CSE 331
Software Design & Implementation

Kevin Zatloukal

Spring 2020

Lecture 3 – Reasoning about Loops

Quick Recap (5 min)

Full Correctness Toolkit

Correctness Toolkit

- Forward and backward reasoning for...
  - assignments
  - if statements
  - loops
  - (essentially) all code can be rewritten to use just these
- Forward / backward reasoning fill in post-/pre-condition
- Check places where assertions meet each other
  - top assertion must imply the bottom assertion

Checking Correctness of a Loop

Consider a while-loop (other loop forms not too different) with a loop invariant **I**.

```
{{ P }}
  S1

{{ Inv: I }}
  while (cond)
    S2

  S3
{{ Q }}
```

Informally, we need:

- **I** holds initially
- **I** holds each time around
- **Q** holds after we exit

Formally, we need validity of:

- {{ P }} `S1` {{ **I** }}
- {{ **I** and cond }} `S2` {{ **I** }}
- {{ **I** and not cond }} `S3` {{ Q }}

Q & A

Another Example

Example: partition array

Consider the following code to put the negative values at the beginning of array `b`:

```
{{ 0 <= n <= b.length }}
i = k = 0;
while (i != n) {
  if (b[i] < 0) {
    swap b[i], b[k];
    k = k + 1;
  }
  i = i + 1;
}
{{ b[0], ..., b[k-1] < 0 <= b[k], ..., b[n-1] }}
```

(Also: precondition is true throughout the code. I'll skip writing that to save space...)

(Also: b contains the same numbers since we use swaps.)

Example: partition array

Consider the following code to put the negative values at the beginning of array `b`:

```
{{ 0 <= n <= b.length }}
i = k = 0;
{{ Inv: b[0], ..., b[k-1] < 0 <= b[k], ..., b[i-1] }}
while (i != n) {
  if (b[i] < 0) {
    swap b[i], b[k];
    k = k + 1;
  }
  i = i + 1;
}
{{ b[0], ..., b[k-1] < 0 <= b[k], ..., b[n-1] }}
```

Example: partition array

Consider the following code to put the negative values at the beginning of array `b`:

{{ 0 <= n <= b.length }}

```
i = k = 0;
```

{{ Inv: b[0], ..., b[k-1] < 0 <= b[k], ..., b[i-1] }}

```
while (i != n) {
  if (b[i] < 0) {
    swap b[i], b[k];
    k = k + 1;
  }
  i = i + 1;
}
```

{{ b[0], ..., b[k-1] < 0 <= b[k], ..., b[n-1] }}

- `I` holds initially:
  - b[0], …, b[-1] is empty

Example: partition array

Consider the following code to put the negative values at the beginning of array `b`:

```
{{ 0 <= n <= b.length }}
i = k = 0;
{{ Inv: b[0], ..., b[k-1] < 0 <= b[k], ..., b[i-1] }}
while (i != n) {
  if (b[i] < 0) {
    swap b[i], b[k];
    k = k + 1;
  }
  i = i + 1;
}
{{ b[0], ..., b[k-1] < 0 <= b[k], ..., b[n-1] }}
```

- **`I`** holds initially
- **`I`** and i = n implies postcondition

Example: partition array

Consider the following code to put the negative values at the beginning of array `b`:

```
{{ 0 <= n <= b.length }}
i = k = 0;
{{ Inv: b[0], ..., b[k-1] < 0 <= b[k], ..., b[i-1] }}
while (i != n) {
  if (b[i] < 0) {
    swap b[i], b[k];
    k = k + 1;
  }
  i = i + 1;
}
{{ b[0], ..., b[k-1] < 0 <= b[k], ..., b[n-1] }}
```

- **`I`** holds initially
- **`I`** and i = n implies postcondition
- **`I`** is maintained by loop body?

