Context
Real World Systems

Digital System

Programming Tools

Analysis
Test
Proof

Solvers
Context
Real World Systems

Digital System

Programming Tools

Analysis
Test
Proof

Solvers

SOPRANO

→ applicability
→ efficiency
→ extensibility
Context
Real World Systems

Digital System

Programming Tools
Analysis
Test Proof

Solvers
SOPRANO

↑ applicability
↑ efficiency
↑ extensibility

safer
less bugs
used more
Toolchains

- Ada → Spark → Why3
- C → Frama-C/ WP → Solvers
Challenges

Currently uses SMT solvers:

- Good handling of arithmetic (integers, bitvectors, reals)
- Good handling of axioms
- Reasonably fast
- Agreed semantics between all provers (SMTLIB)
Challenges

Currently uses SMT solvers:

✔️ Good handling of arithmetic (integers, bitvectors, reals)
✔️ Good handling of axioms
✔️ Reasonably fast
✔️ Agreed semantics between all provers (SMTLIB)

❌ Many properties involving $x/y, x \times y, x^y, x \text{ mod } y, x \text{ rem } y$
❌ Most properties involving floating-point values
❌ Properties involving conversions between types (integers $\leftrightarrow$ bitvectors, integers $\leftrightarrow$ reals, integers $\leftrightarrow$ floats)
Floating Points

✔ Clear Semantic: \( x \oplus y = o(x + y) \)

✘ Few algebraic properties: not associative, \( x \oplus y = x \not\Rightarrow y = 0 \)

✘ Counter-intuitive: \( 0.1 \oplus \cdots \oplus 0.1 \neq 0.1 \otimes 10. = 1 \).

✘ State of the art: current bit-blasting doesn’t scale

✘ Pervasives in programs
Interesting and Simple Real Examples

```c
/*@ requires 0 ≤ x ≤ 1000; 
  requires 0 ≤ y ≤ 1000; 
  ensures 0 ≤ \result ≤ 1;   */

double x_normalisation(double x, double y){
  return x/sqrt(x*x + y*y);
}
```
Domain Specific Approach of CP

\[ X_i \in [1; 10] \implies X_0 \oplus X_1 \oplus X_2 \oplus X_3 \oplus X_4 \oplus X_5 \oplus X_6 \oplus X_7 \in [8; 80] \]

Z3 (SMT): 3s
COLIBRI (CP): 0.1s
Domain Specific Approach of CP

\[ X_i \in [1; 10] \implies X_0 \oplus X_1 \oplus X_2 \oplus X_3 \oplus X_4 \oplus X_5 \oplus X_6 \oplus X_7 \in [8; 80] \]

Z3 (SMT): 3s
COLIBRI (CP): 0.1s

\[ X_i \in [1; 10] \implies X_0 \otimes X_1 \otimes X_2 \otimes X_3 \otimes X_4 \otimes X_5 \otimes X_6 \otimes X_7 \in [1; 10^8] \]

Z3 (SMT): 31min
COLIBRI (CP): 0.1s
Precise domain propagation:
\[ x \oplus y = 0.05 \implies x, y \in [-0.1259\ldots; 0.175\ldots] \]
Precise domain propagation:
\[ x \oplus y = 0.05 \implies x, y \in [-0.1259..; 0.175..] \]

Distance graph on floating-point numbers
Precise domain propagation:
\[ x \oplus y = 0.05 \implies x, y \in [-0.1259..; 0.175..] \]

Distance graph on floating-point numbers

Monotonic functions:
\[ o(f(x)) < o(y) \implies o(x) \leq o(f^{-1}(o(y))) \]
Precise domain propagation:
\[ x \oplus y = 0.05 \implies x, y \in [-0.1259..; 0.175..] \]
Distance graph on floating-point numbers
Monotonic functions:
\[ o(f(x)) < o(y) \implies o(x) \leq o(f^{-1}(o(y))) \]
Instantiated for many functions
Precise domain propagation:
\[ x \oplus y = 0.05 \implies x, y \in [-0.1259..; 0.175....] \]

Distance graph on floating-point numbers

Monotonic functions:
\[ o(f(x)) < o(y) \implies o(x) \leq o(f^{-1}(o(y))) \]

