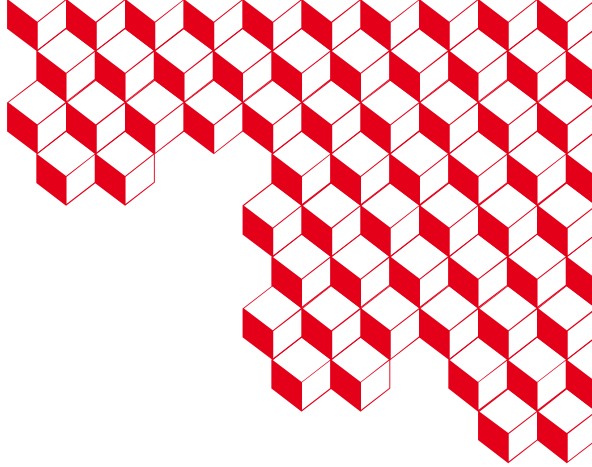


Formal methods for Code Verification with Frama-C
Focus on security properties

Patricia Mouy – patricia.mouy@cea.fr

1. Frama-C in a nutshell

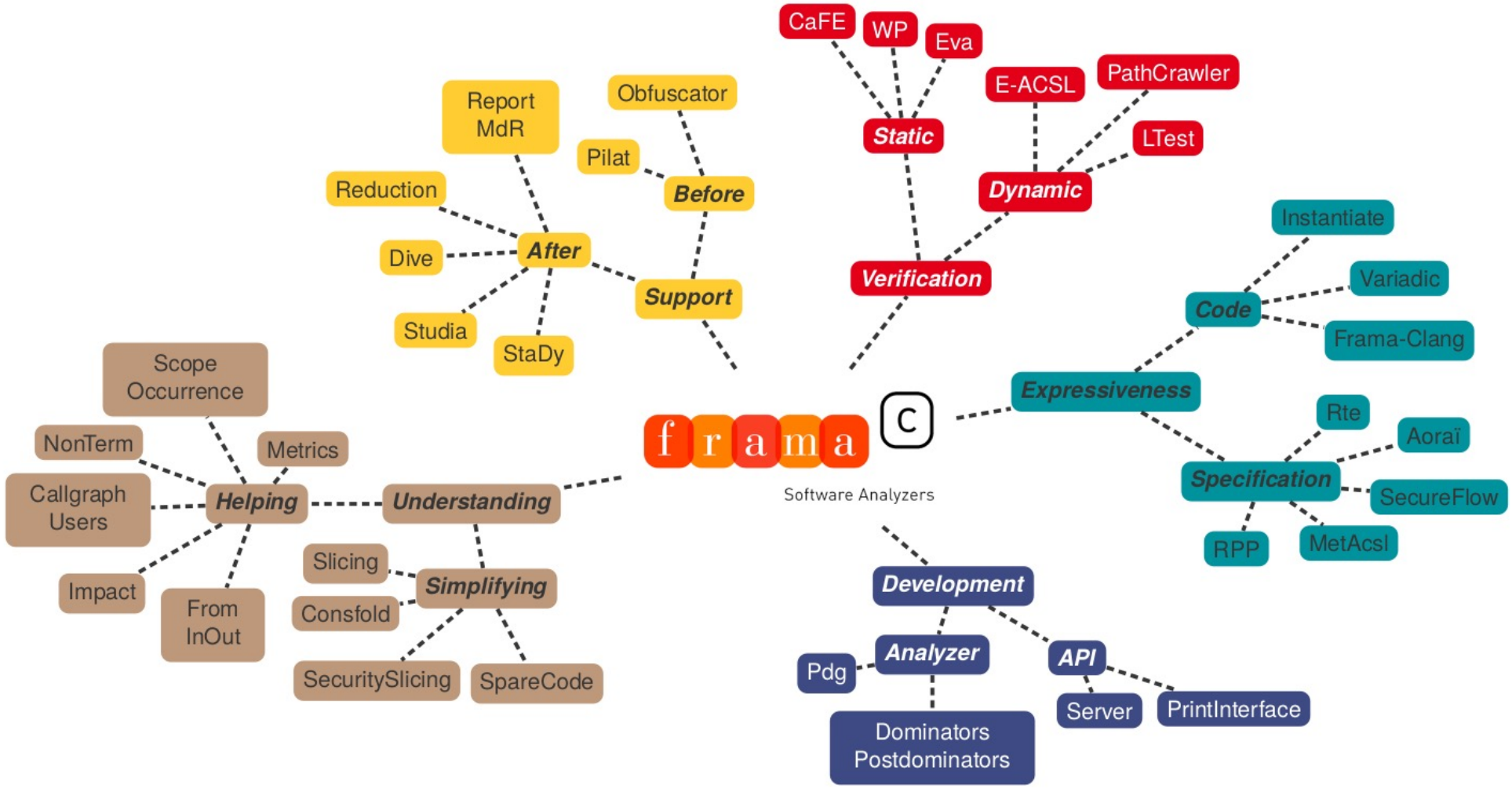
Frama-C: A platform dedicated to analysis of C Code