Example: partition array

Consider the following code to put the negative values at the beginning of array `b`:

```
{{ 0 <= n <= b.length }}
i = k = 0;
{{ Inv: b[0], ..., b[k-1] < 0 <= b[k], ..., b[i-1] }}
while (i != n) {
  if (b[i] < 0) {
    swap b[i], b[k];
    k = k + 1;
  }
  i = i + 1;
  {{ b[0], ..., b[k-1] < 0 <= b[k], ..., b[i-1] }}
}
{{ b[0], ..., b[k-1] < 0 <= b[k], ..., b[n-1] }}
```

- **`I`** holds initially
- **`I`** and i = n implies postcondition
- **`I`** is maintained by loop body?

Example: partition array

Consider the following code to put the negative values at the beginning of array `b`:

```
{{ 0 <= n <= b.length }}
i = k = 0;
{{ Inv: b[0], ..., b[k-1] < 0 <= b[k], ..., b[i-1] }}
while (i != n) {
  if (b[i] < 0) {
    swap b[i], b[k];
    k = k + 1;
  }
  {{ b[0], ..., b[k-1] < 0 <= b[k], ..., b[i] }}
  i = i + 1;
}
{{ b[0], ..., b[k-1] < 0 <= b[k], ..., b[n-1] }}
```


- **`I`** holds initially
- **`I`** and i = n implies postcondition
- **`I`** is maintained by loop body?

Example: partition array

Consider the following code to put the negative values at the beginning of array `b`:

```
{{ 0 <= n <= b.length }}
i = k = 0;
{{ Inv: b[0], ..., b[k-1] < 0 <= b[k], ..., b[i-1] }}
while (i != n) {
   {{ b[0], ..., b[k-1] < 0 <= b[k], ..., b[i-1] }}
   if (b[i] < 0) {
     swap b[i], b[k];
     k = k + 1;
   }
   {{ b[0], ..., b[k-1] < 0 <= b[k], ..., b[i] }}
   i = i + 1;
}
{{ b[0], ..., b[k-1] < 0 <= b[k], ..., b[n-1] }}
```

- **`I`** holds initially
- **`I`** and i = n implies postcondition
- **`I`** is maintained by loop body?

Example: partition array

Consider the following code to put the negative values at the beginning of array `b`:

```
{{ 0 <= n <= b.length }}
i = k = 0;
{{ Inv: b[0], ..., b[k-1] < 0 <= b[k], ..., b[i-1] }}
while (i != n) {
   {{ b[0], ..., b[k-1] < 0 <= b[k], ..., b[i-1] }}
   if (b[i] < 0) {
     swap b[i], b[k];
     k = k + 1;
   } else {
   }
   {{ b[0], ..., b[k-1] < 0 <= b[k], ..., b[i] }}
   i = i + 1;
}
{{ b[0], ..., b[k-1] < 0 <= b[k], ..., b[n-1] }}
```

- **`I`** holds initially
- **`I`** and i = n implies postcondition
- **`I`** is maintained by loop body?

Example: partition array

Consider the following code to put the negative values at the beginning of array `b`:

```
{{ 0 <= n <= b.length }}
i = k = 0;
{{ Inv: b[0], ..., b[k-1] < 0 <= b[k], ..., b[i-1] }}
while (i != n) {
  if (b[i] < 0) {
    {{ b[0], ..., b[k-1] < 0 <= b[k], ..., b[i-1] and b[i] < 0 }}
    swap b[i], b[k];
    k = k + 1;
    {{ b[0], ..., b[k-1] < 0 <= b[k], ..., b[i] }}
  } else {
    {{ b[0], ..., b[k-1] < 0 <= b[k], ..., b[i-1] and b[i] >= 0 }}
    {{ b[0], ..., b[k-1] < 0 <= b[k], ..., b[i] }}
  }
  i = i + 1;
}
```

- **I** holds initially
- **I** and i = n implies postcondition
- **I** is maintained by loop body?

