i = i + 1 ;
}
```

$\llparenthesis\, (s = a^n)\, \rrparenthesis$

Trace of the loop:

a	n	i	s
2	3	0	1
2	3	1	1∗2
2	3	2	1∗2∗2
2	3	3	1∗2∗2∗2

Let's try the invariant $(s = a^i)$

Example 2: Testing the invariant

Using $(s = a^i)$ as an invariant yields the annotations shown at right.

Next, we want to

- Push up for assignments
- Prove the implications

But: implied (c) is false!

We must use a different invariant.

$(\!|\ ((n \geq 0) \wedge (a \geq 0))\ |\!)$
$(\!|\ \dots\ |\!)$
```
s = 1 ;
```
$(\!|\ \dots\ |\!)$
```
i = 0 ;
```
$(\!|\ (s = a^i)\ |\!)$
```
while (i < n) {
```
$(\!|\ ((s = a^i) \wedge (i < n))\ |\!)$ partial-while
$(\!|\ \dots\ |\!)$
```
  s = s * a ;
```
$(\!|\ \dots\ |\!)$
```
  i = i + 1 ;
```
$(\!|\ (s = a^i)\ |\!)$
```
}
```
$(\!|\ ((s = a^i) \wedge (i \geq n))\ |\!)$ partial-while
$(\!|\ (s = a^n)\ |\!)$ implied (c)

Example 2: Adjusted invariant

Try a new invariant:

$((s = a^i) \wedge (i \leq n))$

Now the "implied" conditions are actually true, and the proof can succeed.

$(\!\| ((n \geq 0) \wedge (a \geq 0)) \|\!)$	
$(\!\| ((1 = a^0) \wedge (0 \leq n)) \|\!)$	implied (a)
`s = 1 ;`	
$(\!\| ((s = a^0) \wedge (0 \leq n)) \|\!)$	assignment
`i = 0 ;`	
$(\!\| \big((s = a^i) \wedge (i \leq n)\big) \|\!)$	assignment
`while (i < n) {`	
$\quad (\!\| \Big(\big((s = a^i) \wedge (i \leq n)\big) \wedge (i < n)\Big) \|\!)$	partial-while
$\quad (\!\| (((s \cdot a) = a^{i+1}) \wedge ((i+1) \leq n)) \|\!)$	implied (b)
`    s = s * a ;`	
$\quad (\!\| ((s = a^{i+1}) \wedge ((i+1) \leq n)) \|\!)$	assignment
`    i = i + 1 ;`	
$\quad (\!\| \big((s = a^i) \wedge (i \leq n)\big) \|\!)$	assignment
`}`	
$(\!\| \Big(\big((s = a^i) \wedge (i \leq n)\big) \wedge (i \geq n)\Big) \|\!)$	partial-while
$(\!\| (s = a^n) \|\!)$	implied (c)