Recitation 8

Tyler Chan

March 11, 2025

Agenda

- Creating Specification
- Specification Strength
- Logical Formulas from Specifications
- Quiz 5

Example 1: Specification

Create a specification for the following code:

```
public int max(int[] array) {
    int max = array[0];
    for (int x : array) {
        if (x > max) {
            max = x;
        }
    }
    return max;
}
```

Example 1: Specification

Create a specification for the following code:

```
public int max(int[] array) {
    int max = array[0];
    for (int x : array) {
        if (x > max) {
            max = x;
        }
    }
    return max;
}
```

- Requires:
- Modifies:
- Effects:
- Returns:
- Throws:

Example 1: Specification

Create a specification for the following code:

```
public int max(int[] array) {
    int max = array[0];
    for (int x : array) {
        if (x > max) {
            max = x;
        }
    }
    return max;
}
```

- Requires:
  - `array != null`
  - `array.length > 0`
- Modifies:
- Effects:
- Returns:
- Throws:

Example 1: Specification

Create a specification for the following code:

```
public int max(int[] array) {
    int max = array[0];
    for (int x : array) {
        if (x > max) {
            max = x;
        }
    }
    return max;
}
```

- Requires:
  - `array != null`
  - `array.length > 0`
- Modifies: None.
- Effects:
- Returns:
- Throws:

Example 1: Specification

Create a specification for the following code:

```
public int max(int[] array) {
    int max = array[0];
    for (int x : array) {
        if (x > max) {
            max = x;
        }
    }
    return max;
}
```

- Requires:
  - `array != null`
  - `array.length > 0`
- Modifies: None.
- Effects: None.
- Returns:
- Throws:

Example 1: Specification

Create a specification for the following code:

```
public int max(int[] array) {
    int max = array[0];
    for (int x : array) {
        if (x > max) {
            max = x;
        }
    }
    return max;
}
```

- Requires:
  - `array != null`
  - `array.length > 0`
- Modifies: None.
- Effects: None.
- Returns: The maximum element in the array.
- Throws:

Example 1: Specification

Create a specification for the following code:

```
public int max(int[] array) {
    int max = array[0];
    for (int x : array) {
        if (x > max) {
            max = x;
        }
    }
    return max;
}
```

- Requires:
  - `array != null`
  - `array.length > 0`
- Modifies: None.
- Effects: None.
- Returns: The maximum element in the array.
- Throws: Undefined behavior.

Example 2: Specification

Create a specification for the following code:

```
public void reverse(int[] array) {
    int n = array.length;
    for (int i = 0; i < n / 2; i++) {
        int temp = array[i];
        array[i] = array[n - i - 1];
        array[n - i - 1] = temp;
    }
}
```

Example 2: Specification

Create a specification for the following code:

```
public void reverse(int[] array) {
    int n = array.length;
    for (int i = 0; i < n / 2; i++) {
        int temp = array[i];
        array[i] = array[n - i - 1];
        array[n - i - 1] = temp;
    }
}
```

- Requires:
- Modifies:
- Effects:
- Returns:
- Throws:

Example 2: Specification

Create a specification for the following code:

```
public void reverse(int[] array) {
    int n = array.length;
    for (int i = 0; i < n / 2; i++) {
        int temp = array[i];
        array[i] = array[n - i - 1];
        array[n - i - 1] = temp;
    }
}
```

- Requires: `array != null`
- Modifies:
- Effects:
- Returns:
- Throws:

Example 2: Specification

Create a specification for the following code:

```
public void reverse(int[] array) {
    int n = array.length;
    for (int i = 0; i < n / 2; i++) {
        int temp = array[i];
        array[i] = array[n - i - 1];
        array[n - i - 1] = temp;
    }
}
```

- Requires: `array != null`
- Modifies: array
- Effects:
- Returns:
- Throws:

Example 2: Specification

Create a specification for the following code:

```
public void reverse(int[] array) {
    int n = array.length;
    for (int i = 0; i < n / 2; i++) {
        int temp = array[i];
        array[i] = array[n - i - 1];
        array[n - i - 1] = temp;
    }
}
```

- Requires: `array != null`
- Modifies: array
- Effects: Reverses the order of the elements in the array in place.
- Returns:
- Throws:

Example 2: Specification

Create a specification for the following code:

```
public void reverse(int[] array) {
    int n = array.length;
    for (int i = 0; i < n / 2; i++) {
        int temp = array[i];
        array[i] = array[n - i - 1];
        array[n - i - 1] = temp;
    }
}
```

- Requires: `array != null`
- Modifies: array
- Effects: Reverses the order of the elements in the array in place.
- Returns: None.
- Throws:

Example 2: Specification

Create a specification for the following code:

```
public void reverse(int[] array) {
    int n = array.length;
    for (int i = 0; i < n / 2; i++) {
        int temp = array[i];
        array[i] = array[n - i - 1];
        array[n - i - 1] = temp;
    }
}
```

- Requires: `array != null`
- Modifies: array
- Effects: Reverses the order of the elements in the array in place.
- Returns: None.
- Throws: None.