Example: partition array

Consider the following code to put the negative values at the beginning of array `b`:

```
{{ 0 <= n <= b.length }}
i = k = 0;
{{ Inv: b[0], ..., b[k-1] < 0 <= b[k], ..., b[i-1] }}
while (i != n) {
  if (b[i] < 0) {
    {{ b[0], ..., b[k-1] < 0 <= b[k], ..., b[i-1] and b[i] < 0 }}
    swap b[i], b[k];
    k = k + 1;
    {{ b[0], ..., b[k-1] < 0 <= b[k], ..., b[i] }}
  } else {
    {{ b[0], ..., b[k-1] < 0 <= b[k], ..., b[i-1] and b[i] >= 0 }}
    {{ b[0], ..., b[k-1] < 0 <= b[k], ..., b[i] }}
  }
  i = i + 1;
}
```

equivalent

- **I** holds initially
- **I** and i = n implies postcondition
- **I** is maintained by loop body?

Example: partition array

Consider the following code to put the negative values at the beginning of array `b`:

```
{{ 0 <= n <= b.length }}
i = k = 0;
{{ Inv: b[0], ..., b[k-1] < 0 <= b[k], ..., b[i-1] }}
while (i != n) {
  if (b[i] < 0) {
    {{ b[0], ..., b[k-1] < 0 <= b[k], ..., b[i-1] and b[i] < 0 }}
    swap b[i], b[k];
    k = k + 1;
    {{ b[0], ..., b[k-1] < 0 <= b[k], ..., b[i] }}
  }
  i = i + 1;
}
{{ b[0], ..., b[k-1] < 0 <= b[k], ..., b[n-1] }}
```

- **`I`** holds initially
- **`I`** and i = n implies postcondition
- **`I`** is maintained by loop body?

Remain to check this…

Example: partition array

Consider the following code to put the negative values at the beginning of array `b`:

```
{{ 0 <= n <= b.length }}
i = k = 0;
{{ Inv: b[0], ..., b[k-1] < 0 <= b[k], ..., b[i-1] }}
while (i != n) {
  if (b[i] < 0) {
    {{ b[0], ..., b[k-1] < 0 <= b[k], ..., b[i-1] and b[i] < 0 }}
    swap b[i], b[k];
    k = k + 1;
    {{ b[0], ..., b[k-1] < 0 <= b[k], ..., b[i] }}
  }
  i = i + 1;
}
{{ b[0], ..., b[k-1] < 0 <= b[k], ..., b[n-1] }}
```

- **`I`** holds initially
- **`I`** and i = n implies postcondition
- **`I`** is maintained by loop body?

Example: partition array

Consider the following code to put the negative values at the beginning of array `b`:

```
{{ 0 <= n <= b.length }}
i = k = 0;
{{ Inv: b[0], ..., b[k-1] < 0 <= b[k], ..., b[i-1] }}
while (i != n) {
  if (b[i] < 0) {
    {{ b[0], ..., b[k-1] < 0 <= b[k], ..., b[i-1] and b[i] < 0 }}
    swap b[i], b[k];
    {{ b[0], ..., b[k] < 0 <= b[k+1], ..., b[i] }}
    k = k + 1;
  }
  i = i + 1;
}
{{ b[0], ..., b[k-1] < 0 <= b[k], ..., b[n-1] }}
```