- Dedicated to the analysis of C89/C99/C11 code
- About 15 years of R&D
- Extensible plug-in oriented platform
  - Collaboration of analyses over the same code
  - Inter plug-in communication through ACSL formulas
- Specification language (ACSL)
  - Based on the notion of contract

frama C

Software Analyzers

Plug-ins gallery



Used in industry for years

THALES
RESEARCH & TECHNOLOGY

EDF

AIRBUS

DASSAULT
AVIATION

IRSN
INSTITUT
DE RADIOPROTECTION
ET DE SÛRETÉ NUCLÉAIRE

MITSUBISHI
ELECTRIC

- 2 releases/year
- over a hundred downloads/release
- users all over the world

Main usage of Frama-C

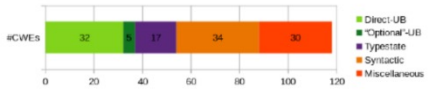
Proof of absence of runtime errors

- Based on abstract interpretation
- Mostly automatic

1) There is a claim that the tool's findings are always correct.
2) The tool produces findings for most of the program.
3) Even one incorrect finding disqualifies a tool.

An implicit criterion is that the tool is useful, not merely a toy.

interpretations. Frama-C reports led us to discover three unintentional, systematic flaws in the Juliet test suite involving 416 test cases. Our conclusion is that Frama-C satisfied the SATE V Ockham Sound Analysis Criteria.



NISTIR 8113

SATE V Ockham Sound Analysis Criteria

Paul E. Black
Athos Ribeiro

This publication is available free of charge from:
http://dx.doi.org/10.6028/NIST.IR.8113

NIST
National Institute of Standards and Technology
U.S. Department of Commerce

Examples from analysis benchmarks

```
#define TAILLE_TAB 1024
int tab[TAILLE_TAB];

void f(void){
    int index;
    for (index = 0; index < TAILLE_TAB
; index++)
    {
        tab[index] = 0;
    }
    tab[index] = 1;
}
```

out of bounds access

```
[value] warning: accessing out of bounds index. assert index < 1024;
```

```
void main(void)
{
    int* p;
    *p = 42;
}
```

uninitialized pointer

```
[value] warning: accessing uninitialized left-value. assert \initialized(&p);
```

Main usage of Frama-C

Full functional correctness

- Based on deductive verification (optimized Dijkstra weakest precondition calculus)
- Backend: Why3 + SMT solvers (+ proof assistants)
- Manual specification
- Most of proofs are automatic (90-98% depending on use cases)

