FROM RESEARCH TO INDUSTRY

cea tech

frama C

Software Analyzers

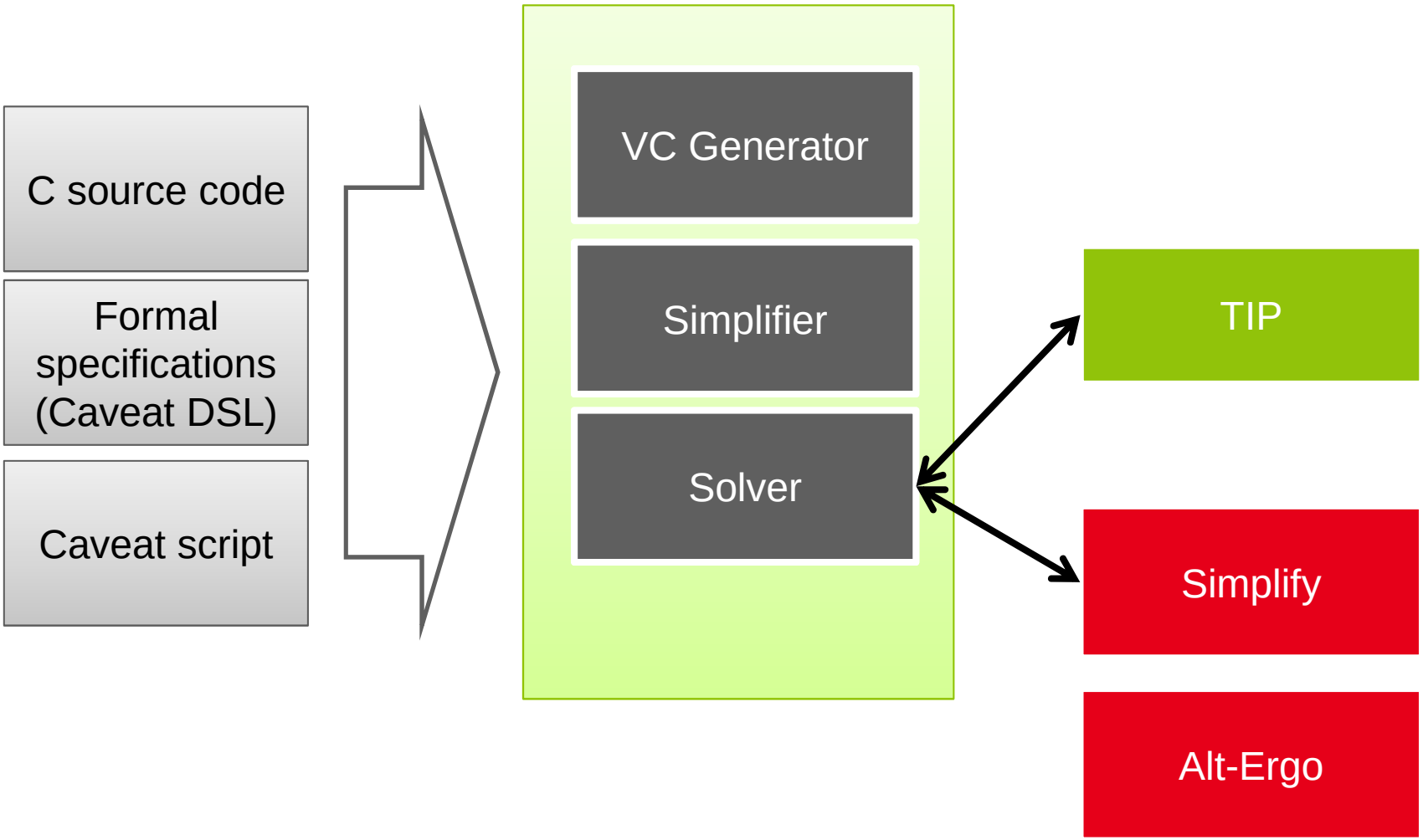
LEVERAGING DEDUCTIVE VERIFICATION IN INDUSTRIAL CONTEXTS

list

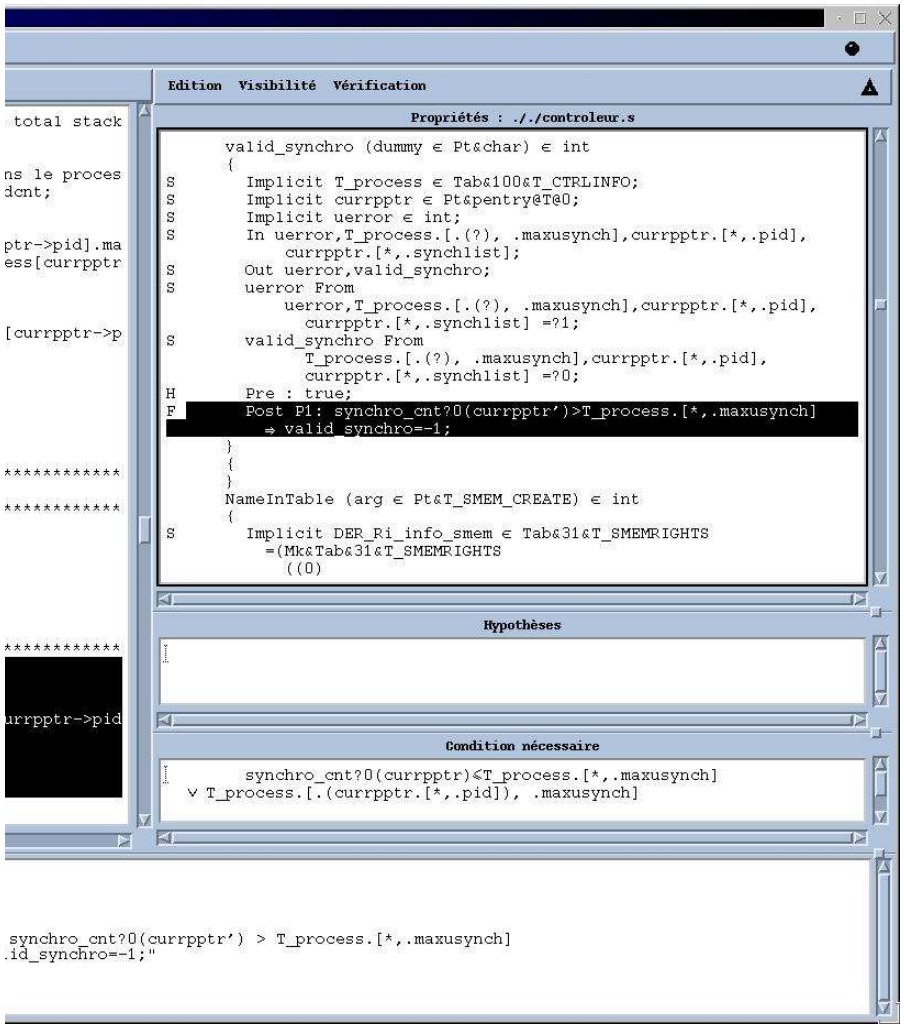
CEA’S SOFTWARE SAFETY LABORATORY

1995: LEAP FROM DYNAMIC TO STATIC CODE ANALYSIS!

