Refining Abstract Interpretation Based Value Analysis with Constraint Programming Techniques

Olivier Ponsini,
Claude Michel, Michel Rueher

University of Nice-Sophia Antipolis / I3S – CNRS
France

Acknowledgements to Sylvie Putot, Éric Goubault and Franck Védrine (CEA-LIST)

Introduction

- **Problem:** verification of programs with floating-point computations

  ⇝ Embedded systems written in C (transportation, nuclear plants)

- **Classical approach:** Abstract Interpretation

  + **scalability**

  − **precision**

- **Proposition:** Combining constraint programming (**CP**) and Abstract Interpretation (**AI**)

Floating-point arithmetic pitfalls

Rounding $\rightsquigarrow$ Counter-intuitive properties

$$(0.1)_{10} = (0.000110011001100\cdots)_2$$

simple precision $\rightsquigarrow$ 0.100000001490116119384765625

- Neither associative nor distributive operators

  $$(-10000001 + 10^7) + 0.5 \neq -10000001 + (10^7 + 0.5)$$

- Absorption, cancellation phenomena

  Absorption: $10^7 + 0.5 = 10^7$

  Cancellation: $((1 - 10^{-7}) - 1) * 10^7 = -1.192...(\neq -1)$

$\rightarrow$ Floats are source of errors in programs

Real numbers versus floating-point numbers semantics

Programs run over the floats BUT

- Specification $\rightsquigarrow$ written with the semantics of reals "in mind"
- Program $\rightsquigarrow$ written with the semantics of reals "in mind"

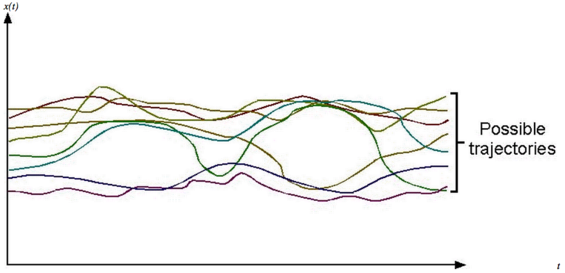
Difference between semantics $\rightsquigarrow$ problems

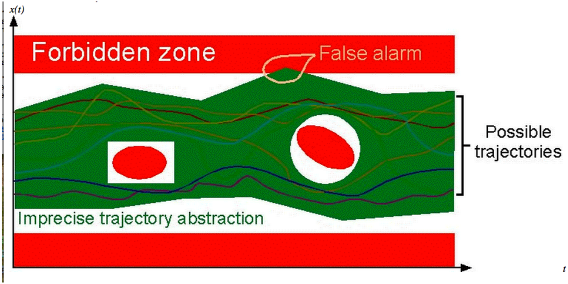
Classical Approach: static analysis from source code

- Abstraction of program states
  - Showing absence of runtime errors
  - Estimating rounding errors and their propagation
  - Checking properties of programs

- Problems
  - Approximations may be very coarse
  - Over-approximation $\rightsquigarrow$ possible false alarms

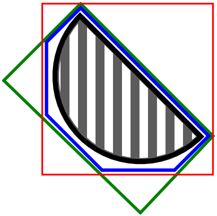
AI & False alarm





From Cousot: `http://www.di.ens.fr/~cousot/AI/IntroAbsInt.html`

Abstract domains



Intervals, **zonotopes**, polyhedra...