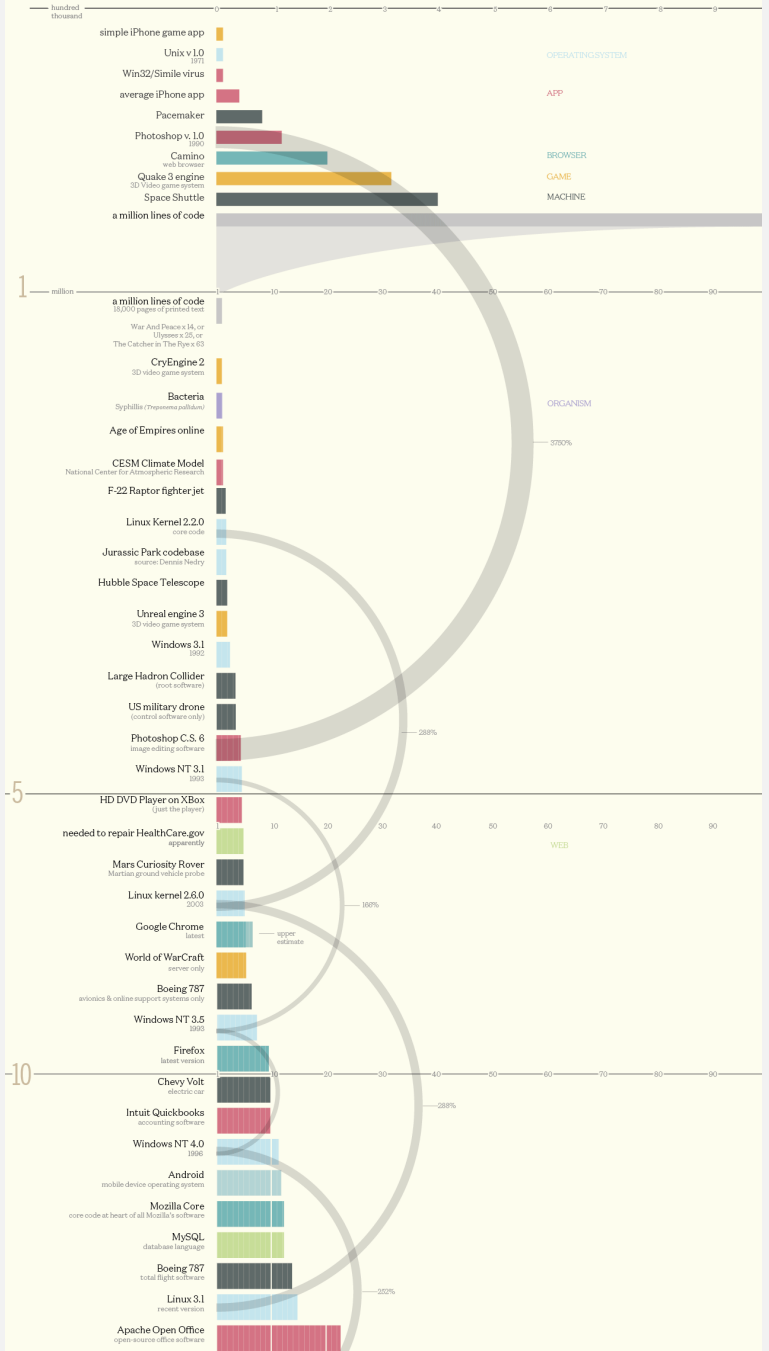
Runtime monitoring

- Detect invalid properties at runtime
- Mostly in combination with static analyzers

2. Frama-C & Code security

Codebases

Millions of lines of code



*Human Genome = 3,300 billion "lines" of code

SOFTWARE ANALYZERS

PROVING THE ABSENCE OF FLAWS

Designed at CEA & INRIA

State-of-the art analyzers based on strong logical and mathematical foundations.

