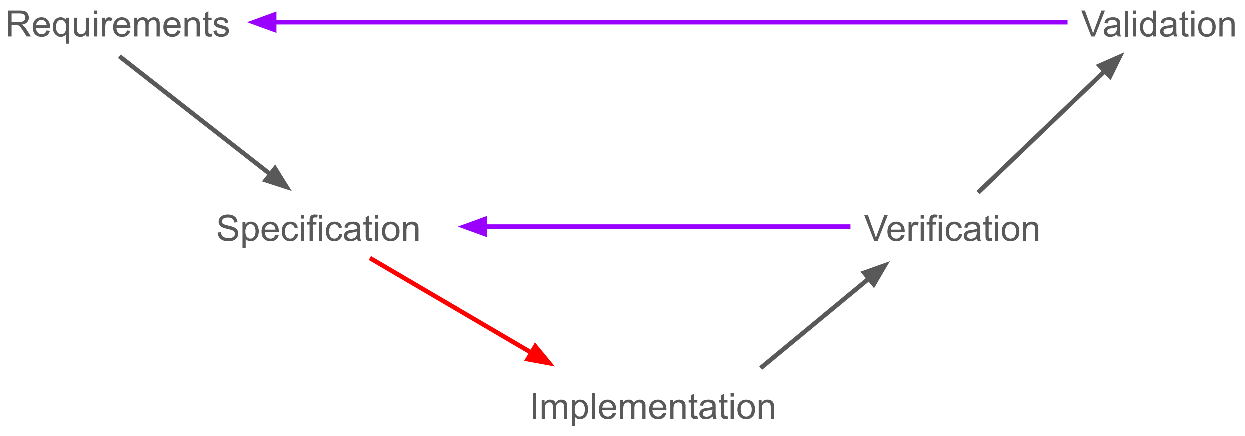
COM4506/6506: Testing and Verification in Safety Critical Systems

Dr Ramsay Taylor

Contents

- Retaining our formal specs in the code
- Compile-time checks
- Taking this to extremes - SPARK Ada