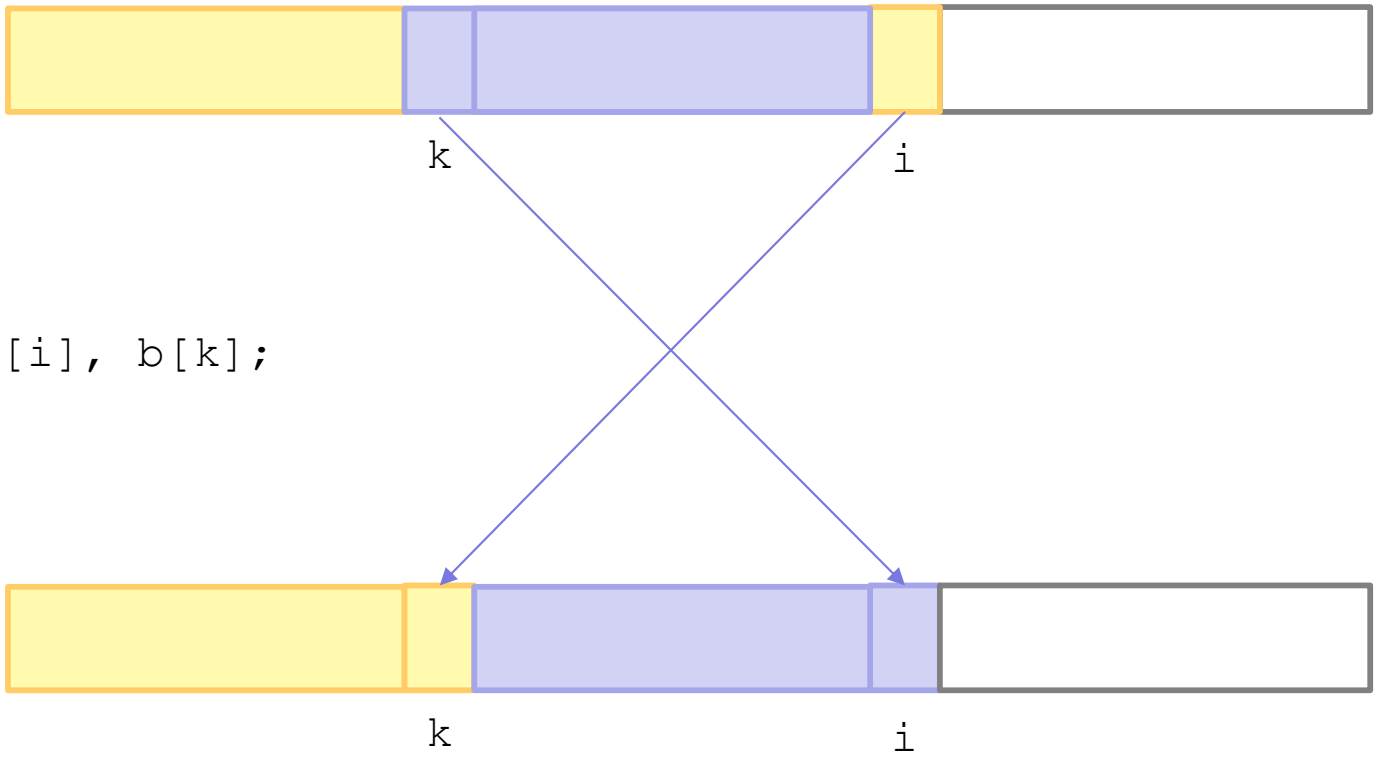
- **`I`** holds initially
- **`I`** and i = n implies postcondition
- **`I`** is maintained by loop body?

This is a valid triple.
(Takes some thought.)

Example: partition array

{{ b[0], ..., b[k-1] < 0 <= b[k], ..., b[i-1] and b[i] < 0 }}

yellow is < 0
purple is >= 0



{{ b[0], ..., b[k] < 0 <= b[k+1], ..., b[i] }}

Example: partition array

Consider the following code to put the negative values at the beginning of array `b`:

```
{{ 0 <= n <= b.length }}
i = k = 0;
{{ Inv: b[0], ..., b[k-1] < 0 <= b[k], ..., b[i-1] }}
while (i != n) {
  if (b[i] < 0) {
    swap b[i], b[k];
    k = k + 1;
  }
  i = i + 1;
}
{{ b[0], ..., b[k-1] < 0 <= b[k], ..., b[n-1] }}
```

- **`I`** holds initially
- **`I`** and i = n implies postcondition
- **`I`** is maintained by loop body