CAVEAT: ARCHITECTURE



CAVEAT – MAIN FEATURES



- **Simplifier: rewriting engine and declarative rules**
- **Solver: (mainly) propositional sequent calculus**
- **Dedicated specification language**
  - first-order language
  - functional properties,
  - dependency/assignment clauses
  - functional expression of output values
  - properties of sequences of program constructions
- **Scripting and journaling**

CAVEAT – JOURNALING AND SCRIPTING

- **Usage:**
  - Build the proof project: associate specifications to program points
  - Launch the generation and proofs
  - In case of failure, script proof manipulations
    e.g. split a conjunction and send each term to the solver.
  - Finish by hand using the TIP
  - Output summary script

→ **High automation**

→ **High reuse value**

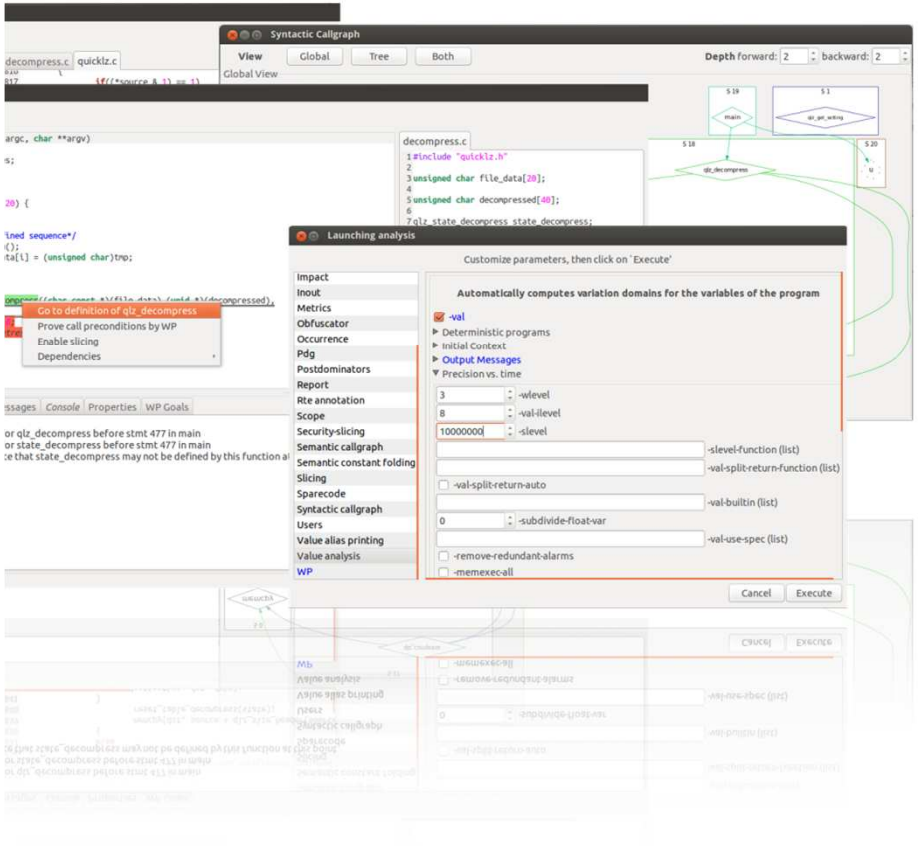
→ **Support for report generation**

```
0 : 0 chargerGph ./proj_r/demoplus.gph
1 : 2 chargerCetS 0
2 : 5 collerP 0 1 5 a       Post add : mode > 0
=> G_Command > G_Command';

3 : 5 collerP 0 1 6 a       Post
add_positive_delta : mode > 0 && delta > 0 =>
G_Command > G_Command';

4 : 5 collerP 0 2 5 a Pre : choice = 0 ||
choice = 1;
5 : 6 couperP 0 2 7

6 : 5 collerP 0 2 6 a     Post explicit :
Get_var = if (choice = 1) then G_Command else
G_Tmp;

7 : 5 collerP 0 2 -1 a Assert pathcond : At
sortie true;

8 : 5 collerP 0 3 4 a     Post : Get_modif = if
(t >= 0) then t else -t;

9 : 5 collerP 0 4 4 a Pre : 0 <= nb &&  nb<100;
10 : 6 couperP 0 4 6

11 : 5 collerP 0 4 5 a     Post : nb = 0 =>
SommeTabVal = 0;

12 : 5 collerP 0 4 6 a     Post : G_Command =
G_Command';
```