Correct by Construction



Requirements

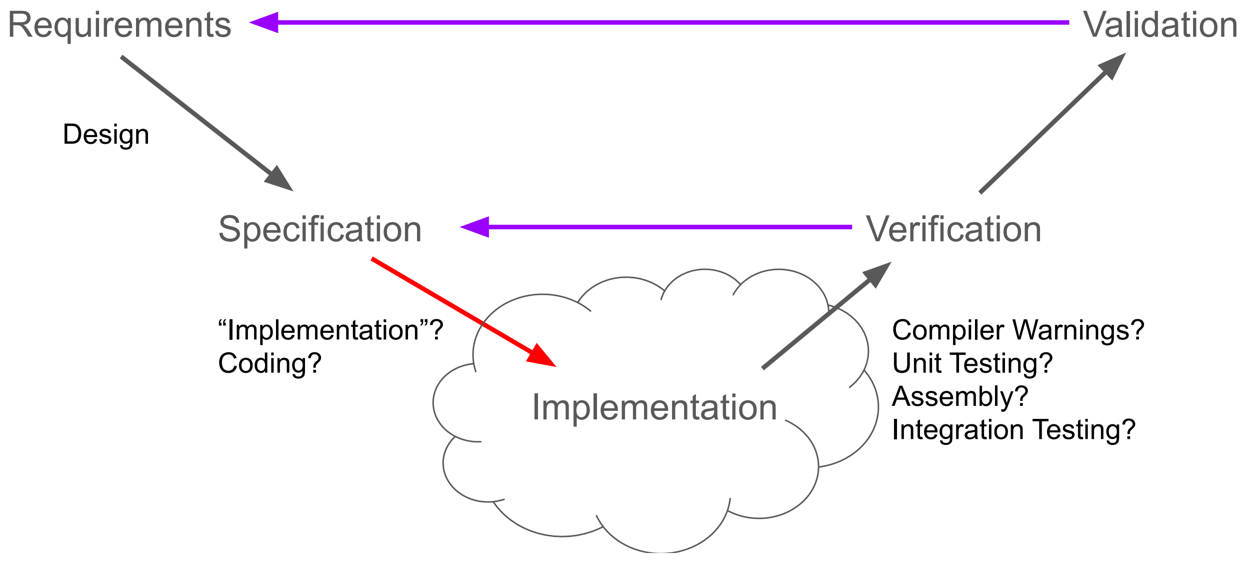
Validation

Specification

Verification

Implementation

Correct by Construction



Requirements

Validation

Design

Specification

Verification

“Implementation”?
Coding?

Implementation

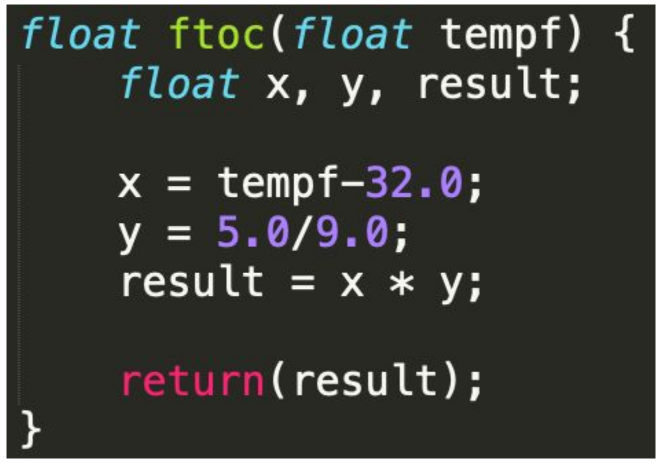
Compiler Warnings?
Unit Testing?
Assembly?
Integration Testing?

Compiler Checking

ConvertFtoC

$tempF? : FLOAT32$
$tempC! : FLOAT32$

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



```
float ftoc(float tempf) {
    float x, y, result;

    x = tempf-32.0;
    y = 5.0/9.0;
    result = x * y;

    return(result);
}
```

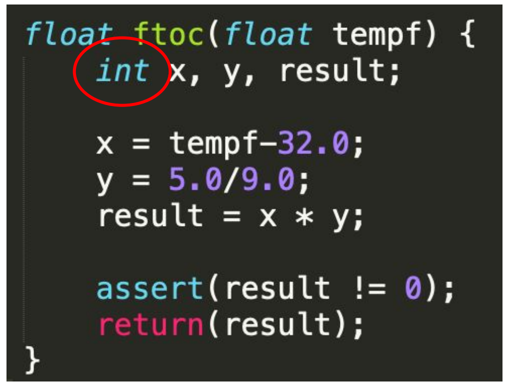
Code Assertions

```
private void setRefreshInterval(int interval) {
 // Confirm adherence to precondition in nonpublic method
 assert interval > 0 && interval <= 1000/MAX_REFRESH_RATE : interval;

 ... // Set the refresh interval
}
```

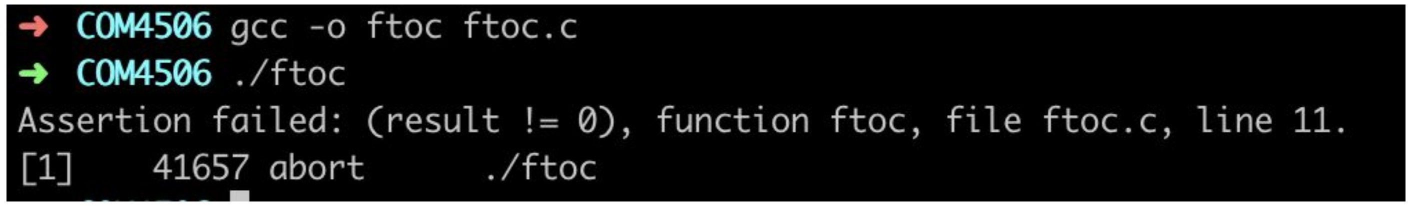
Code Assertions

```
public BigInteger modInverse(BigInteger m) {
  if (m.signum <= 0)
    throw new ArithmeticException("Modulus not positive: " + m);
  ... // Do the computation
  assert this.multiply(result).mod(m).equals(ONE) : this;
  return result;
}
```

