Componolit
Secure Systems Engineering

Program verification with SPARK

Johannes Kliemann, Componolit GmbH
Dresden, 22.09.2018

About

- **Me**
  - Informationssystemtechnik at TU Dresden
  - Working since 2017 at Componolit
  - OS/driver development with Ada, SPARK, C/C++

- **Componolit**
  - Secure component based systems
  - Mobile devices
  - Industrial security
  - Free software
  - Open development

Ada

- ISO/8652:1987, 1995, 2012, (2020)
- Object oriented
- Extensive compile time and runtime checks
- Strong type system
- Contract based programming
- Easy integration with C-FFI

```
1 with Ada.Text_Io;
2 use Ada.Text_Io;
3
4 procedure Main is
5 begin
6    Put_Line(„Hello World“);
7 end Main;
```

Contract based programming

Componolit
Secure Systems Engineering

```
1 #ifndef _STDLIB_H_
2 #define _STDLIB_H_
3
4 /* returns the absolute value */
5 int abs(int j);
6
7 #endif
```

Contract based programming

Componolit
Secure Systems Engineering

```
1 package Std is
2
3 function Abs_Value (J : Integer) return Integer
4    with
5       Post => Abs_Value‘Result >= 0;
6
7 end Std;
```

Example: Abs

```
1 int abs(int j)
2 {
3     if(j > 0)
4         return j
5     else
6         return -j
7 }
```

```
abs(-1) => 1

abs(1) => 1

abs(0) => 0

abs(-2147483648) => -2147483648
```

Example: Abs

```
1 function Abs_Value(J : Integer) return Integer is
2 begin
3    if J > 0 then
4       return J;
5    else
6       return -J;
7    end if;
8 end Abs_Value;
```

```
Abs_Value(-2147483648) => raised CONSTRAINT_ERROR : overflow check failed
```

What about testing?

Componolit
Secure Systems Engineering

- **Still important**
  - Integration
  - Validating the specification
- **But**
  - Exhaustive testing is impossible
  - State explosion for non-trivial software
  - Testing achieves 1 defect every 1000 lines at BEST
  - Formal methods can be 50-100 times better than that

SPARK

Componolit
Secure Systems Engineering

- Subset of Ada
- Static analysis
- Formal verification of contracts
- Data flow and dependency checking
- Variable depth of verification

- **Valid SPARK**
  - No access types
  - Functions without side effects
- **Absence of runtime errors**
  - No overflows, exceptions, etc.
- **Full functional proof**
  - Proof of semantic properties of a function

Verification Conditions

Componolit
Secure Systems Engineering

- Code

```
1 C := A / B;
```

- Verification Condition

```
1 B /= 0
```

Example: Abs

Componolit
Secure Systems Engineering

```
1 function Abs_Value (J : Integer) return Integer is
2 begin
3    if J > 0 then
4       return J;
5    else
6       return -J;
7    end if;
8 end Abs_Value;
```

```
medium: overflow check might fail (e.g. when Abs_Value'Result = 0 and J = -2147483648)
```

Example: Abs

Componolit
Secure Systems Engineering

```
1 function Abs_Value (J : Integer) return Integer
2    with
3       Pre => J /= Integer'First,
4       Post => Abs_Value'Result >= 0;
```

```
info: overflow check proved
info: postcondition proved
```

Example: Abs

```
1 procedure Do_Something is
2    Value : Integer;
3 begin
4    …
5    Value := Abs_Value (Value);
6    …
7 end Do_Something;
```

```
medium: precondition might fail (e.g. when Value = -2147483648)
```

Example: Abs

```
1 begin
2    …
3    if Value /= Integer‘First then
4       Value := Abs_Value (Value);
5    end if;
6    …
7 end Do_Something;
```

```
info: precondition proved
```

Example: Abs

Componolit
Secure Systems Engineering

```
1 type Symmetric_Int is new Integer -2**31 + 1 .. 2**31 - 1;
2 --  regular Integer goes from -2**31 to 2**31 - 1
3
4 function Abs_Value (J : Symmetric_Int) return Symmetric_Int
5    with
6       Post => Abs_Value‘Result >= 0;
```

Example: Abs

Componolit
Secure Systems Engineering

```
1 procedure Do_Something is
2    Value : Symmetric_Int;
3 begin
4    …
5    Value := Abs_Value (Value);
6    …
7 end Do_Something;
```

Where is it used?

Componolit
Secure Systems Engineering

- **Safety**
  - Rolls-Royce Trent Jet Engines
  - EuroFigther
  - Air traffic management, rail signaling
  - Space Flight (e.g. CubeSat project)
- **Security**
  - Muen microkernel
  - WooKey, a secure and trustworthy USB mass storage device

Example: Converting a string from C to Ada

Componolit
Secure Systems Engineering

- **Strings in C**
  - Pointer to a sequence of char
  - Null terminated
  - No explicit length attribute
- **Strings in Ada**
  - Array of Character
  - Explicit size

- Calling Ada from C
- Passing a C string and using it in Ada
- Proofs:
  - Absence of runtime errors
  - An empty string or null pointer yields a preset default string

Questions?

Componolit
Secure Systems Engineering

- **Johannes Kliemann**
  - jk@jkliemann.de
  - github.com/jklmnn
  - https://gitlab.com/jklmnn/sparkds2018
- **Componolit**
  - https://componolit.com/
  - @Componolit
  - github.com/Componolit
- **AdaCore**
  - https://learn.adacore.com/
  - GNAT GPL Toolchain https://www.adacore.com/download

Componolit
Secure Systems Engineering