An open-source, cooperative and extensible platform design for industrial support & deployment

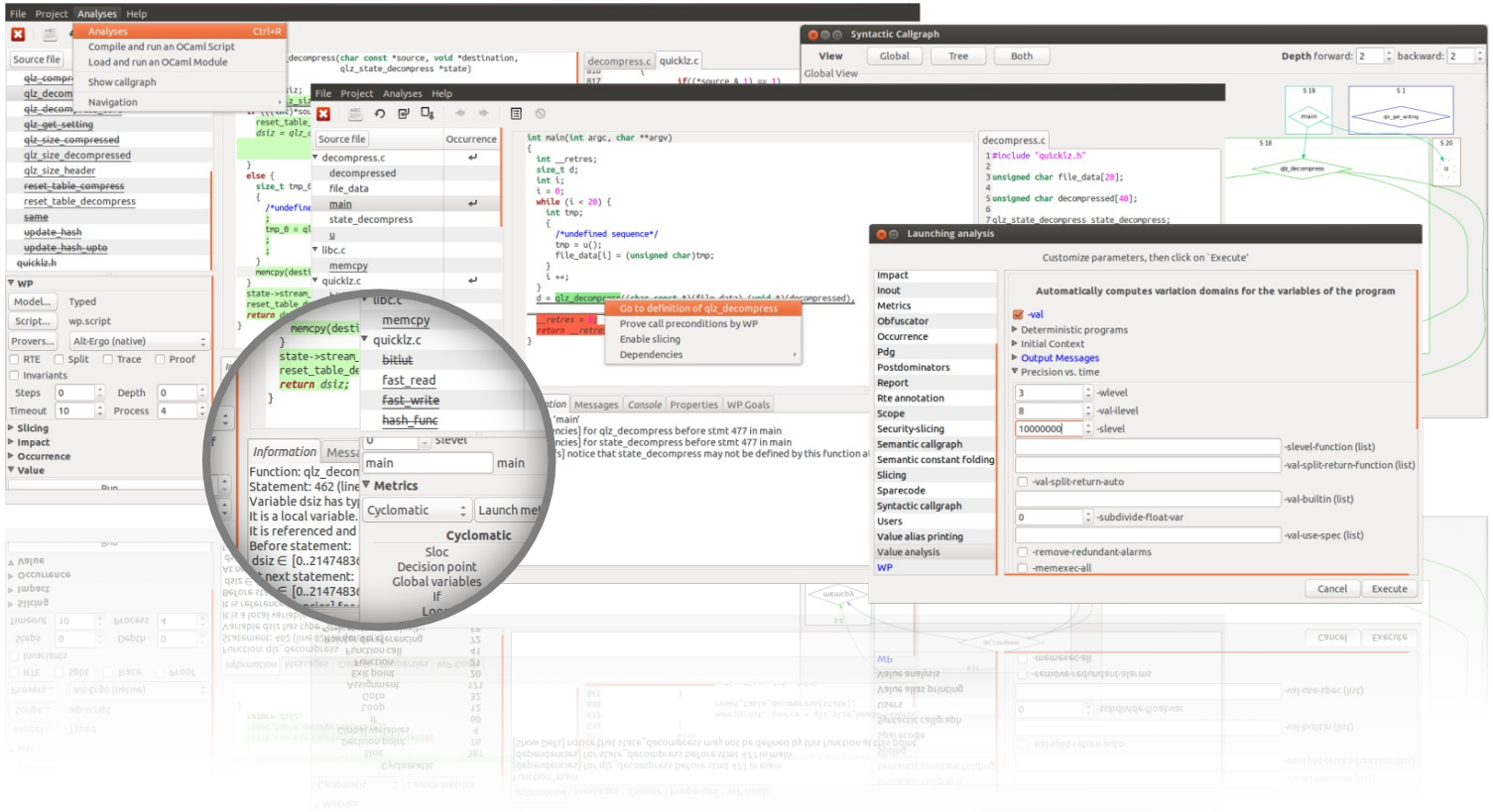
http://frama-c.com

frama C

Software Analyzers

PROOF OF ABSENCE OF UNDEFINED BEHAVIORS (buffer overflows…)

PROOF OF SECURITY PROPERTY (confidentiality, data integrity)



```
[eva] warning: signed overflow. assert -x ≤ 2147483647;
```

```
int abs(int x)
{
  int r;
  if (x >= 0)
    r = x;
  else
    r = - x;
  return r;
}
```

CWE190 – integer overflow

Frama-C with Eva satisfied the SATE VI Ockham Sound Analysis Criteria.