CEA’S SOFTWARE SAFETY LABORATORY

2005: FROM TOOL TO PLATFORM

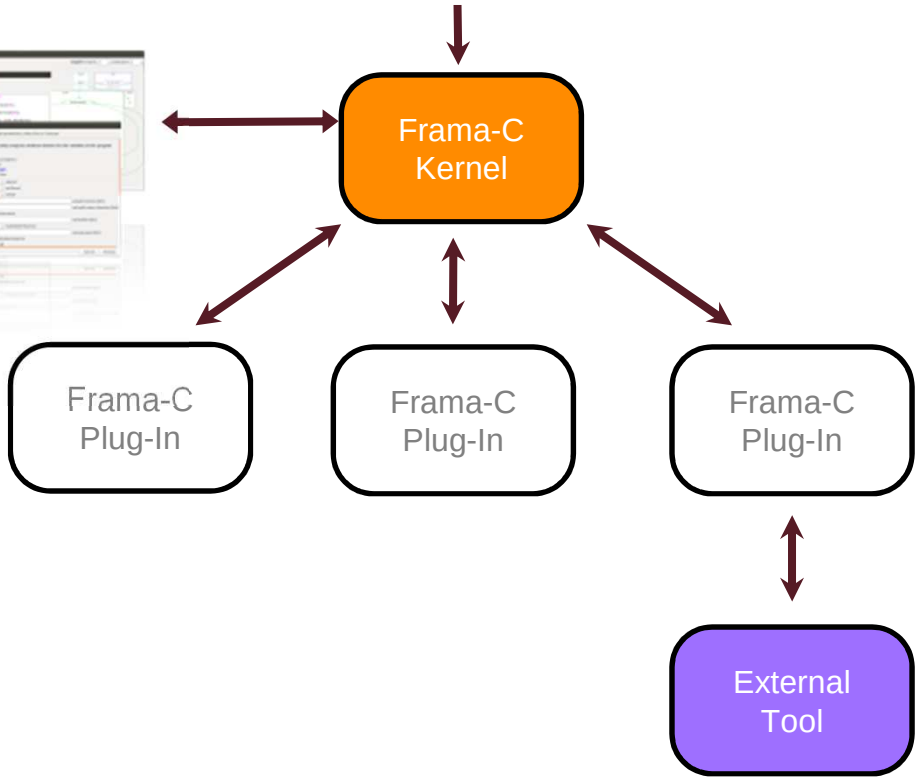
frama C
Software Analyzers



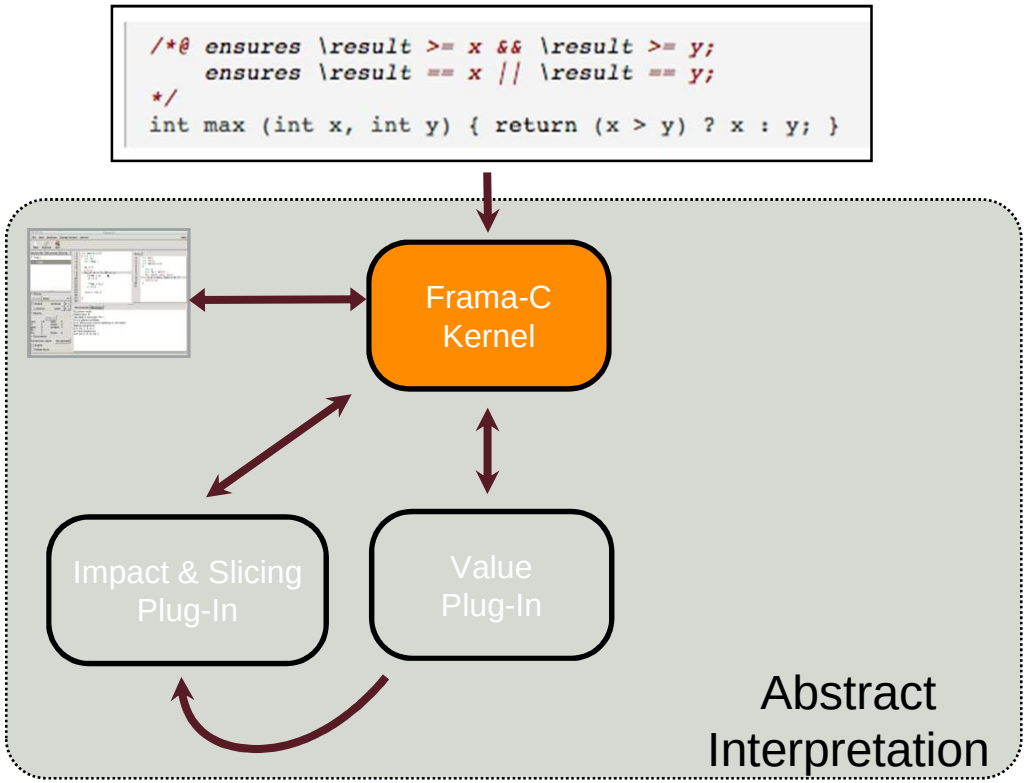
- **Designed at CEA & Inria**
  Supported by an Open-Source community
- **ANSI / ISO C99**
  Pointers & function pointers, floating-point arithmetics
- **The ANSI C Specification Language – ACSL**
  Independence from memory models
- **Coherent combination of modular and collaborative techniques**
  Abstract interpretation and deductive verification

FRAMA-C: AN EXTENSIBLE PLATFORM

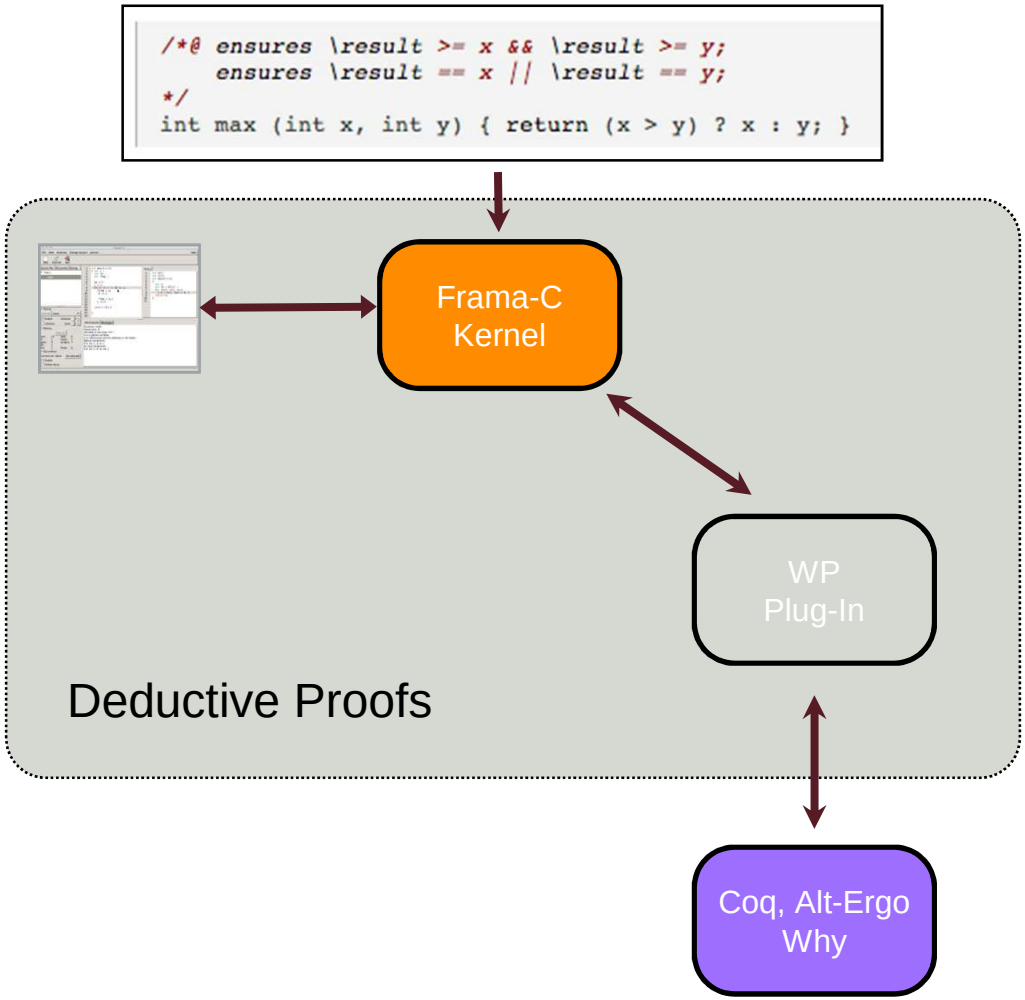
```
/*@ ensures \result >= x && \result >= y;
    ensures \result == x || \result == y;
*/
int max (int x, int y) { return (x > y) ? x : y; }
```