Instantiated for many functions

Linearization of constraints for simplex
COLIBRI: Example of Reasoning

\[ 0 \leq x, y \leq 1000 \implies \sqrt{x^2 + y^2} \geq x \]
COLIBRI: Example of Reasoning

$$0 \leq x, y \leq 1000 \implies \sqrt{x^2 \oplus y^2} \geq x ?$$

$$o \left( \sqrt{o(x^2) + o(y^2)} \right) < x$$

$$o(x^2) + o(y^2) \leq o(x^2)$$

$$o(x^2) + o(y^2) = o(x^2)$$

$$o \left( \sqrt{o(x^2)} \right) < x$$

$$x < x$$ if $$o(x^2)$$ is normalized

$$o(x^2)$$ is denormalized

$$x$$ the minimum of the remaining values is a solution
COLIBRI: Example of Reasoning

\[ 0 \leq x, y \leq 1000 \implies \sqrt{x^2 \oplus y^2} \geq x ? \]

\[ o\left(\sqrt{o(x^2) + o(y^2)}\right) < x \]
\[ o(x^2) + o(y^2) \leq o(x^2) \]
\[ o(x^2) + o(y^2) = o(x^2) \]
\[ o\left(\sqrt{o(x^2)}\right) < x \]

\[ x < x \text{ if } o(x^2) \text{ is normalized} \]
\[ o(x^2) \text{ is denormalized} \]
\[ x \text{ the minimum of the remaining values is a solution} \]

There is a counter-example!
Interesting and Simple Real Examples:

Corrected

```c
/*@ requires 0.0001 \leq x \leq 1000;
  requires 0.0001 \leq y \leq 1000;
  ensures 0 \leq \text{\result} \leq 1;  @*/

double x_normalisation(double x, double y) {

  return x / sqrt(x*x + y*y);
}
```
procedure User_Rule_7 (X, Y, Z, A : Float;
                        Res : out Boolean)

  is
  begin
    pragma Assume (Z ≥ 0.0);
    pragma Assume (X ≥ Y);
    pragma Assume (Y ≥ Z);
    pragma Assume (X > Z);
    pragma Assume (A ≥ 1.0);
    Res := (X − Y) / (X − Z) ≤ A;
    pragma Assert (Res);  —— valid
end User_Rule_7;
Other Examples: From SPARK User Rule

\[ A \leq \frac{X \oplus Y}{X \oplus Z} \leq B \quad \text{with ...} \]

\[ \sqrt{X^2 \oplus Y^2} \leq X \quad \text{with ...} \]

\[ \frac{X}{\sqrt{X^2 \oplus Y^2}} \leq 1 \quad \text{with ...} \]
Floating-point rounding operator on rational constants; Interval domains

\begin{axiom}
\textit{rounding\_operator\_1} :
\forall x : \text{real} . \\
\forall i, j : \text{real} . \\
\forall md : \text{fpa\_rounding\_mode} . \\
\forall p, m : \text{int} \\
[\text{round}(m,p,md,x), x \in [i, j]]. \\
i \leq x \leq j \rightarrow \\
\text{round}(m,p,md,i) \leq \text{round}(m,p,md,x) \leq \text{round}(m,p,md,j)
\end{axiom}

(Kailiang Ji, post-doc SOPRANO)
## Floating-Point Arithmetic: Recap

Progression of COLIBRI and Alt-Ergo on AdaCore benchmarks:

<table>
<thead>
<tr>
<th></th>
<th>Before SOPRANO</th>
<th>Current Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>COLIBRI</td>
<td>18 / 28</td>
<td>25 / 28</td>
</tr>
<tr>
<td>Alt-Ergo</td>
<td>2 / 28</td>
<td>19 / 28</td>
</tr>
</tbody>
</table>
COLIBRI:
- High-level view of bitvectors
- New propagations for integers ↔ bitvectors
Evaluations & Diffusion

- AdaCore examples
Evaluations & Diffusion

- AdaCore examples
- Participation at SMT-COMP 2017
COLIBRI on SMT-COMP FP category
Evaluations & Diffusion

- AdaCore examples
- Participation at SMT-COMP 2017
Evaluations & Diffusion

- AdaCore examples
- Participation at SMT-COMP 2017

COLIBRI: Freeware For Research


Floating-Point Arithmetic: Monotonic functions
(CEA, UPSud)

Theorem
Let \( D, E \subset \mathbb{R} \), \( f : D \mapsto E \) and \( f^{-1} : E \mapsto D \) such that

- \( \forall x : D, f^{-1}(f(x)) = x \)
- \( f \) increasing

We have

- \( \forall x \in D, o(y) \in E, o(f(x)) < o(y) \implies o(x) \leq o(f^{-1}(o(y))) \)
- \( \forall x \in D, y \in E, o(f(x)) < o(f(y)) \implies x < y \)

Instantiated for many functions in COLIBRI’s DBM
Interesting and Simple Real Examples

```c
/*@ ensures \result \leq (double) 1; @*/
double test2(){
    double x = read_sensor();
    /*@ assert (double) 0 \leq x \leq (double) 1000; @*/
    double y = read_sensor();
    double z = read_sensor();

    x = x * x + z * z + y * y + 1;

    if (z \leq y){
        return (x-y)/(x-z);
    } else {
        return (x-z)/(x-y);
    }
}
```
The Problems

SMT

Engine

CP

Engine

30 May 2017 | SOPRANO project | p. 24