Code Assertions

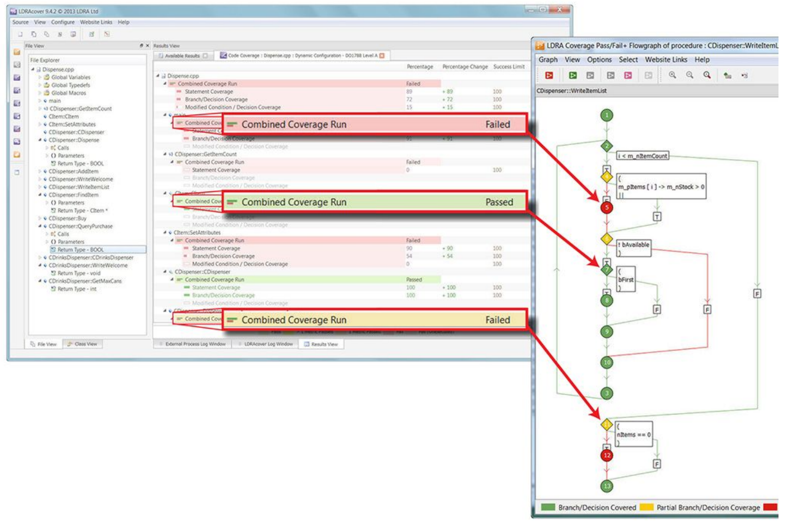


```
float ftoc(float tempf) {
   int x, y, result;

   x = tempf-32.0;
   y = 5.0/9.0;
   result = x * y;

   assert(result != 0);
   return(result);
}
```



```
➜ COM4506 gcc -o ftoc ftoc.c
➜ COM4506 ./ftoc
Assertion failed: (result != 0), function ftoc, file ftoc.c, line 11.
[1]    41657 abort      ./ftoc
```

Static Analysis



Static Analysis (i.e. checking the code without running it) is like compiler checks but taken far further.

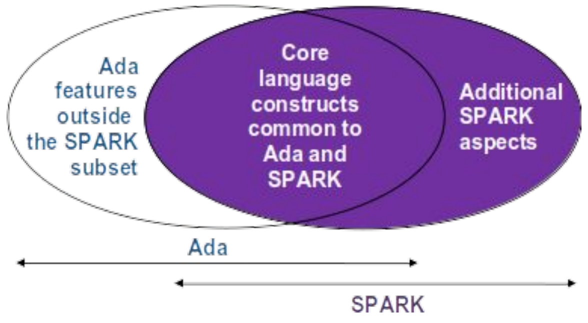
We will discuss that more as we work back up the far side of the V model, but one language takes it further...

SPARK Ada

Ada was already a strongly typed, structured language used in aviation and military systems.

SPARK Ada expands on that:

- Removes some ambiguous elements
- Adds various *specification* elements
- Includes a *toolchain* of analysis tools.



Ada features outside the SPARK subset

Core language constructs common to Ada and SPARK

Additional SPARK aspects

Ada

SPARK

SPARK Ada

```
procedure Increment
  (X : in out Integer)
is
begin
  X := X + 1;
end Increment;
```

