Lecture 09/30/16

Lecturer: Xiaodi Wu

Reading: Chapter 1.3, Note on loop invariants.

Loop Invariant: Framework to analyze loops

Notations for Loop Invariant

State of computation : (**boolean predicate**) a ”snap-shot” of the computation; inter-relations of values of variables.

Program Statement : (**predicate transformers**) a program statement S causes state $\langle P \rangle$ to state $\langle Q \rangle$. (denoted $\langle P \rangle \, S \, \langle Q \rangle$).

Loop Invariant: Framework to analyze loops

Notations for Loop Invariant

State of computation : (**boolean predicate**) a "snap-shot" of the computation; inter-relations of values of variables.

Program Statement : (**predicate transformers**) a program statement S causes state $\langle P\rangle$ to state $\langle Q\rangle$. (denoted $\langle P\rangle\, S\, \langle Q\rangle$).

- State: $\langle x = 0; y = 1\rangle$, $\langle x = y^2\rangle$, ...
- Program Statement: $x \leftarrow x + 1$, $x \leftarrow x \times y$, ...

Loop Invariant: Framework to analyze loops

Notations for Loop Invariant

State of computation : (**boolean predicate**) a "snap-shot" of the computation; inter-relations of values of variables.

Program Statement : (**predicate transformers**) a program statement S causes state $\langle P \rangle$ to state $\langle Q \rangle$. (denoted $\langle P \rangle\, S \,\langle Q \rangle$).

- State: $\langle x = 0; y = 1 \rangle$, $\langle x = y^2 \rangle$, ...
- Program Statement: $x \leftarrow x + 1$, $x \leftarrow x \times y$, ...

Thus, any line in the program can be expressed as a statement that causes the state (before-line) to the state (after-line).

Loop Invariant

```
$\langle P \rangle$, the state before the loop
while Condition (C) do Body (B)
end while
$\langle Q \rangle$, the state after the loop
```

A **loop invariant** I is a boolean predicate that does not change during the execution of the loop.

- $P \rightarrow I$ before the loop.
- $\langle I \text{ and } C \rangle \, B \, \langle I \rangle$ in the loop.
- $(I \text{ and } \neg C) \rightarrow Q$ after the loop.

Loop Invariant: Example 1

Code 1

```
$s \leftarrow 0, k \leftarrow 0$
while $k < n + 1$ do
    $s \leftarrow s + k, k \leftarrow k + 1$
end while
```

Loop Invariant: Example 1

Code 1

```
$s \leftarrow 0, k \leftarrow 0$
**while** $k < n+1$ **do**
    $s \leftarrow s + k, k \leftarrow k + 1$
**end while**
```

- Invariant I: $s = k(k-1)/2 \wedge (0 \leq k \leq n+1)$.

Loop Invariant: Example 1

Code 1

```
$s \leftarrow 0, k \leftarrow 0$
while $k < n+1$ do
    $s \leftarrow s + k, k \leftarrow k+1$
end while
```

- Invariant I: $s = k(k-1)/2 \wedge (0 \leq k \leq n+1)$.
- P is $s = 0, k = 0$ and $P \rightarrow I$.

Loop Invariant: Example 1

Code 1

```
$s \leftarrow 0, k \leftarrow 0$
while $k < n + 1$ do
    $s \leftarrow s + k, k \leftarrow k + 1$
end while
```

- Invariant I: $s = k(k-1)/2 \wedge (0 \leq k \leq n+1)$.
- P is $s = 0, k = 0$ and $P \rightarrow I$.
- C is $k < n + 1$ and B is $s \leftarrow s + k, k \leftarrow k + 1$.

Loop Invariant: Example 1

Code 1

```
$s \leftarrow 0, k \leftarrow 0$
**while** $k < n + 1$ **do**
    $s \leftarrow s + k, k \leftarrow k + 1$
**end while**
```

- Invariant I: $s = k(k-1)/2 \wedge (0 \leq k \leq n+1)$.
- P is $s = 0, k = 0$ and $P \rightarrow I$.
- C is $k < n+1$ and B is $s \leftarrow s + k, k \leftarrow k + 1$.
- $\langle I \text{ and } C \rangle \, B \, \langle I \rangle$ in the loop.

Loop Invariant: Example 1

Code 1

```
$s \leftarrow 0, k \leftarrow 0$
while $k < n + 1$ do
    $s \leftarrow s + k, k \leftarrow k + 1$
end while
```

- Invariant I: $s = k(k-1)/2 \wedge (0 \leq k \leq n+1)$.
- P is $s = 0, k = 0$ and $P \rightarrow I$.
- C is $k < n + 1$ and B is $s \leftarrow s + k, k \leftarrow k + 1$.
- $\langle I$ and $C \rangle\, B \,\langle I \rangle$ in the loop.
- Q is $s = s(n) = n(n+1)/2$. (I and $\neg C$) $\rightarrow Q$ after the loop.