Zonotopes: convex polytopes with a central symmetry

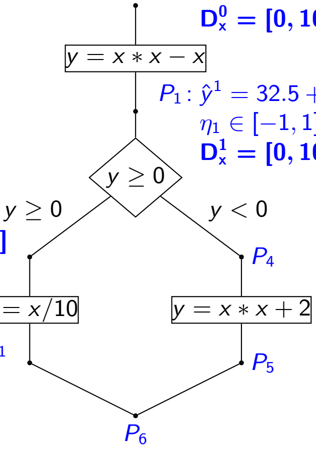
Sets of affine forms

$$\left.\begin{array}{c} \hat{a} = a_0 + a_1\varepsilon_1 + \cdots + a_n\varepsilon_n \\ \hat{b} = b_0 + b_1\varepsilon_1 + \cdots + b_n\varepsilon_n \\ \vdots \end{array}\right\} \quad \text{with } \varepsilon_i \in [-1, 1]$$

- \+ Good trade-off between performance and precision
- – Not very accurate for nonlinear expressions
- – Not accurate on very common program constructs such as conditionals

Example 1: Abstract Interpretation (zonotopes)

```
float x = [0,10];
float y = x*x - x;
if (y >= 0)
  y = x/10;
else
  y = x*x + 2;
```



$P_0 : \hat{x}^0 = 5 + 5\varepsilon_1 \quad \varepsilon_1 \in [-1, 1]$

$\mathbf{D_x^0} = \mathbf{[0, 10]}$

$P_1 : \hat{y}^1 = 32.5 + 45\varepsilon_1 + 12.5\eta_1$

$\eta_1 \in [-1, 1]$

$\mathbf{D_x^1} = \mathbf{[0, 10]} \quad \mathbf{D_y^1} = \mathbf{[-10, 90]}$

$P_2 : \hat{y}^2 = \hat{y}^1 \quad \mathbf{D_x^2} = \mathbf{[0, 10]}$

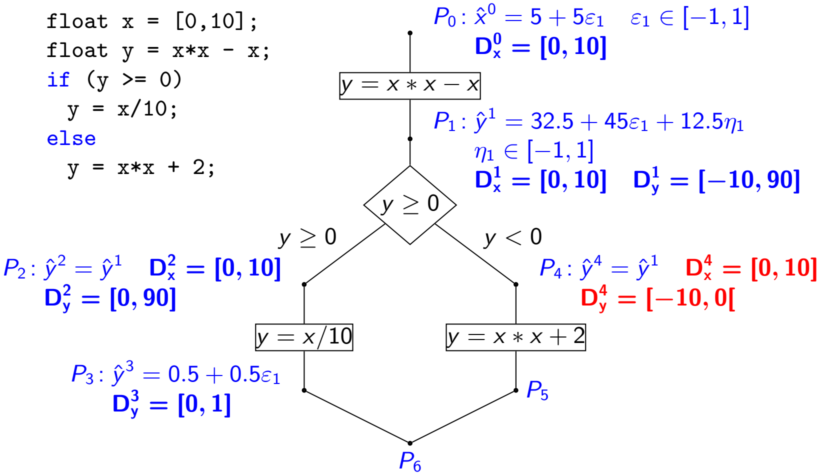
$\mathbf{D_y^2} = \mathbf{[0, 90]}$

$P_3 : \hat{y}^3 = 0.5 + 0.5\varepsilon_1$

$\mathbf{D_y^3} = \mathbf{[0, 1]}$

Example 1: Abstract Interpretation (zonotopes)

```
float x = [0,10];
float y = x*x - x;
if (y >= 0)
  y = x/10;
else
  y = x*x + 2;
```

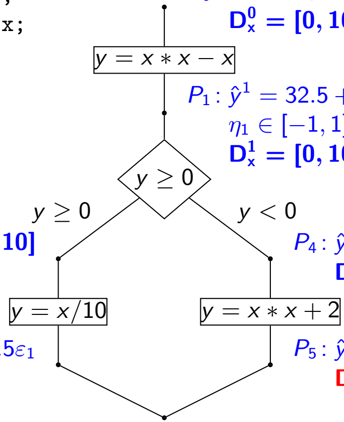


Example 1: Abstract Interpretation (zonotopes)

```
float x = [0,10];
float y = x*x - x;
if (y >= 0)
  y = x/10;
else
  y = x*x + 2;
```

