LEVERAGING DEDUCTIVE VERIFICATION IN INDUSTRIAL CONTEXTS
CEA’S SOFTWARE SAFETY LABORATORY

1995: LEAP FROM DYNAMIC TO STATIC CODE ANALYSIS!
CAVEAT: ARCHITECTURE

C source code
Formal specifications (Caveat DSL)
Caveat script

VC Generator
Simplifier
Solver

TIP
Simplify
Alt-Ergo
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**
CEA’S SOFTWARE SAFETY LABORATORY

2005: FROM TOOL TO PLATFORM
- 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

```c
/*@ ensures \result >= x && \result >= y;
  ensures \result == x \iff \result == y;
*/
int max (int x, int y) { return (x > y) ? x : y; }
```
FRAMA-C: CORE COMPONENTS

```c
/*@ 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

```c
/*@ 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

```c
/*@ 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
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

\[
\text{conflict}(s, v_o, v_i, D) \triangleq \exists t \in [0, T], \ldots \\
\text{cd2d}(s, v_o, v_i, D) \triangleq \Omega(s, v_o, v_i) < D^2
\]
A FORMAL GUARANTEE

**Theorem** (CD2D Correctness & Completeness)

\[ \forall s, v_o, v_i, \text{conflict}(s, v_o, v_i, D) \iff \text{cd2d}(s, v_o, v_i, D) \]
THE CHALLENGE

PVS
Ideal Specification
Correct & Complete

C-Implementation
Frama-C / WP
Actual Epsilon & Bounds
Gappa + Alt-Ergo
THE CHALLENGE

```c
requires DISTANCE(s_x) && DISTANCE(s_y);
requires VELOCITY(v_x) && VELOCITY(v_y);
ensures \abs(\text{result} - \text{omega}_\text{vvR}_\text{outDr}(s_x,s_y,v_x,v_y,D,0.0,T)) < 4.0 \times E_{\text{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 = sgs(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;
}
```

\[ c = \text{round}_{\text{ieee}_64, \text{ne}}(a+b) \]
**Theorem** (Omega Rounding)

\[ \forall s, v_o, v_i, |\Omega(s, v_o, v_i) - \Omega_F(s, v_o, v_i)| < \epsilon \]
REP HRASING THE ALGORITHM

\[
\begin{align*}
\text{conflict}(s, v_o, v_i, D) & \triangleq \text{cd2d}_R(s, v_o, v_i, D) \\
\text{cd2d}_R(s, v_o, v_i, D) & \triangleq \text{hlos}_R(s, D) \lor \text{wlos}_R(s, v_o - v_i, D) \\
\text{hlos}_R(s, D) & \triangleq s^2 < D^2 \\
\text{wlos}_R(s, v, D) & \triangleq v^2(\Omega_R(s, v) - D)^2 < 0
\end{align*}
\]
REPHRASING THE ALGORITHM

\[ \begin{align*}
\text{cd2d}_F(s, v_o, v_i, D) & \triangleq \text{hlos}_F(s, D) \lor \text{wlos}_F(s, v_o - v_i, D) \\
\text{hlos}_F(s, D) & \triangleq s^2 < D^2 + \epsilon_1 \\
\text{wlos}_F(s, v, D) & \triangleq \text{omegavv}_F(s, v, D) < \epsilon_2 \\
|\text{omegavv}_F(s, v, D) - \omega| & < \epsilon_2 \\
\text{with } \omega & \triangleq s^2v^2 + 2\tau s \cdot v + \tau^2 - D^2v^2 \\
\text{and } \tau & \triangleq [-s \cdot v]_0^Tv^2
\end{align*} \]
SAFETY AND FAIRNESS

**Theorem (CD2D Safety)**

\[ \forall s, v_o, v_i, \text{conflict}(s, v_o, v_i, D) \implies \text{cd2d}_F(s, v_o, v_i, D) \]

**Theorem (CD2D Fairness)**

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

\[ \forall s, v_o, v_i, \text{cd2d}_F(s, v_o, v_i, D) \implies \text{conflict}(s, v_o, v_i, D + \alpha) \]
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
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

Frama-C for software safety and security

www.trust-in-soft.com
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

- 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 HYPERSISOR 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.


• NASA (en fin de 1ère partie, emphase sur l’aspect académique),

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