Modular Static Analysis with Zonotopes

Eric Goubault, Sylvie Putot and Franck Védrine
LMeASI, CEA LIST
Sept 11, SAS 2012, Deauville, France

Motivations

Context: validate numerical programs

- Check/infer invariant properties both in floating-point and real number semantics
- Very refined and complex (costly) properties: stability, decomposition of numerical errors etc.

Some success already with our abstract interpreter FLUCTUAT (see FMICS'07, DASIA'09, FMICS'09), mostly on $\leq$ 50KLoC programs

Motivations

Long term goal for this talk:

- Static analysis of large control systems (e.g. primary flight computers, numerical protection for nuclear plants...), designed in a modular manner (e.g. generated by SCADE)
- Want to improve scalability of FLUCTUAT

Long term goal for this talk:

- Static analysis of large control systems (e.g. primary flight computers, numerical protection for nuclear plants...), designed in a modular manner (e.g. generated by SCADE)
- Want to improve scalability of FLUCTUAT

A modular analysis:

- Of classical inspiration (Cousot&Cousot 77, Sharir&Pnueli 81)
- Particular to the abstract domain of zonotopes / affine sets (SAS'06, CAV'09, CAV'10, VMCAI'11)

Summary-based algorithm for the zonotopic abstract domain

Outline of the talk

- A zonotopic abstract domain
- Summary-based modular analysis
- Experiments

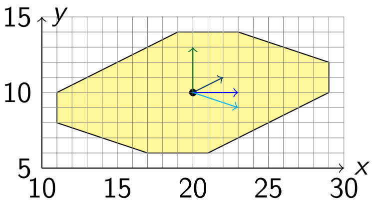
Zonotopic abstraction based on Affine Arithmetic (Stolfi 93)

- Affine form $\hat{x}$ for variable x:

$$\hat{x} = x_0 + x_1\varepsilon_1 + \ldots + x_n\varepsilon_n,$$

  where $x_i \in \mathbb{R}$ and the ε_i are independent symbolic variables called *noise symbols*, with unknown value in $[-1, 1]$.
- Sharing ε_i between variables expresses *implicit dependency*: concretization as a zonotope

$$\begin{array}{rcllll} \hat{x} & = & 20 & -4\varepsilon_1 & & +2\varepsilon_3 & +3\varepsilon_4 \\ \hat{y} & = & 10 & -2\varepsilon_1 & +\varepsilon_2 & & -\varepsilon_4 \end{array}$$



Arithmetic operations

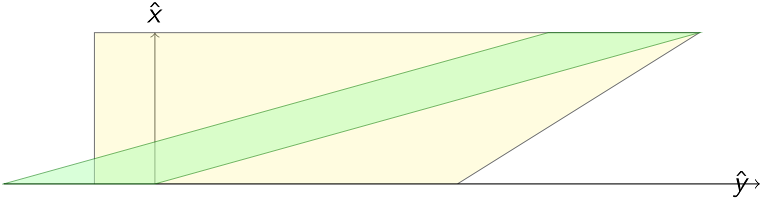
- *Assignment* of a variable x whose real value is given in a range $[a, b]$ introduces a noise symbol ε_i :
$$\hat{x} = \frac{(a+b)}{2} + \frac{(b-a)}{2}\varepsilon_i.$$
  - functional abstraction: link to the inputs via the noise symbols, allowing sensitivity analysis and worst case generation
- *Addition* is computed componentwise (no new noise symbol):
$$\hat{x} + \hat{y} = (x_0 + y_0) + (x_1 + y_1)\varepsilon_1 + \ldots + (x_n + y_n)\varepsilon_n$$
- *Multiplication* : we select an approximate linear form, the approximation error creates a new noise term :
$$\hat{x} \times \hat{y} = x_0 y_0 + \sum_{i=1}^{n}(x_i y_0 + y_i x_0)\varepsilon_i + R(x, y)\varepsilon_{n+1}.$$
- *Non linear operations* : approximate linear form (Taylor expansion), new noise term for the approximation error

Zonotopes define input-output relations

- We distinguish two kinds of noise symbols:
  - input noise symbols (ε_i) express sensitivity to the inputs
  - perturbation noise symbols (η_j) express uncertainty due to the analysis (non affine operations, join)
- Partial order:
  - geometric inclusion on zonotopes augmented by the input noise symbols
  - functional order: preserves input-output relations
  - weaker structure than lattice: no least upper bound (join) or greatest lower bound (meet)
- Relational abstraction for functions in the sense of [Jeannet, Gopan and Reps 2005]