FRAMA-C: CORE COMPONENTS

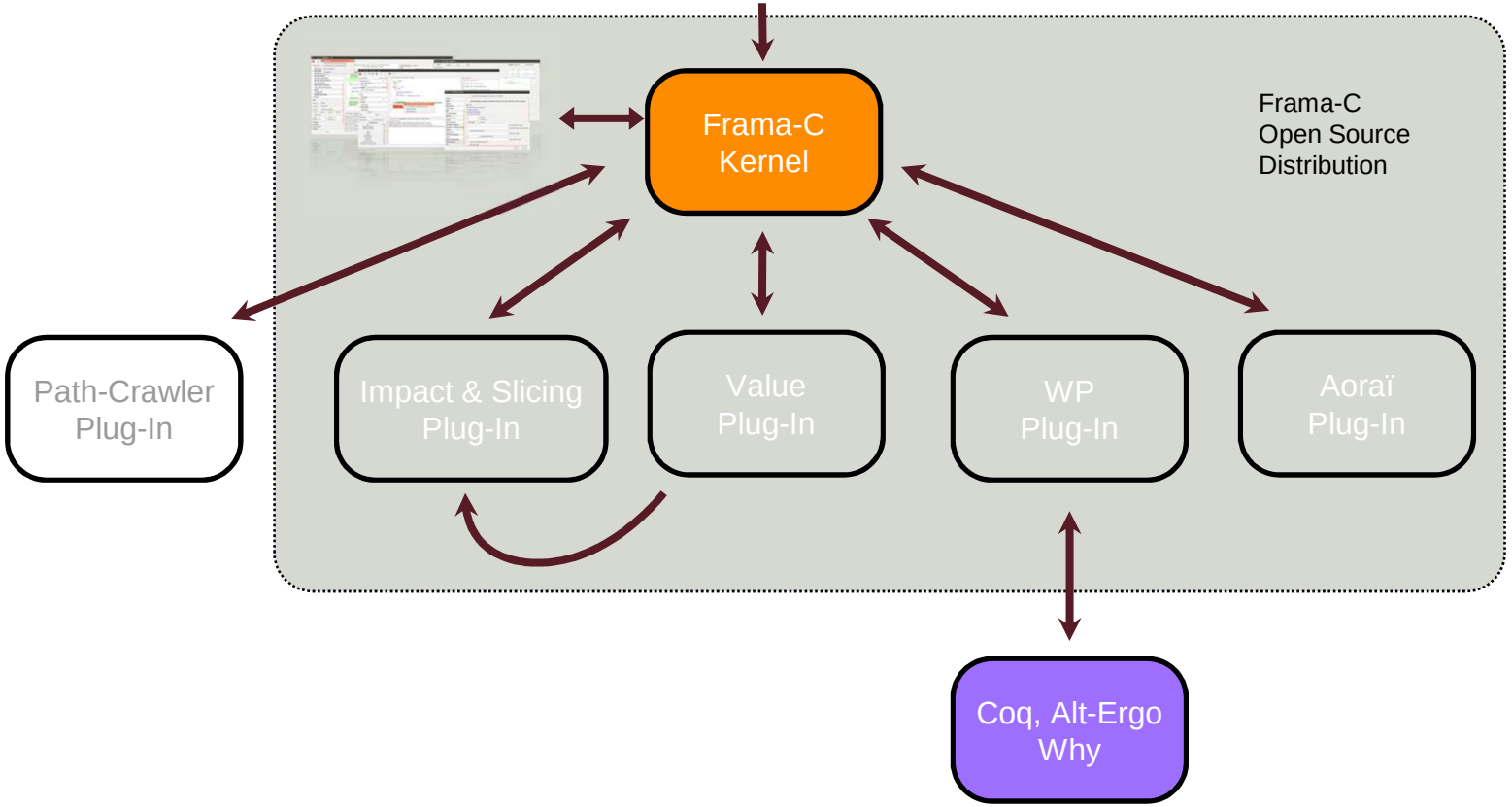


FRAMA-C: CORE COMPONENTS



FRAMA-C: CORE COMPONENTS

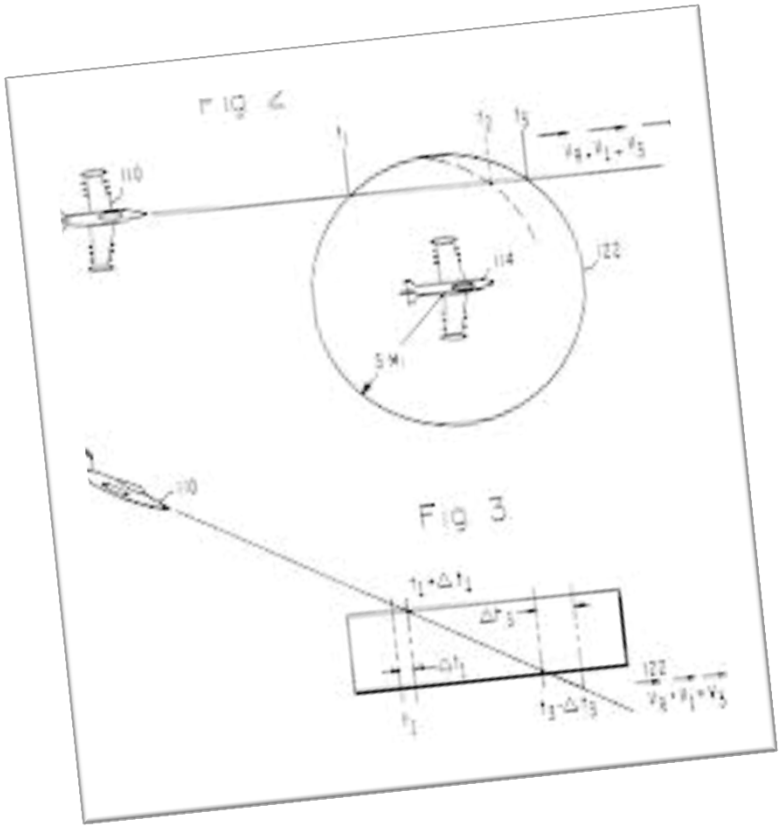
```
/*@ ensures \result >= x && \result >= y;
    ensures \result == x || \result == y;
*/
int max (int x, int y) { return (x > y) ? x : y; }
```



CEA’S SOFTWARE SAFETY LABORATORY

2014: CURRENT EXPERIMENTS

FORMAL ALGORITHMIC CONFORMANCE PROOF



Can we verify that model-based properties hold on source code?