Warnings from Frama-C led us to discover 38 test cases with incorrect “good” code in CWE190.

PROVING SECURITY PROPERTIES ...

From ANSSI internal laboratories

From ANSSI partner evaluations

Journey to a RTE-free X

Arnaud EBALARD, Patric[illegible]OUY, and Ryad BE[illegible]
prenom.nom@ssi.gouv.fr

ANSSI

Abstract. The C programming language is a security nightmare. It is error-prone and unsafe, but, year after year, the conclusion remains the same: no credible alternative will replace C in a foreseeable future; especially in low-level developments or for constrained environments.

Additionally, even though some C developers are keen to drop this language when possible for more robust ones like ADA or Rust, converting the existing code basis to safer alternatives seems unrealistic.

One of the positive aspects with C is nevertheless that its inherent flaws became a long time ago a full-time research topic for many people. Various static analysis tools exist to try and verify security aspects of C code, from the absence of run-time errors (RTE) to the verification of functional aspects.

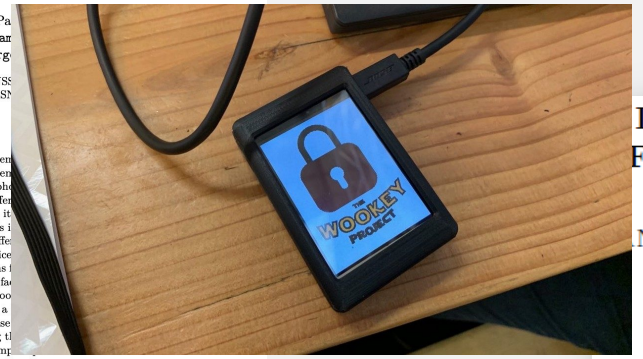
At the time of writing, none of these tools is capable of verifying the full-fledged C source code from most well-known software projects, even the smallest ones. Those tools are nonetheless getting better, and they may be able to handle large software code bases in the near future.

Doing some steps in the direction of static analysis tools is sometimes sufficient to achieve full verification of a complex piece of code. This article details this kind of meet-in-the-middle approach applied to the development in C99 of a X.509 parser, and then its later verification using Frama-C.

From CVEs to proof:
Make your USB device stack great again

Ryad Benadjila[1], Cyril Debergé[2], Pa[illegible]
[1]firstname.lastnam[illegible]
[2]cyril.deberg[illegible]

[1] ANSS[illegible]
[2] IRSN[illegible]



Abstract. Nowadays, many devices en[illegible] components are made of software elem[illegible] USB sticks, hard-disk drives, smartph[illegible] tons, IoT devices: they all usually offe[illegible] a USB software driver dealing with it[illegible] attackers are able to tamper with this i[illegible] Run Time Error (RTE) such as a buffe[illegible] code execution on the vulnerable device[illegible] is even worse when the USB stack runs [illegible] unpatchable software. This matter of fa[illegible] RTE-free USB stack with concrete proo[illegible] open-source implementation of such a [illegible] work [35], with proofs and various use[illegible] and more to come). Beyond providing t[illegible] a generic methodology to adapt comp[illegible] FRAMA-C with strong embedded contexts constraints.

1 Introduction

Software is becoming the core component of many systems, from small embedded devices to bigger desktop Personal Computers. Even for what seems to be simple and low-level tasks, dedicated hardware with hard-wired logic circuits are almost always driven by pieces of software that tend to become more and more complex. From network cards to hard-drives controllers and motherboards chipsets, nowadays every piece of hardware contains firmware that is often made of thousands of lines of (usually) C or assembly code, let alone the drivers that interact with them on the Operating System side.