A simple example

```
real x = [0,10];
real y = x*x - x;
```



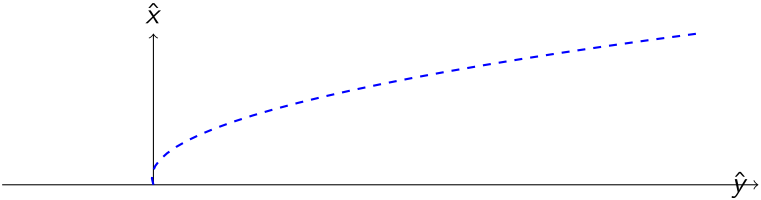
Zonotope (green) is

$$\begin{aligned} x &= 5 + 5\varepsilon_1 \\ y &= (5 + 5\varepsilon_1)(5 + 5\varepsilon_1) - 5 - 5\varepsilon_1 \\ &= 20 + 45\varepsilon_1 + 25\varepsilon_1^2 = 20 + 45\varepsilon_1 + 25\,(0.5 + 0.5\eta_1) \\ &= 32.5 + 45\varepsilon_1 + 12.5\eta_1 \end{aligned}$$

Polyhedron: $-x + 10 \geq 0;\ y + 10 \geq 0;\ x \geq 0;\ 4x - y + 50 \geq 0$

Functional interpretation of this example

```
real x = [0,10];
real y = x*x - x;
```

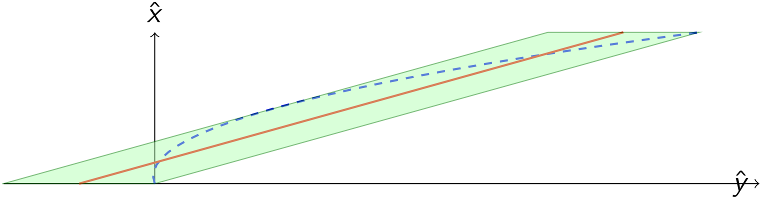


Abstraction of function $x \to y = x^2 - x$ as

$$y = 32.5 + 45\varepsilon_1 + 12.5\eta_1$$

Functional interpretation of this example

```
real x = [0,10];
real y = x*x - x;
```



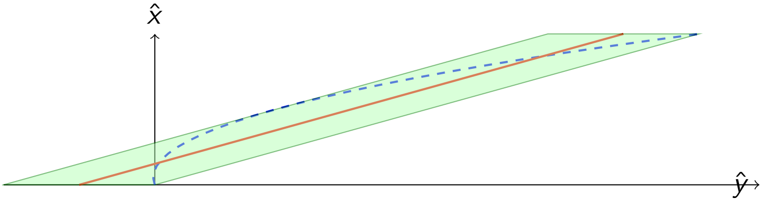
Abstraction of function $x \to y = x^2 - x$ as

$$\begin{aligned} y &= 32.5 + 45\varepsilon_1 + 12.5\eta_1 \\ &= -12.5 + 9x + 12.5\eta_1 \end{aligned}$$

(since $x = 5 + 5\varepsilon_1$)

Functional interpretation of this example

```
real x = [0,10];
real y = x*x - x;
```



Abstraction of function $x \to y = x^2 - x$ as

$$\begin{aligned} y &= 32.5 + 45\varepsilon_1 + 12.5\eta_1 \\ &= -12.5 + 9x + 12.5\eta_1 \end{aligned}$$

- Almost modular by construction!
- *But valid only for inputs in [0,10]* $\Rightarrow$ *partition of contexts though a summary-based algorithm.*

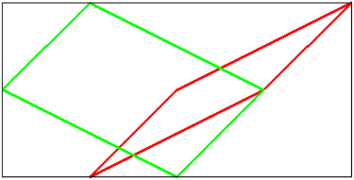
A componentwise upper bound operator

Keep "minimal common dependencies":

$$z_i = \underset{x_i \wedge y_i \leq r \leq x_i \vee y_i}{\text{argmin}} |r|, \ \forall i \geq 1$$

$$\begin{aligned} \hat{x} &= 3 + \varepsilon_1 + 2\varepsilon_2 \\ \hat{y} &= 1 - 2\varepsilon_1 + \varepsilon_2 \\ \hat{u} &= \varepsilon_1 + \varepsilon_2 \end{aligned}$$

$\hat{z} = \hat{x} \cup \hat{y} = 2 + 0\varepsilon_1 + 1\varepsilon_2 + 3\eta_1$



$\hat{x}$, $\hat{y}$ functions of $\hat{u}$

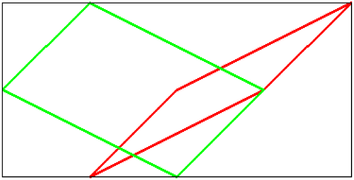
A componentwise upper bound operator

For each var, the concretization is the interval union of the concretizations ($\gamma(\hat{z}) = [-2, 6]$) :

$$\begin{array}{ll} i) & z_0 = mid(\gamma(\hat{x}) \cup \gamma(\hat{y})) \\ ii) & \beta^z = \sup \gamma(\hat{x}) \cup \gamma(\hat{y}) - z_0 - \|\hat{z}\|_1 \end{array}$$

$$\begin{aligned} \hat{x} &= 3 + \varepsilon_1 + 2\varepsilon_2 \\ \hat{y} &= 1 - 2\varepsilon_1 + \varepsilon_2 \\ \hat{u} &= \varepsilon_1 + \varepsilon_2 \end{aligned}$$

$$\hat{z} = \hat{x} \cup \hat{y} = 2 + 0\varepsilon_1 + 1\varepsilon_2 + 3\eta_1$$