- **Bi-dimensional conflict detection algorithm**
- **Derived from PVS specifications and proofs**
- **Floating-point vs real number arithmetic**

THE SPECIFICATION & ALGORITHM

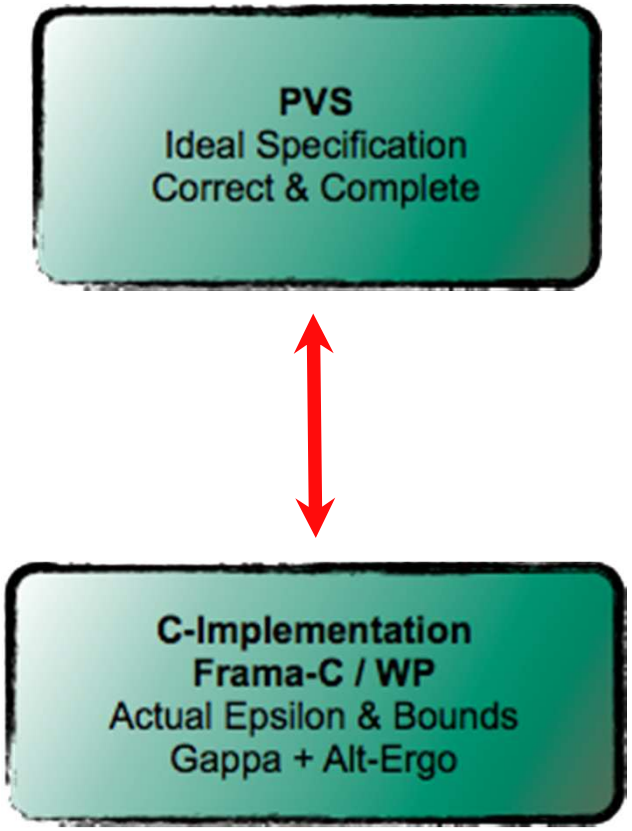
$$\begin{aligned} \mathsf{conflict}(\mathbf{s}, \mathbf{v}_o, \mathbf{v}_i, D) &\triangleq \exists\, t \in [0, T], \ldots \\ \mathsf{cd2d}(\mathbf{s}, \mathbf{v}_o, \mathbf{v}_i, D) &\triangleq \Omega(\mathbf{s}, \mathbf{v}_o, \mathbf{v}_i) < D^2 \end{aligned}$$

A FORMAL GUARANTEE

Theorem (CD2D Correctness & Completeness)

$$\forall\, \mathbf{s}, \mathbf{v}_o, \mathbf{v}_i, \mathsf{conflict}(\mathbf{s}, \mathbf{v}_o, \mathbf{v}_i, D) \iff \mathsf{cd2d}(\mathbf{s}, \mathbf{v}_o, \mathbf{v}_i, D)$$

THE CHALLENGE



THE CHALLENGE

```
/*@
  requires DISTANCE(s_x) && DISTANCE(s_y);
  requires VELOCITY(v_x) && VELOCITY(v_y);
  ensures \abs(\result - omega_vvR_outDr(s_x,s_y,v_x,v_y,D,0.0,T)) < 4.0 * E_omega;
  assigns \nothing;
 */
double omega_vv(double s_x, double s_y, double v_x, double v_y){
  double tau = tau_vv(s_x, s_y, v_x, v_y);
  double vv = sqv(v_x,v_y);
  double ss = sqs(s_x,s_y);
  double r1 = vv * ss ;
  double r2 = 2.0 * tau ;
  double r3 = tau * tau ;
  double r4 = D*D*sqv(v_x,v_y);
  return r1 + r2 + r3 - r4 ;
}
```

Frama-C / WP
Gappa + Alt-Ergo

c = round<ieee_64,ne>(a+b)

APPROXIMATIONS

Theorem (Omega Rounding)

$$\forall \mathbf{s}, \mathbf{v}_o, \mathbf{v}_i, |\Omega(\mathbf{s}, \mathbf{v}_o, \mathbf{v}_i) - \Omega_F(\mathbf{s}, \mathbf{v}_o, \mathbf{v}_i)| < \epsilon$$

REPHRASING THE ALGORITHM

$$
\begin{aligned}
\mathsf{conflict}(\mathbf{s}, \mathbf{v}_o, \mathbf{v}_i, D) &\triangleq \mathsf{cd2d}_R(\mathbf{s}, \mathbf{v}_o, \mathbf{v}_i, D) \\
\mathsf{cd2d}_R(\mathbf{s}, \mathbf{v}_o, \mathbf{v}_i, D) &\triangleq \mathsf{hlos}_R(\mathbf{s}, D) \vee \mathsf{wlos}_R(\mathbf{s}, \mathbf{v}_o - \mathbf{v}_i, D) \\
\mathsf{hlos}_R(\mathbf{s}, D) &\triangleq \mathbf{s}^2 < D^2 \\
\mathsf{wlos}_R(\mathbf{s}, \mathbf{v}, D) &\triangleq \mathbf{v}^2(\Omega_R(\mathbf{s}, \mathbf{v}) - D)^2 < 0
\end{aligned}
$$

