COM4506/6506: Testing and Verification in Safety Critical Systems

Dr Ramsay Taylor

Contents

- Detailed Function Specs
- Code conformance to specs - Refinement again
- Coding language checks

Detailed Specifications

Our iterative design process produces steadily more specific specifications…

Eventually these get down to detail specs of individual operations or functions in the final software.

We can do various ``upward'' verification on the rest of the system, *assuming these functions work correctly.*

$$\begin{array}{|l} \textit{ConvertFtoC} \\ \hline tempF? : FLOAT32 \\ tempC! : FLOAT32 \\ \hline tempC! = (tempF? - 32) \times \frac{5}{9} \\ \hline \end{array}$$

“Assuming these functions work correctly”

```
x = tempF - 32;
y = 5.0 / 9.0;
tempC = x * y;
```

=?=

$$\begin{array}{|l} \textit{ConvertFtoC} \\ \hline tempF? : FLOAT32 \\ tempC! : FLOAT32 \\ \hline tempC! = (tempF? - 32) \times \frac{5}{9} \\ \hline \end{array}$$

Hoare Logic

$$\{x = 42\}y := x/2;\ \{x = 42;\ y = (x/2)\}$$

Precondition
Anterior State
“Before” state

Pseudocode

Postcondition
Posterior State
“After” state

A “Hoare Triple”

Hoare Logic

$$\{x = 42\}y := x/2;\ \{x = 42;\ y = (x/2)\}$$

$$\{x = 42\}y := x/2;\ \{x = 42;\ y = 21\}$$

Hoare Logic

$$\{x = 42\}y := x/2;\ x := y + 4;\ \{???\}$$

Sequential
Composition

Hoare Logic

$$\{x = 42\}y := x/2;\ x := y + 4;\ \{???\}$$

Sequential
Composition

```
x = tempF - 32;
y = 5.0 / 9.0;
tempC = x * y;
```

Hoare Logic

$$\{x = 42\}y := x/2;\ x := y + 4;\ \{???\}$$

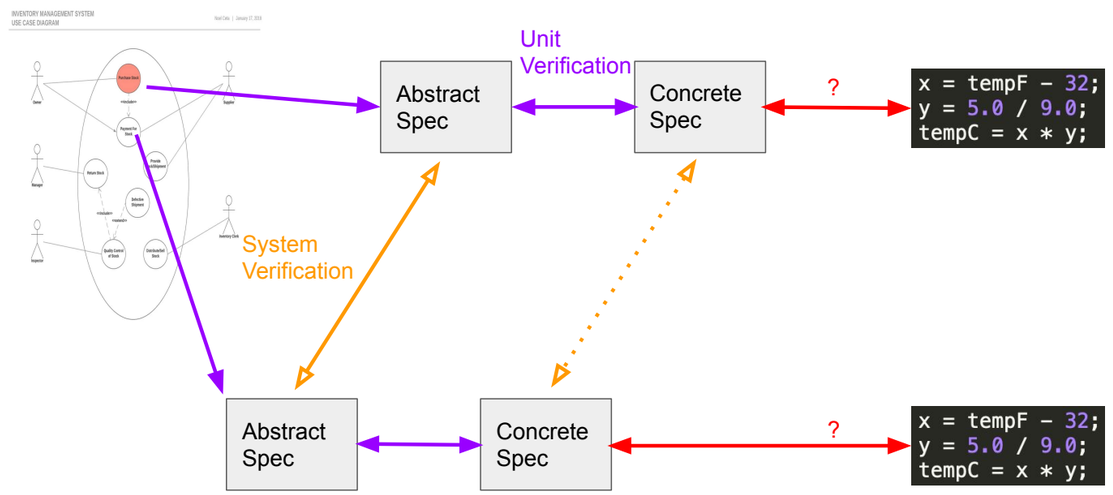
$$\{x = 42\}y := x/2;\ \{x = 42;\ y = 21\}x := y + 4;\ \{x = 25;\ y = 21\}$$

A Hoare Triple

Another Hoare Triple!

Shared intermediate state

Refinement Calculus



Refinement Calculus

ConvertFtoC

$tempF? : FLOAT32$

$tempC! : FLOAT32$

$tempC! = (tempF? - 32) \times \frac{5}{9}$

$\{tempF = TF\}$ //*FIXME Code goes here...*$\{tempF = TF;\ tempC = (TF - 32) \times \frac{5}{9}\}$

Refinement Calculus

$\{tempF = TF\}$ $x = tempF - 32;$ $\{tempF = TF;\ x = TF - 32\}$

$y = 5.0/9.0;$ $\{tempF = TF;\ x = TF - 32;\ y = \frac{5}{9}\}$

$tempC = x * y;$ $\{tempF = TF;\ x = TF - 32;\ y = \frac{5}{9};\ tempC = (TF - 32) \times \frac{5}{9}\}$

Refinement Calculus

$$\{tempF = TF\} \quad x = tempF - 32; \quad \{tempF = TF;\ x = TF - 32\}$$

$$y = 5.0/9.0; \quad \{tempF = TF;\ x = TF - 32;\ y = \tfrac{5}{9}\}$$

$$tempC = x * y; \quad \{tempF = TF;\ x = TF - 32;\ y = \tfrac{5}{9};\ tempC = (TF - 32) \times \tfrac{5}{9}\}$$

$$\{tempF = TF\}//FIXME\ Code\ goes\ here...\{tempF = TF;\ tempC = (TF - 32) \times \tfrac{5}{9}\}$$

Refinement Calculus

“Can’t I just autocode this?”

Sadly, it’s not “algorithmic” - you can’t *compute* the correct line of code to write, you can only prove that the one(s) you wrote are valid.

Which of these is “right”?

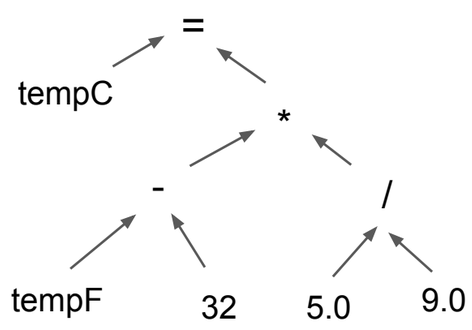
```
x = tempF - 32;
y = 5.0 / 9.0;
tempC = x * y;
```

```
y = 5.0 / 9.0;
x = tempF - 32;
tempC = x * y;
```

```
tempC = (tempF - 32) * 5.0/9.0
```

Language Assumptions

```
tempC = (tempF - 32) * 5.0/9.0
```



Language Assumptions

```
t = x++ / (42 - x--) * y;
```

????

Language Assumptions

What operation is this?

```
x = tempF - 32;
y = 5.0 / 9.0;
tempC = x * y;
```

What number is this?

01000010000000000000000000000000

Or

00000000000000000000000000100000

Does it matter?

Language Assumptions

```
>>> tf = 42.2
>>> tf - 32
10.200000000000003
```

Summary

- After much iterative design, we have some very specific Specifications - we then need to be sure that the code meets them.
- We can continue the formal refinement processes into code.
- This is a laborious, human process
- We have to be careful about code assumptions