SPARK Ada

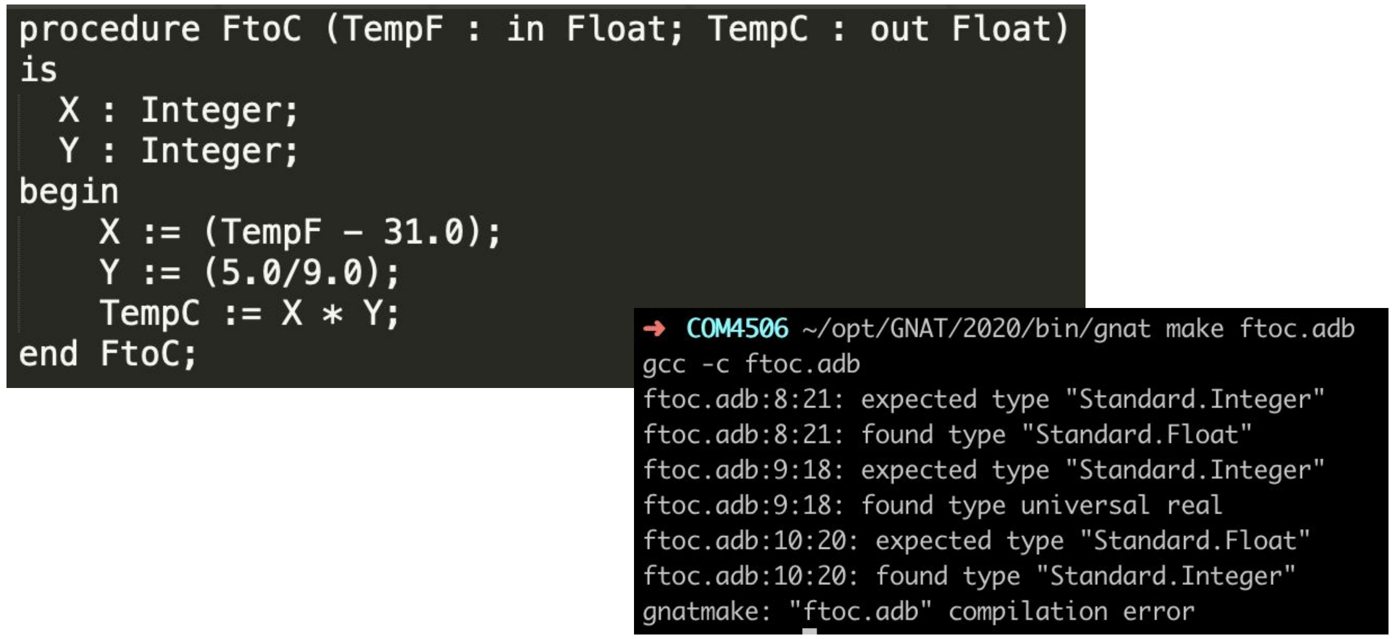
Types

Structure

```
procedure Increment
  (X : in out Integer)
is
begin
  X := X + 1;
end Increment;
```

