THALES

# Specification and Verification of High-level Properties with Frama-C and MetAcsl


Nikolai KOSMATOV

Thales Research & Technology, cortAIx Labs

Joint work with Adel DJOUDI, Martin HANA, Pascale LE GALL, Virgile PREVOSTO, Louis RILLING, Virgile ROBLES


Dagstuhl Seminar 25172, April 23-25, 2025

# Tool context: ACSL, Frama-C and its deductive verification plugin WP

**Frama-C** is a platform for analysis and verification of C programs

- **ACSL (ANSI C Specification Language)** supported by Frama-C

Software Analyzers

**WP plugin: Weakest Precondition** based tool for deductive verification

- **Proof** of **semantic properties** of the program
- **Modular** verification (function by function)
- **Input**: a program and its specification in ACSL
- WP generates verification conditions (VCs)
- Relies on **Why3** and **Automatic Theorem Provers** to discharge VCs
  - Alt-Ergo, Z3, CVC4, CVC5, …





THALES

# Example of a C program annotated in ACSL

```
/*@ requires n>=0 && \valid(t+(0..n-1));
    assigns \nothing;
    ensures \result != 0 <==>
      (\forall integer j; 0 <= j < n ==> t[j] == 0);
*/
int all_zeros(int t[], int n) {
  int k;
  /*@ loop invariant 0 <= k <= n;
      loop invariant \forall integer j; 0<=j<k ==> t[j]==0;
      loop assigns k;
      loop variant n-k;
  */
  for(k = 0; k < n; k++)
    if (t[k] != 0)
      return 0;
  return 1;
}
```

Can be proven with Frama-C/WP





THALES

# Outline



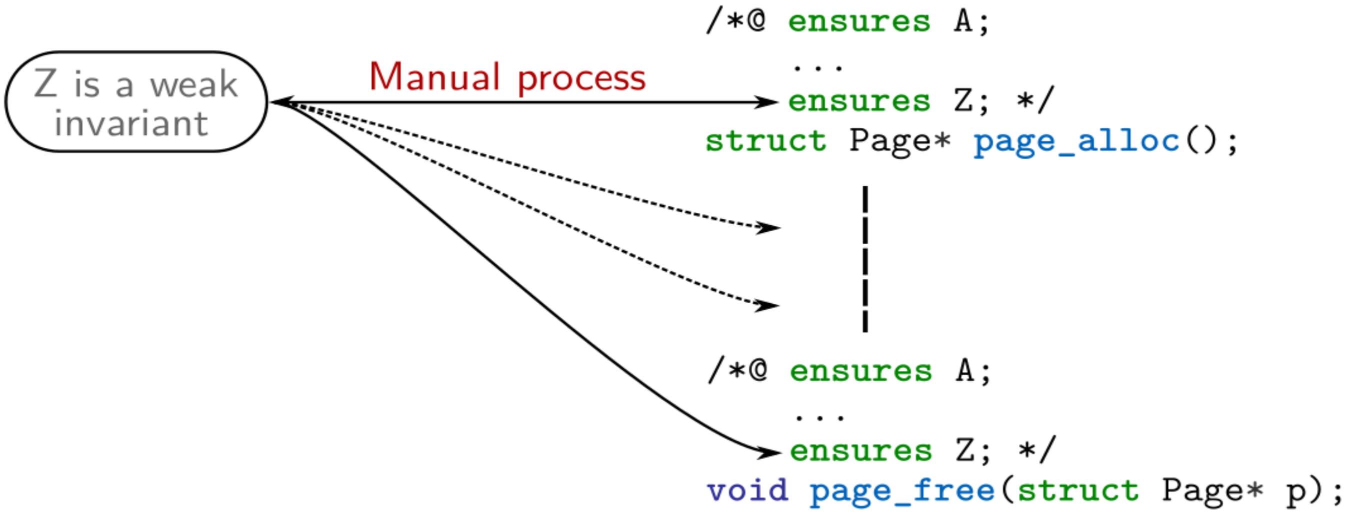
- Motivation: Specification and verification of global (security) properties
- High-Level ACSL Requirements (HILARE), or Metaproperties, and MetAcsl tool
- Examples of Proof with MetAcsl and WP
- Application to certification of JavaCard Virtual Machine
- Conclusion



THALES

# Motivation: Global (High-Level) properties are hard to specify and to maintain

Specifying global properties with contracts: manual and tedious. No explicit link between clauses.