Example 3: Specification Comparision

Which of the following specifications is **stronger**?

```
public int factorial(int n) {
    ...
}
```

Specification A:

- Requires: `n >= 0`
- Returns: The factorial of n, but if `n > 12` then return INT_MAX

Specification B:

- Requires:
  - `n >= 0`
  - `n <= 12` (to ensure the result fits the 32-bit limit)
- Returns: The factorial of n

Solution to Example 3

To show that A is stronger than B we must show either:

1. $[P_B \to (Q_B \vee P_A)] \wedge [(P_B \wedge Q_A) \to Q_B]$ or
2. $(P_B \to P_A) \wedge (Q_A \to Q_B)$

1. is from $(P_A \to Q_A) \to (P_B \to Q_B)$ and 2. is simpler but stricter.

Solution to Example 3

We will use $(P_B \to P_A) \wedge (Q_A \to Q_B)$:

Solution to Example 3

We will use $(P_B \to P_A) \wedge (Q_A \to Q_B)$:

$$P_B \to P_A$$

Solution to Example 3

We will use $(P_B \to P_A) \land (Q_A \to Q_B)$:

$$P_B \to P_A$$

$$(n \geq 0) \land (n \leq 12) \to (n \geq 0)$$

Solution to Example 3

We will use $(P_B \to P_A) \wedge (Q_A \to Q_B)$:

$$P_B \to P_A$$

$$(n \geq 0) \wedge (n \leq 12) \to (n \geq 0)$$

$$True$$

Solution to Example 3

We will use $(P_B \to P_A) \wedge (Q_A \to Q_B)$:

$$P_B \to P_A$$

$$(n \geq 0) \wedge (n \leq 12) \to (n \geq 0)$$

$$True$$

Solution to Example 3

We will use $(P_B \to P_A) \land (Q_A \to Q_B)$:

$$P_B \to P_A$$

$$(n \geq 0) \land (n \leq 12) \to (n \geq 0)$$

$$True$$

$$Q_A \to Q_B$$

Solution to Example 3

We will use $(P_B \to P_A) \land (Q_A \to Q_B)$:

$$P_B \to P_A$$

$$(n \geq 0) \land (n \leq 12) \to (n \geq 0)$$

$$True$$

$$Q_A \to Q_B$$

$$(n! \lor INT\_MAX) \to (n!)$$

Solution to Example 3

We will use $(P_B \rightarrow P_A) \wedge (Q_A \rightarrow Q_B)$:

$$P_B \rightarrow P_A$$

$$(n \geq 0) \wedge (n \leq 12) \rightarrow (n \geq 0)$$

$$True$$

$$Q_A \rightarrow Q_B$$

$$(n! \vee INT\_MAX) \rightarrow (n!)$$

Since Q_B is under the condition $n \leq 12$, the branch where INT_MAX is returned is not taken:

Solution to Example 3

We will use $(P_B \to P_A) \wedge (Q_A \to Q_B)$:

$$P_B \to P_A$$

$$(n \geq 0) \wedge (n \leq 12) \to (n \geq 0)$$

$$True$$

$$Q_A \to Q_B$$

$$(n! \vee INT\_MAX) \to (n!)$$

Since Q_B is under the condition $n \leq 12$, the branch where INT_MAX is returned is not taken:

$$True$$

Example 4: Specification to Logical Form

Convert the following specification for the following code to a logical formula in the form $P \rightarrow (E \wedge M)$:

```
public int remove(Integer[] array, int n) {
    ...
}
```

- Requires: (`array != null`) and (`0 <= n < array.length`)
- Modifies: `array`
- Effects: `Removes the element at index n from the array and shifts the remaining elements to the left.`
- Returns: `The removed element.`
- Throws: `None.`

Solution to Example 4

```
E = returns array[n] &&
    for i >= n:
        array_post[i] = array_pre[i+1];

M = for i < n:
        array_post[i] = array_pre[i];
```

$$(array != null) \land (0 \geq n < array.length) \rightarrow (E \land M)$$

Quiz 5

Do quiz 5 now on Submitty.