Data-Flow Checking

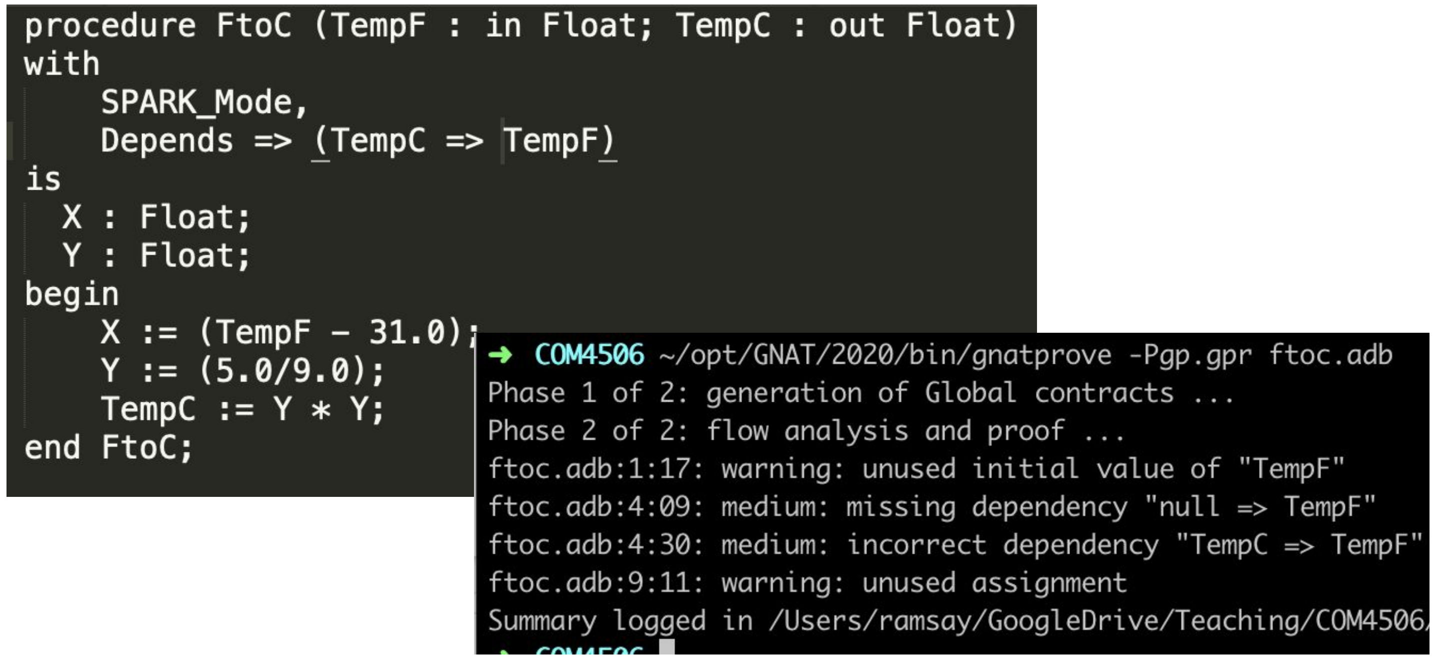
SPARK Ada



```
procedure FtoC (TempF : in Float; TempC : out Float)
is
  X : Integer;
  Y : Integer;
begin
    X := (TempF - 31.0);
    Y := (5.0/9.0);
    TempC := X * Y;
end FtoC;
```

```
➜ COM4506 ~/opt/GNAT/2020/bin/gnat make ftoc.adb
gcc -c ftoc.adb
ftoc.adb:8:21: expected type "Standard.Integer"
ftoc.adb:8:21: found type "Standard.Float"
ftoc.adb:9:18: expected type "Standard.Integer"
ftoc.adb:9:18: found type universal real
ftoc.adb:10:20: expected type "Standard.Float"
ftoc.adb:10:20: found type "Standard.Integer"
gnatmake: "ftoc.adb" compilation error
```