REPHRASING THE ALGORITHM

$$
\begin{aligned}
\mathsf{cd2d}_F(\mathbf{s}, \mathbf{v}_o, \mathbf{v}_i, D) &\triangleq \mathsf{hlos}_F(\mathbf{s}, D) \vee \mathsf{wlos}_F(\mathbf{s}, \mathbf{v}_o - \mathbf{v}_i, D) \\
\mathsf{hlos}_F(\mathbf{s}, D) &\triangleq \mathbf{s}^2 < D^2 + \epsilon_1 \\
\mathsf{wlos}_F(\mathbf{s}, \mathbf{v}, D) &\triangleq \mathsf{omegavv}_F(\mathbf{s}, \mathbf{v}, D) < \epsilon_2 \\
|\mathsf{omegavv}_F(\mathbf{s}, \mathbf{v}, D) - \omega| &< \epsilon_2 \\
\text{with } \omega &\triangleq \mathbf{s}^2\mathbf{v}^2 + 2\tau\,\mathbf{s}\cdot\mathbf{v} + \tau^2 - D^2\mathbf{v}^2 \\
\text{and } \tau &\triangleq [-\mathbf{s}\cdot\mathbf{v}]_0^T \mathbf{v}^2
\end{aligned}
$$

SAFETY AND FAIRNESS

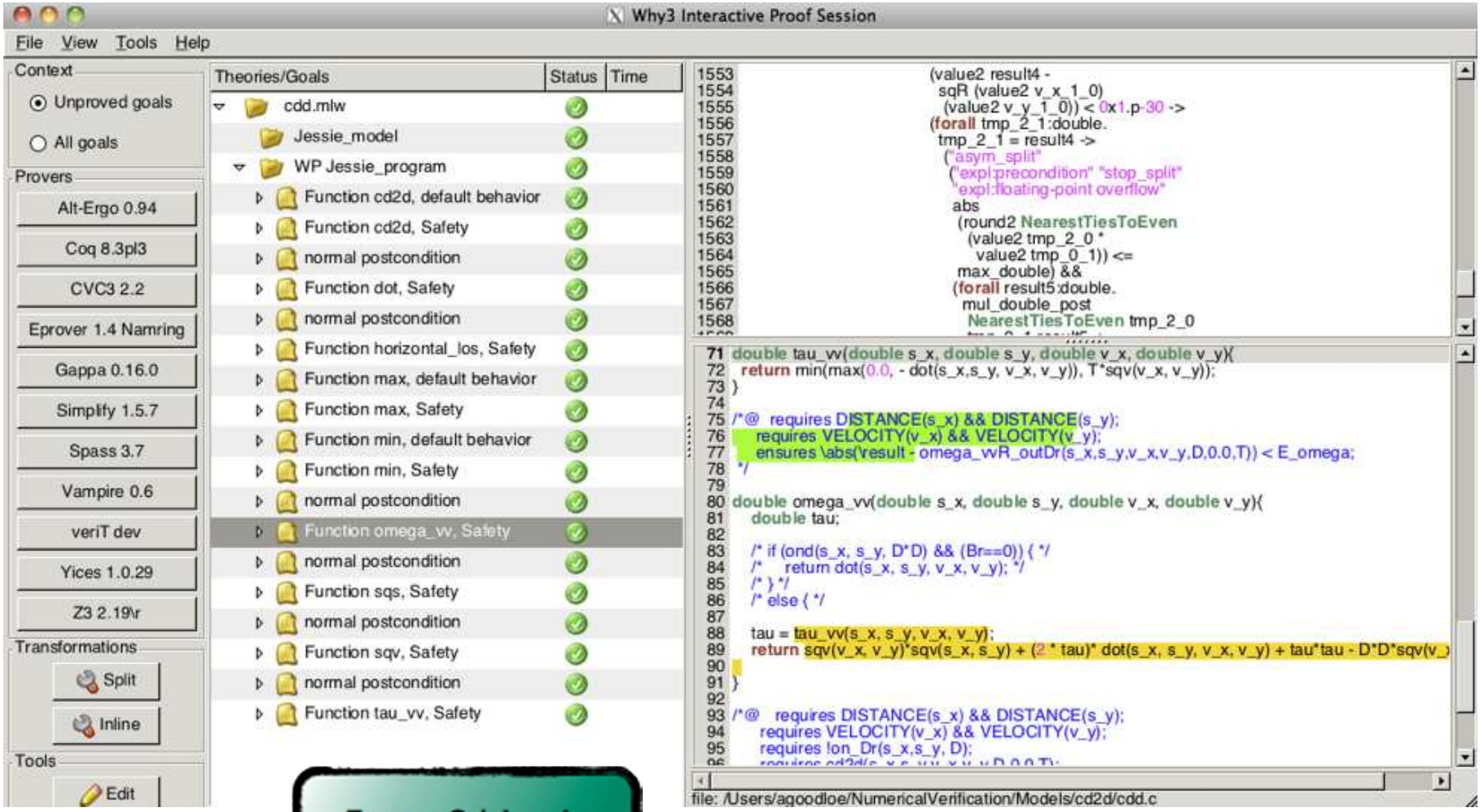
Theorem (CD2D Safety)

$$\forall \mathbf{s}, \mathbf{v}_o, \mathbf{v}_i, \mathsf{conflict}(\mathbf{s}, \mathbf{v}_o, \mathbf{v}_i, D) \implies \mathsf{cd2d}_F(\mathbf{s}, \mathbf{v}_o, \mathbf{v}_i, D)$$

Theorem (CD2D Fairness)