Assessing if contracts form a global property is difficult, especially after an update.

# Examples of High-Level Properties

- A non-privileged user never reads a privileged (private) data page
- A privileged user never writes to a non-privileged (public) page
- The privilege level of a page cannot be changed unless...
- The privilege level of a user cannot be changed unless…
- A free page cannot be read or written, and must contain zeros
- Object data can be written only by the object owner
- Object data can be read only by the object owner

## Such properties can be expressed as

- Constraints on reading / writing operations, calls to some functions,
- Strong or weak invariants





THALES

# Solution: Metaproperties, or HILARE (High-Level ACSL Requirements)

We introduce meta-properties, which are a combination of:

- **A set of targets functions**, on which the property must hold.
  foo {foo, bar} \ALL \diff(\ALL,{foo, bar})
- **A context**, which characterizes the situation in which the property must hold.
  \strong_invariant \writing \reading
- **An ACSL predicate**, expressed over the set of global variables.
  A < B *p == 0 \separated(\written, p)

```
meta \prop,
    \name(A < B everywhere in foo and bar),
    \targets({foo, bar}),
    \context(\strong_invariant),
    A < B;
```





THALES

# Available Contexts



- **Strong invariant:** Everywhere in the function
- **Weak invariant:** Before and after the function
- **Upon writing:** Whenever the memory is modified. The predicate can use a special meta-variable `\written`, referencing the address(es) being written to at a particular point.

```
meta \prop, \name(X is only modified if null),
      \targets(\ALL), \context(\writing),
      !\separated(\written, &X) ⇒ X == 0;
```

- **Upon reading:** Similarly, when memory is read
- **Upon calling:** Similarly, when a function is called

```
meta \prop, \name(foo can only be called from bar),
      \targets(\diff(\ALL, bar)),
      \context(\calling), \called ≠ &foo;
```

# Example: Integrity Metaproperty Verified with MetAcsl – Writing context

## Resulting code after generating assertions with MetAcsl and proof with Frama-C/WP:

```
/*@ meta "A_unchanged_unless";
  */
/*@ requires A
    ensures
      (C ≥ 0
      (C < 0
    assigns A,
  */
void foo(voi
{
  if (C >= 0) {
    /*@ check A_unchanged_unless: _1: meta: C < 0 → \separated(&A, &A);
    A = C;
    /*@ check A_unchanged_unless: _2: meta: C < 0 → \separated(&B, &A);
    B = C;
  }
  return;
}
```

If all instances are proved, the metaproperty is true

Contrary to an assert, a check is not kept in the proof context and does not overload the proof

MetAcsl

## Initial C code:

test5.c

```
int A, B, C;
/*@
  meta \prop, \name(A_unchanged_unless),
    \targets(\ALL), \context(\writing),
    C < 0 ==> \separated(\written, &A);
*/
/*@
  requires A==B;
  assigns A,B;
  ensures C>=0 && A==C && B==C ||
    C<0 && A==\old(A) && B==\old(B); */
void foo(){
  if ( C >= 0 ){
    A = C;
    B = C;
  }
```

MetAcsl instantiates a metaproperty in all relevant locations

THALES

# Example: Confidentiality Metaproperty Verified with MetAcsl – Reading context

## Resulting code after generating assertions with MetAcsl and proof with Frama-C/WP:

```
/*@ meta "A_not_read";
 */
/*@ requires A ≡ B;
    ensures
      (C ≥ 0 ∧ A ≡ C ∧ B ≡ C) ∨
      (C < 0 ∧ A ≡ \old(A) ∧ B ≡ \old(B));
    assigns A, B;
 */
void foo(void)
{
  /*@ check A_not_read: _1: meta: \separated(&C, &A); */
  if (C >= 0) {
    /*@ check A_not_read: _2: meta: \separated(&C, &A); */
    A = C;
    /*@ check A_not_read: _3: meta: \separated(&C, &A); */
    B = C;
  }
  return;
}
```

MetAcsl

## Initial C code:

test4.c

```
int A, B, C;
/*@
  meta \prop, \name(A_not_read),
    \targets(\ALL), \context(\reading),
    \separated(\read, &A);
*/
/*@
  requires A==B;
  assigns A,B;
  ensures C>=0 && A==C && B==C ||
    C<0 && A==\old(A) && B==\old(B); */
void foo(){
  if ( C >= 0 ){
    A = C;
    B = C;
  }
}
```