$P_0 : \hat{x}^0 = 5 + 5\varepsilon_1 \quad \varepsilon_1 \in [-1, 1]$

$\mathbf{D_x^0 = [0, 10]}$



$P_1 : \hat{y}^1 = 32.5 + 45\varepsilon_1 + 12.5\eta_1$

$\eta_1 \in [-1, 1]$

$\mathbf{D_x^1 = [0, 10]} \quad \mathbf{D_y^1 = [-10, 90]}$

$P_2 : \hat{y}^2 = \hat{y}^1 \quad \mathbf{D_x^2 = [0, 10]}$

$\mathbf{D_y^2 = [0, 90]}$

$P_4 : \hat{y}^4 = \hat{y}^1 \quad \mathbf{D_x^4 = [0, 10]}$

$\mathbf{D_y^4 = [-10, 0[}$

$P_3 : \hat{y}^3 = 0.5 + 0.5\varepsilon_1$

$\mathbf{D_y^3 = [0, 1]}$

$P_5 : \hat{y}^5 = 39.5 + 50\varepsilon_1 + 12.5\eta_1$

$\mathbf{D_y^5 = [2, 102]}$

$P_6 : \hat{y}^6 = \hat{y}^3 \cup \hat{y}^5 = 39.5 + 0.5\varepsilon_1 + 62\eta_2$

$\eta_2 \in [-1, 1]$

$\mathbf{D_y^6 = D_y^3 \cup D_y^5 = [0, 102]}$

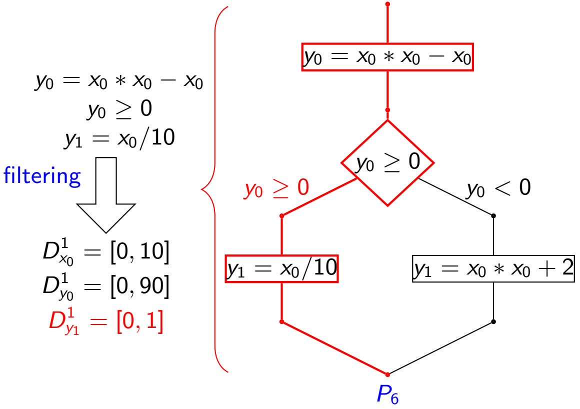
Our Constraint Programming approach

Use of local consistencies to “shave” the domains computed by AI

1. Build a constraint system C_i for each branch between two join nodes (N_1,N_2) in the CFG of the program
2. With each C_i, use local consistencies to shrink the domains computed by AI at node N_2
3. Compute the union D_{N_2} of the reduced domains from each C_i
4. Continue analysis from node N_2 with domains D_{N_2}

Example 1: our Constraint Programming approach

$P_0 : D_{x_0} = [0, 10] \quad D_{y_0} = [-10, 90] \quad D_{y_1} = [0, 102]$



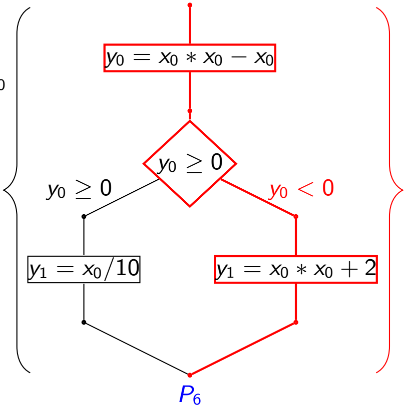
Example 1: our Constraint Programming approach