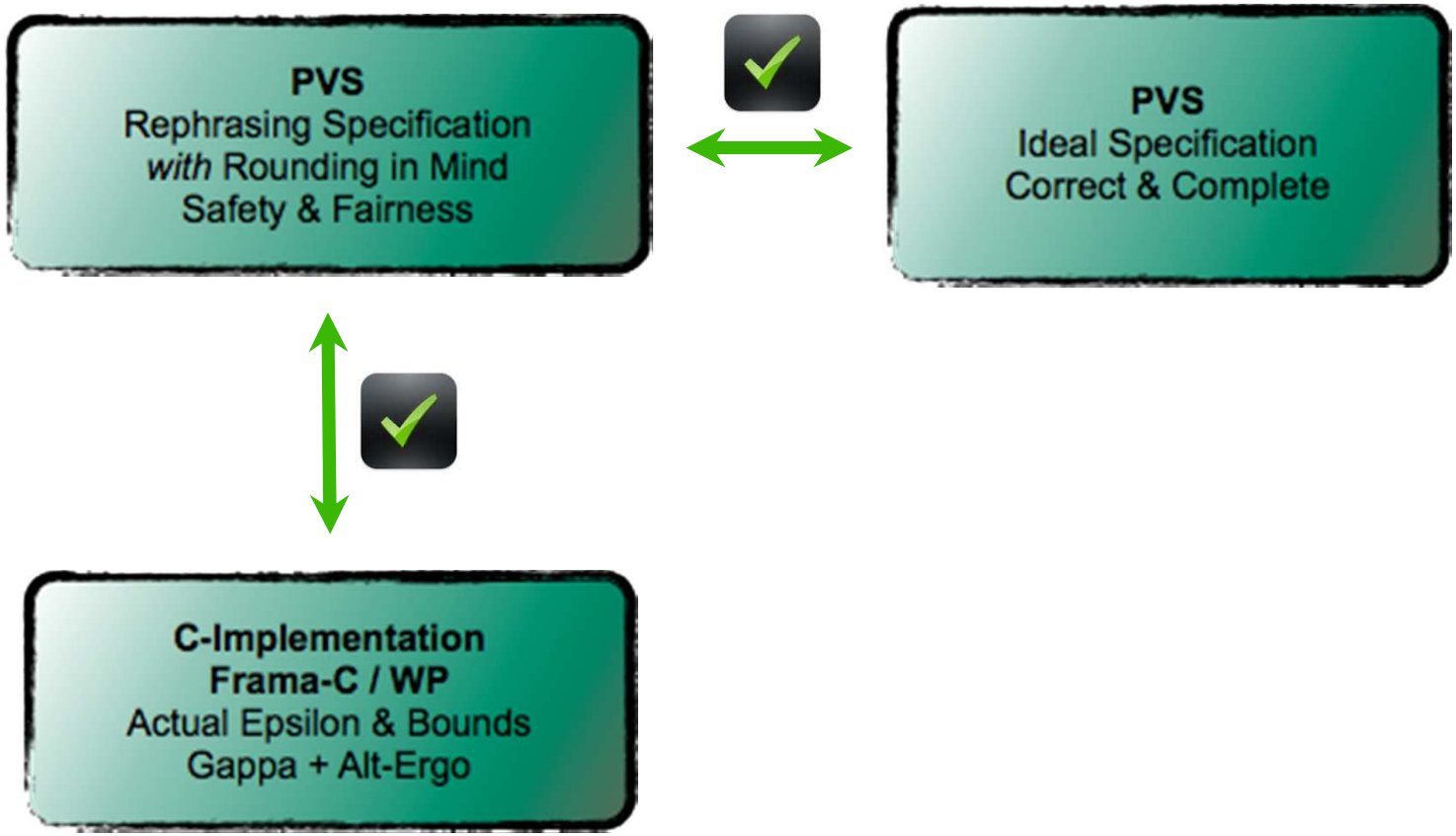
Choose α *close to* 2ϵ *such that* $(D+\alpha)^2 > D^2 + 2\epsilon$, *then:*

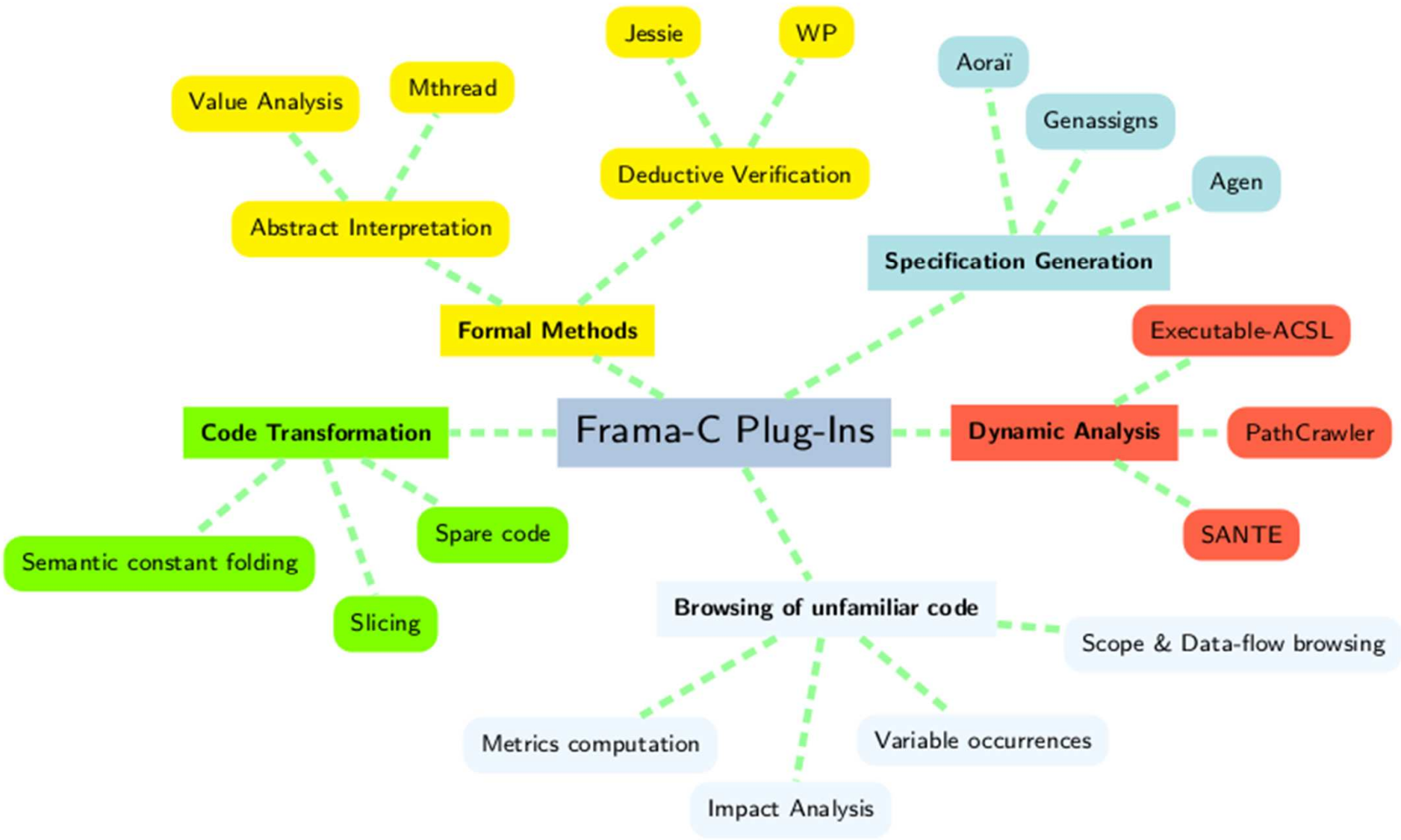
$$\forall \mathbf{s}, \mathbf{v}_o, \mathbf{v}_i, \mathsf{cd2d}_F(\mathbf{s}, \mathbf{v}_o, \mathbf{v}_i, D) \implies \mathsf{conflict}(\mathbf{s}, \mathbf{v}_o, \mathbf{v}_i, D+\alpha)$$