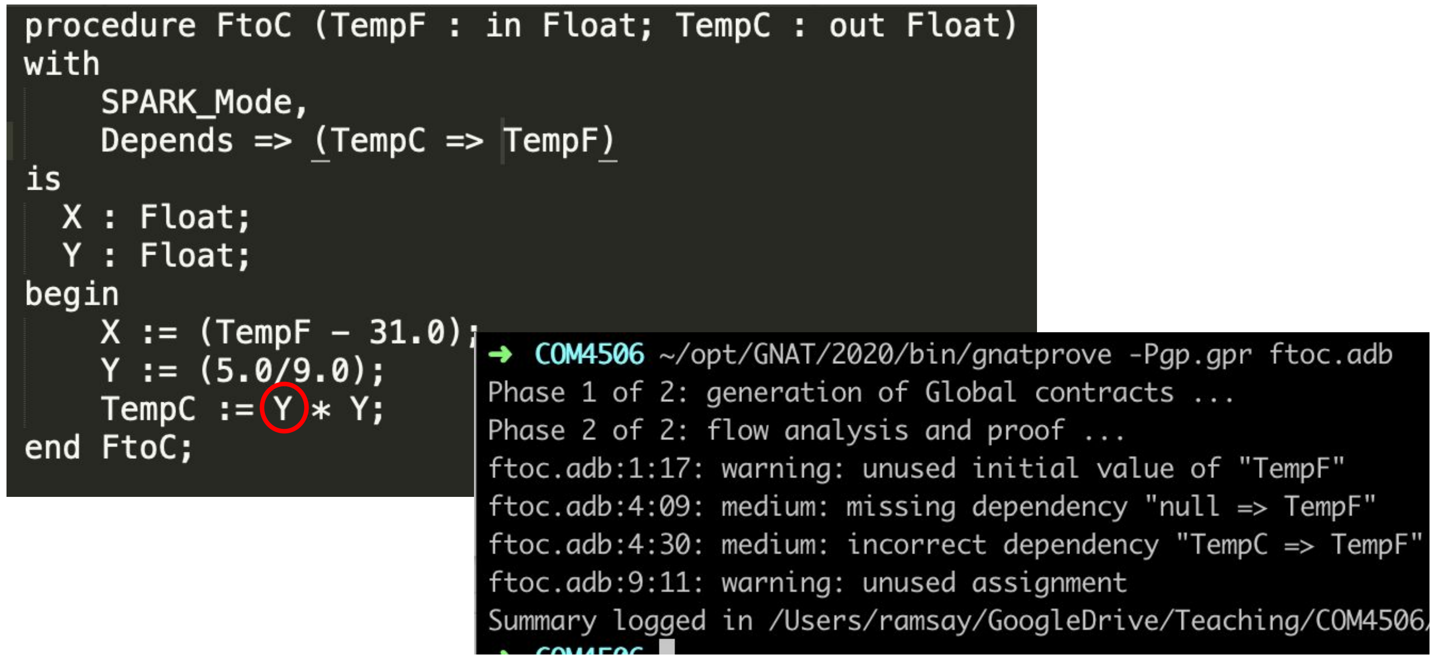
SPARK Ada



```
procedure FtoC (TempF : in Float; TempC : out Float)
with
    SPARK_Mode,
    Depends => (TempC => TempF)
is
  X : Float;
  Y : Float;
begin
    X := (TempF - 31.0);
    Y := (5.0/9.0);
    TempC := Y * Y;
end FtoC;
```

```
➜ COM4506 ~/opt/GNAT/2020/bin/gnatprove -Pgp.gpr ftoc.adb
Phase 1 of 2: generation of Global contracts ...
Phase 2 of 2: flow analysis and proof ...
ftoc.adb:1:17: warning: unused initial value of "TempF"
ftoc.adb:4:09: medium: missing dependency "null => TempF"
ftoc.adb:4:30: medium: incorrect dependency "TempC => TempF"
ftoc.adb:9:11: warning: unused assignment
Summary logged in /Users/ramsay/GoogleDrive/Teaching/COM4506/
```

SPARK Ada



```
procedure FtoC (TempF : in Float; TempC : out Float)
with
    SPARK_Mode,
    Depends => (TempC => TempF)
is
  X : Float;
  Y : Float;
begin
    X := (TempF - 31.0);
    Y := (5.0/9.0);
    TempC := Y * Y;
end FtoC;
```

```
➜ COM4506 ~/opt/GNAT/2020/bin/gnatprove -Pgp.gpr ftoc.adb
Phase 1 of 2: generation of Global contracts ...
Phase 2 of 2: flow analysis and proof ...
ftoc.adb:1:17: warning: unused initial value of "TempF"
ftoc.adb:4:09: medium: missing dependency "null => TempF"
ftoc.adb:4:30: medium: incorrect dependency "TempC => TempF"
ftoc.adb:9:11: warning: unused assignment
Summary logged in /Users/ramsay/GoogleDrive/Teaching/COM4506/
```

Summary

- Structured programming languages already help to retain some of our specification.
- Language Assertions can check some things, but they are often run-time only - so it might be too late!
- Static Analysis is “compiler checking” taken a lot further.
- SPARK Ada takes this to the extremes, but it tries to create systems that are *Correct by Construction*