Vulnerabilities in such software stacks have proven to be critical from a security standpoint [3, 24, 27, 29]: due to the *bare metal* nature of such code, any Run Time Error (RTE) allowing remote code execution (RCE)

Inter-CESTI: Methodological and Technical Feedbacks on *Hardware Devices* Evaluations

ANSSI, Amossys, EDSI, LETI, Lexfo, Oppida, Quarkslab, SERMA, Synacktiv, Thales, Trusted Labs

Abstract. The aim of the current submission is to provide both methodological and technical feedback on the "Inter-CESTI" challenge organized by ANSSI in 2019 with all 10 ITSEFs licensed for the French CSPN scheme. The purpose of this challenge is to evaluate their approaches to attack a common target representative of the "Hardware devices with boxes" domain, which groups products containing embedded software and combines hardware and software security elements. The common target chosen was the open-source and open-hardware project WooKey, presented at SSTIC 2018 [30, 31]. It is a relevant test vehicle both in terms of software and hardware due to its architecture and threat model. The article aims to capitalize on the feedback from the challenge, with a focus on the hardware and software tests that the labs were able to conduct in a white box setting, as well as the identified attack paths.

SSTIC

frama C
Software Analyzers

STRONG CYBERSECURITY EXPERTISE

Common Criteria Assurance Levels

- Under the Common Criteria model, an evaluation is carried out on a product and it is assigned an Evaluation Assurance Level (EAL).
- The Common Criteria has seven assurance levels:
  - EAL1: Functionally tested
  - EAL2: Structurally tested
  - EAL3: Methodically tested and checked
  - EAL4: Methodically designed, tested and reviewed
  - EAL5: Semiformally designed and tested
  - EAL6: Semiformally verified design and tested
  - EAL7: Formally verified design and tested.

Common Criteria

PREMIÈRE MINISTRE
Liberté
Égalité
Fraternité

Schéma français d'évaluation et de certification de la sécurité des technologies de l'information

CERTIFICAT ANSSI-CC-2022/37

Ce certificat est associé au rapport de certification ANSSI-CC-2022/37

ultiApp V5 Java Card Virtual Machine

Version 5.0

peurs : THALES DIS FRANCE SAS, INFINEON TECHNOLOGIES AG
Commanditaire : THALES DIS FRANCE SAS
Centre d'évaluation : CEA - LETI

itères Communs version 3.1, révision 5

EAL7

Proof of security properties (*data integrity, confidentiality*)

frama C
Software Analyzers

cea

STRONG CYBERSECURITY COLLABORATIONS

L'ANSSI et le CEA renforcent leur collaboration en cybersécurité

Cyrille Dupont/The Pulses

Le CEA et l'Agence nationale de la sécurité des systèmes d'information (ANSSI) renforcent leur collaboration par la signature d'un accord-cadre d'une durée de trois ans ce mardi 28 juin. Au travers de cette alliance, les deux partenaires visent à améliorer la sécurité des systèmes numériques en préparant en particulier les futures générations d'environnements d'analyse logicielle.

Publié le 29 juin 2022

AMOSSYS
EXPERT IN CYBERSECURITY

quarkslab
SECURING EVERY BIT OF YOUR DATA

CAMPUS CYBER

enisa

cnrs GDR Sécurité Informatique

DGA
Liberté • Égalité • Fraternité
RÉPUBLIQUE FRANÇAISE
MINISTÈRE DE LA DÉFENSE

THALES

OPPIDA
EXPERT EN SÉCURITÉ DES SYSTÈMES D'INFORMATION

Inria
INVENTEURS DU MONDE NUMÉRIQUE

cea

3. Work in progress

Take away

New features

- Incremental analysis
- Integration of the support of mutli-threading
- Frama-All: support of news languages
- Integration of ML techniques …

To be followed…

Merci

CEA SACLAY
91191 Gif-sur-Yvette Cedex
France
nathalie.guillaume@cea.fr
Standard. + 33 1 69 08 60 00