$P_0: D_{x_0} = [0, 10] \quad D_{y_0} = [-10, 90] \quad D_{y_1} = [0, 102]$

$y_0 = x_0 * x_0 - x_0$
$y_0 \geq 0$
$y_1 = x_0 / 10$

filtering

$D^1_{x_0} = [0, 10]$
$D^1_{y_0} = [0, 90]$
$D^1_{y_1} = [0, 1]$



$y_0 = x_0 * x_0 - x_0$
$y_0 < 0$
$y_1 = x_0 * x_0 + 2$

filtering

$\mathbf{D^2_{x_0} = [0, 1.026]}$

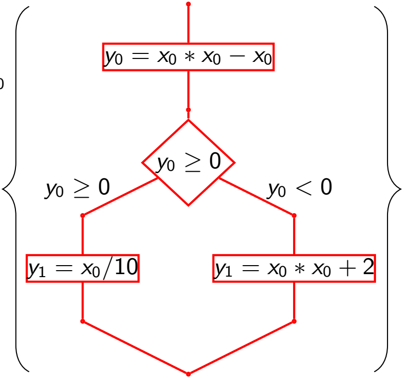
Example 1: our Constraint Programming approach

$P_0: D_{x_0} = [0, 10] \quad D_{y_0} = [-10, 90] \quad \mathbf{D_{y_1} = [0, 102]}$

$y_0 = x_0 * x_0 - x_0$
$y_0 \geq 0$
$y_1 = x_0/10$

filtering

$D^1_{x_0} = [0, 10]$
$D^1_{y_0} = [0, 90]$
$D^1_{y_1} = [0, 1]$



$y_0 = x_0 * x_0 - x_0$
$y_0 < 0$
$y_1 = x_0 * x_0 + 2$

filtering

$D^2_{x_0} = [0, 1.026]$
$D^2_{y_0} = [-0.257, 0]$
$\mathbf{D^2_{y_1} = [2, 3.027]}$

$P_6: D^3_{y_1} = D^1_{y_1} \cup D^2_{y_1} = \mathbf{[0, 3.027]}$

Filtering techniques

- FPCS: 3B(w)-consistency over the floats
  - Projection functions for floats
  - Handling of rounding modes
  - Handling of x86 architecture specifics

- RealPaver: Hull & Box-consistency over the reals
  - Reliable approximations of continuous solution sets
  - Correctly rounded interval methods and constraint satisfaction techniques

Experiments: refining AI approximations

Fluctuat : state-of-the-art AI analyzer for estimating rounding errors and their propagation using zonotopes

	Fluctuat (AI)		**rAiCp (AI + CP)**	
	Domain	Time	Domain	Time
quadratic$_1$ x_0	$[-\infty, \infty]$	0.13 s	$\mathbf{[-\infty, 0]}$	0.39 s
quadratic$_1$ x_1	$[-\infty, \infty]$	0.13 s	$\mathbf{[-8.125, \infty]}$	0.39 s
quadratic$_2$ x_0	$[-2e6, 0]$	0.13 s	$[-1e6, 0]$	0.39 s
quadratic$_2$ x_1	$[-1e6, 0]$	0.13 s	$\mathbf{[-3906, 0]}$	0.39 s
sinus7	$[-1.009, 1.009]$	0.12 s	$[-0.853, 0.852]$	0.22 s
rump	$[-1.2e37, 2e37]$	0.13 s	$[-1.2e37, 2e37]$	0.22 s
sqrt$_1$	$[2.116, 2.354]$	0.13 s	$[2.121, 2.347]$	0.81 s
sqrt$_2$	$[-\infty, \infty]$	0.2 s	$\mathbf{[2.232, 3.168]}$	1.59 s
bigLoop	$[-\infty, \infty]$	0.15 s	$\mathbf{[0, 10]}$	0.7 s
Total		**1.25 s**		**5.1 s**

Experiments: eliminating false alarms

CDFL: state-of-the-art program analyzer for proving the absence of runtime errors in program with floating-point computations

	rAiCp	**Fluctuat**	**CDFL**
False alarms	0	11	0
Total time	40.55 s	18.37 s	208.99 s

Computed on the 55 benchs from CDFL paper (TACAS'12)

Conclusion

Abstract Interpretation

+ Good **scaling** capabilities
+ Handling of **linear** expressions
− Loss of accuracy

CP framework

+ Good **refutation** capabilities
+ Handling of **nonlinear** expressions
− Scalability

AI + CP framework:

+ Efficient computation of good domain approximations