

C Program Specification and Verification with ACSL and Frama-C/WP

VerifyThis Tutorial

Virgile Prevosto
virgile.prevosto@cea.fr

CEA Tech List

2019-04-06



- ▶ short general introduction to Frama-C and ACSL
- ▶ examples of writing ACSL specifications
- ▶ verification of implementations with the WP plugin of Frama-C
- ▶ All material available on [Frama-C github](#):

<https://frama.link/fc-tuto-2019-04>




- ▶ **Examples** contains plain (unannotated) C code
 - ▶ **Solutions** contains the corresponding annotations...
 - ▶ ... that should be provable by Frama-C 18.0, Alt-Ergo and Coq
- ▶ `vagrant init virgile/VerifyThis-fc-tutorial`
(very experimental)

It's 2019! Why bother with proving C programs?

- ✗ Lack of modularity and high-level constructs
- ✗ Many quirks in the semantics (pointers)
- ✗ Small standard library
- ✓ Lot of legacy code
- ✓ Embedded world (aka IoT) still uses it in many places
- ✓ And in some cases they care about safety and (cyber)security

A few recent use cases

- ▶ **S2OPC** OPC (communication protocol for industrial systems),
result of INGOPCS French project
- ▶ **Bureau Veritas Cybersecurity Guidelines**
- ▶ **Vessedia H2020 project** 
including verification of parts of **Contiki OS**

- ▶ A Framework for modular analysis of C code.
- ▶ <http://frama-c.com/>
- ▶ Developed at CEA Tech List and Inria
- ▶ Released under LGPL license (v18.0 Argon in November 2018)
- ▶ Kernel based on CIL (Necula et al. – Berkeley).
- ▶ ACSL annotation language.
- ▶ Extensible platform
 - ▶ Collaboration of analyses over same code
 - ▶ Inter plug-in communication through ACSL formulas.
 - ▶ Adding specialized plug-in is easy



What is verified by Frama-C?

Code Properties

- ▶ Functional properties (contract)
- ▶ Absence of run-time error
- ▶ Termination
- ▶ Dependencies
- ▶ Non-interference
- ▶ Temporal properties

Perimeter of the verification

- ▶ Which part of the code is under analysis?
- ▶ Which initial context?

Trusted Code Base

- ▶ ACSL Axioms
- ▶ Hypotheses made by analyzers
 - ▶ WP memory model
- ▶ Stub Functions
- ▶ Frama-C itself

ANSI/ISO C Specification Language

Presentation

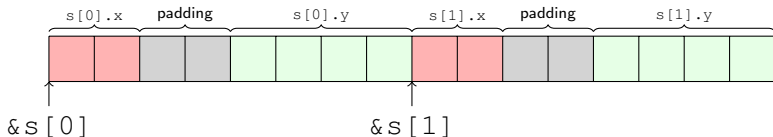
- ▶ Based on the notion of contract, like in Eiffel
- ▶ Allows users to specify functional properties of their code
- ▶ Allows communication between various plugins
- ▶ Independent from a particular analysis
- ▶ <https://github.com/acsl-language/acsl>

Basic Components

- ▶ First-order logic
- ▶ Pure C expressions
- ▶ C types + \mathbb{Z} (integer) and \mathbb{R} (real)
- ▶ Built-ins predicates and logic functions, particularly over pointers.

- ▶ All operations are done over \mathbb{Z} : **no overflow**
- ▶ ACSL predicate $\text{INT_MIN} \leq x + y \leq \text{INT_MAX}$
 \Leftrightarrow
 C operation $x+y$ does not overflow (undefined behavior)
- ▶ $(\text{int}) z \equiv z \bmod 2^{8*\text{sizeof}(\text{int})}$
- ▶ and $\text{INT_MIN} \leq (\text{int}) z \leq \text{INT_MAX}$

Memory description in ACSL



```

struct S {
short x;
int y;
} s[2];
  
```

```

\valid(&s[0]+(0 .. 1))
\valid((char*)&s[0] + (0 .. 15))
!\initialized(*((char*)&s[0].x+2))
\block_length(&s[0]) == 16
\base_addr(&s[0].y) == s
\offset(&s[1].y) == 12
\separated(&s[0], &s[1])
  
```