Frama-C / Jessie
Gappa + Alt-Ergo

CONCLUSION


PVS
Rephrasing Specification
with Rounding in Mind
Safety & Fairness
PVS
Ideal Specification
Correct & Complete
C-Implementation
Frama-C / WP
Actual Epsilon & Bounds
Gappa + Alt-Ergo


Jessie
WP
Aoraï
Value Analysis
Mthread
Genassigns
Deductive Verification
Agen
Abstract Interpretation
Specification Generation
Formal Methods
Executable-ACSL
Code Transformation
Frama-C Plug-Ins
Dynamic Analysis
PathCrawler
Spare code
SANTE
Semantic constant folding
Slicing
Browsing of unfamiliar code
Scope & Data-flow browsing
Metrics computation
Variable occurrences
Impact Analysis

TRUST IN SOFT

Frama-C for software safety and security

www.trust-in-soft.com

SOFTWARE DEVELOPERS

- Industrial support
- Commercial licenses
- Preinstalled workstations

SOFTWARE INTEGRATORS

Off-the-shelf validation kits for common open-source packages

SERVICE PROVIDERS

Dedicated affiliate programs

Laboratoire pour la Sûreté des Logiciels
Département d'Ingénierie des Logiciels et Systèmes
CEA LIST

Florent Kirchner
florent.kirchne@cea.fr

METHODS AND TOOLS FOR HCS&S

- **Scientific roots and community**
  - Formal proof
  - Model checking
  - Constraint solving
  - Simulation
  - Abstract interpretation
  - Test case generation
  - Architecture Exploration
  - Synchronous languages
- **Prototyping and development of industrial-strength tools**
  for academia
  for the industry

Guided by industrial requirements

Scaling & Performance

- **Objectives**
  Raise the level of confidence in software
  Lower the costs of verification

Technological strategy	**creation of collaborative platforms**
Scientific strategy:	**combination of approaches**
Applicative strategy:	**cross-domain fertilization – aero, space, rail, energy, banking, defense**

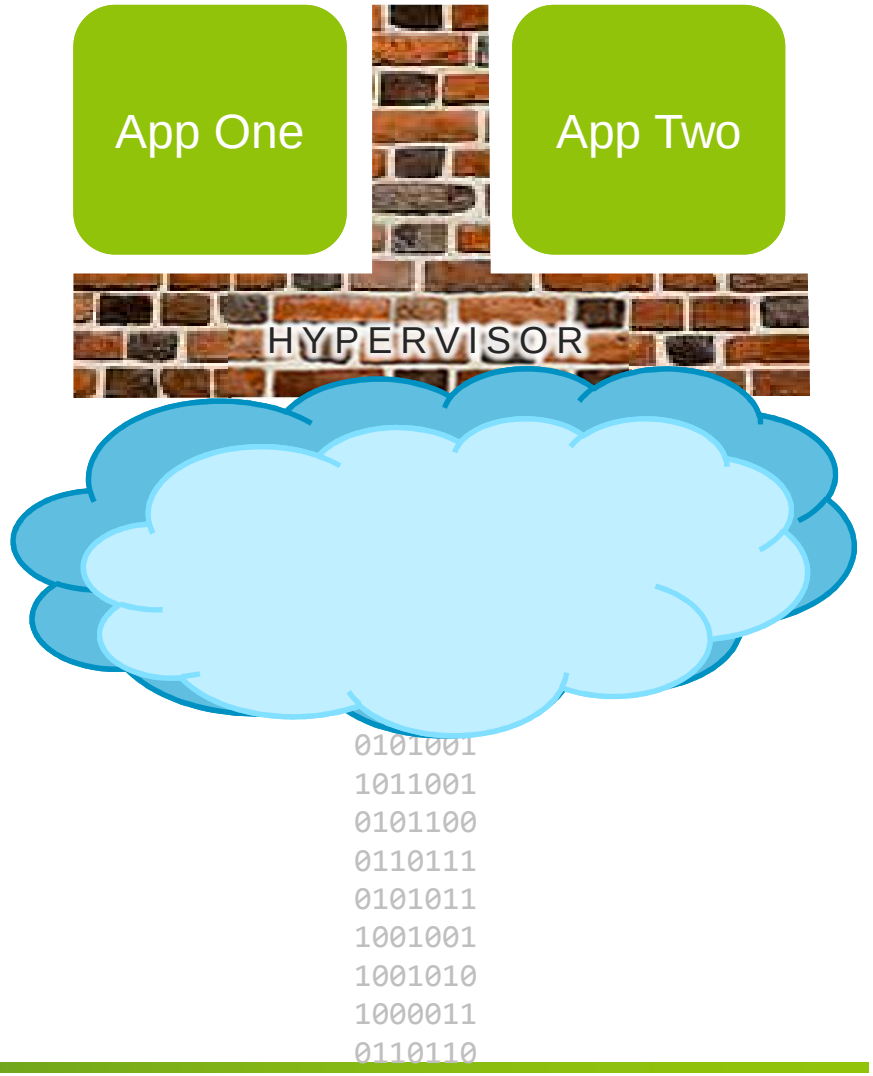
```
unsigned int M ;
/*@
requires \valid (p) && \valid (q);
ensures  M == (*p + *q) / 2;
*/
void mean (  unsigned int* p,
             unsigned int* q ) {
if (* p >= * q )
  M = (* p - * q ) / 2 + * q ;
else
  M = (* q - * p ) / 2 + * p ;
}
```

- Caller-callee contract
- Callee **requires** some pre-conditions from the caller
- Callee **ensures** some post-conditions hold when it returns

ADVANCED VALIDATION OF A SET OF HYPERVISOR PROPERTIES

Can we instantiate and verify security-policies on a custom execution platform?

- **Software property specifications for confidentiality**
- **Derived from system-level policies**
- **Formal interactive verification of the page allocation algorithm**



- **Autres: B.**

- **Frama-C/WP et Caveat: 20 minutes de présentation de l'outil. Différences entre Caveat et Frama-C, évolutions. Types de propriétés, langage utilisé. Caveat fortement auto, bas niveau ok. Périmètre de Caveat limité par l'usage des pointeurs / aliasing. Limite sur modèle bas niveau, travaux en cours.**

- **NASA (en fin de 1ere partie, emphase sur l'aspect académique),**

- **30 minutes: présentation de l'utilisation de l'outil Airbus (dont expérience certif), Dassault, Atos/CNES… ?**