# Examples of HILAREs

```
meta \prop, \name(Do not write to lower pages outside free),
  \targets(\diff(\ALL , {page_free})),
  \context( \writing ),

  \forall integer i; 0 <= i < MAX_PAGE_NB ==>
  \let p = pages + i;
  p->status == PAGE_ALLOCATED &&
  user_level > p->confidentiality_level ==>
  \separated(\written, p->data + (0.. PAGE_SIZE - 1));

meta \prop, \name(Free pages are never read),
  \targets(\ALL),
  \context( \reading ),

  \forall integer i; 0 <= i < MAX_PAGE_NB &&
  pages[i].status == PAGE_FREE ==>
  \separated(\read, pages[i].data + (0 .. PAGE_SIZE - 1));
```





THALES

# Application to certification of JavaCard Virtual Machine: Verification of security properties with MetAcsl

- **Integrity and Confidentiality cannot be verified with WP** as global invariants
- We use metaproperties:

name — targets all function(s) — application context: whenever a location is read

```
meta \prop,\name(meta_persi_objects_confident),\targets(\ALL),\context(\reading),

( \forall integer i; 0 <= i < gNumObjs && !gIsTrans[i] &&
                    ObjHeader[gHeadStart[i] + 0] != JCC ==>
   \separated(\read,PersiData+(gDataStart[i]..gDataEnd[i])) ); */
```

The read location must be separated from the data of any persistent object if the current context is not its owner.

- **MetAcsl** translates metaproperties into **assertions/checks** at each relevant program point.
- If all **assertions/checks** are proved, the metaproperty is proved.
- Thanks to the translation of metaproperties into **checks** that do not overload proof contexts, the metaproperty-based approach scales very well, despite a great number of generated annotations.

# Conclusion



## Large sets of properties can be automatically translated into basic annotations

- High-level (e.g. security) properties using MetAcsl, but also:
  - relational properties with RPP, temporal logic properties with Aorai, test objectives with LTest

## Various tools can be applied on the resulting annotations

- This facilitates tool collaboration

## Successful industrial application of deductive verification with Frama-C / MetAcsl

- World-first proof of real-life JavaCard Virtual Machine code
- EAL7 certificate issued by ANSSI, the French certification body
- High level of automation (99% of goals proved automatically)
- MetAcsl is crucial for specification of security properties



THALES

# References


- Lionel Blatter, Nikolai Kosmatov, Pascale Le Gall and Virgile Prevosto.
  "RPP: Automatic Proof of Relational Properties by Self-Composition." **TACAS 2017.** Springer.
- Virgile Robles, Nikolai Kosmatov, Virgile Prevosto, Louis Rilling, and Pascale Le Gall.
  "MetAcsl: Specification and Verification of High-Level Properties." **TACAS 2019.** Springer.
- Virgile Robles, Nikolai Kosmatov, Virgile Prevosto, Louis Rilling, and Pascale Le Gall.
  "Tame your annotations with MetAcsl: Specifying, Testing and Proving High-Level Properties". **TAP 2019**. Springer.
- Virgile Robles, Nikolai Kosmatov, Virgile Prevosto, Louis Rilling, and Pascale Le Gall.
  "Methodology for Specification and Verification of High-Level Properties with MetAcsl". **FormaliSE 2021.** IEEE.
- Adel Djoudi, Martin Hana and Nikolai Kosmatov.
  "Formal verification of a JavaCard virtual machine with Frama-C". **FM 2021.** Springer.
- Adel Djoudi, Martin Hána, Nikolai Kosmatov, Milan Křříženecký, Franck Ohayon, Patricia Mouy, Arnaud Fontaine and David Féliot.
  "A Bottom-Up Formal Verification Approach for Common Criteria Certification:
  Application to JavaCard Virtual Machine". **ERTS 2022, Best paper award**.
- Lionel Blatter, Nikolai Kosmatov, Virgile Prevosto and Pascale Le Gall.
  "An Efficient VCGen-based Modular Verification of Relational Properties." **ISOLA 2022.** Springer.
- Lionel Blatter, Nikolai Kosmatov, Virgile Prevosto and Pascale Le Gall.
  "Certified Verification of Relational Properties." **iFM 2022**. Springer.






THALES