$\hat{x}$, $\hat{y}$ functions of $\hat{u}$

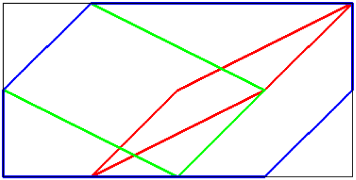
A componentwise upper bound operator

For each var, the concretization is the interval union of the concretizations ($\gamma(\hat{z}) = [-2, 6]$) :

$$\begin{array}{ll} i) & z_0 = mid(\gamma(\hat{x}) \cup \gamma(\hat{y})) \\ ii) & \beta^z = \sup \gamma(\hat{x}) \cup \gamma(\hat{y}) - z_0 - \|\hat{z}\|_1 \end{array}$$

$$\begin{aligned} \hat{x} &= 3 + \varepsilon_1 + 2\varepsilon_2 \\ \hat{y} &= 1 - 2\varepsilon_1 + \varepsilon_2 \\ \hat{u} &= \varepsilon_1 + \varepsilon_2 \end{aligned}$$

$$\hat{z} = \hat{x} \cup \hat{y} = 2{+}0\varepsilon_1{+}1\varepsilon_2{+}3\eta_1$$



$\hat{x}$, $\hat{y}$ and $\hat{z}$ functions of $\hat{u}$

A componentwise upper bound operator

Keep "minimal common dependencies":

$$z_i = \underset{x_i \wedge y_i \leq r \leq x_i \vee y_i}{\operatorname{argmin}} |r|, \ \forall i \geq 1$$

For each var, the concretization is the interval union of the concretizations ($\gamma(\hat{z}) = [-2, 6]$) :

$$\begin{array}{ll} i) & z_0 = mid(\gamma(\hat{x}) \cup \gamma(\hat{y})) \\ ii) & \beta^z = \sup \gamma(\hat{x}) \cup \gamma(\hat{y}) - z_0 - \|\hat{z}\|_1 \end{array}$$

See NSAD'12 for an improved join operator

Outline of the talk

- A zonotopic abstract domain
- Summary-based modular analysis
- Experiments

Summary-based analysis

A summary is a pair of zonotopes (I, O):

- Summary of program functions
- I abstracts the calling context, O the output
- Both zonotopes abstract functions of the inputs of the program (linear form of the ε_i)
- Output can be instantiated for a particular calling context $C \subseteq I$ (constraints on the noise symbols that define a restriction of the function)

A case of symbolic relational separate analysis as in [Cousot Cousot 2002, Modular static program analysis]

Interpretation of function call $f(C)$, given calling context C, and function summary $S_f = (I, O)$:

if $!(C \leq I)$ **then**
  $I \leftarrow I \sqcup C$ // Join
  $S_f \leftarrow (I, [\![f]\!](I))$ // New summary
end if
return $[\![I == C]\!]O$ // Summary instantiation

Interpretation of function call $f(C)$, given calling context C, and function summary $S_f = (I, O)$:

if $!(C \leq I)$ **then**
  $I \leftarrow I \sqcup C$ // Join
  $S_f \leftarrow (I, [\![f]\!](I))$ // New summary
end if
return $[\![I == C]\!]O$ // Summary instantiation

Join: $I \leftarrow I \sqcup C$

Join calling context C and summary input I, using the join operator on zonotopes

Interpretation of function call $f(C)$, given calling context C, and function summary $S_f = (I, O)$:

if $!(C \leq I)$ **then**
$I \leftarrow I \sqcup C$ // Join
$S_f \leftarrow (I, [\![f]\!](I))$ // New summary
end if
return $[\![I == C]\!]O$ // Summary instantiation

New summary: $S_f \leftarrow (I, [\![f]\!](I))$

Interpret function call f with calling context I, using the zonotopic abstract domain, and create a new summary

Algorithm

Interpretation of function call $f(C)$, given calling context C, and function summary $S_f = (I, O)$:

if $!(C \leq I)$ **then**

$I \leftarrow I \sqcup C$ // Join

$S_f \leftarrow (I, [\![f]\!](I))$ // New summary

end if

return $[\![I == C]\!]O$ // Summary instantiation

Summary instantiation: $[\![I == C]\!]O$

$I == C$ generate constraints on noise symbols that allow to substitute some perturbation noise symbols in O, and thus instantiate the output of the summary O

Summary instantiation $[\![I == C]\!]O$

- Starts with a summary (I, O) of a function f, and a less general input context C $(C \leq I)$
- Outputs $O' = [\![I == C]\!]O$ less general output context than O $(O' \leq O)$ which still overapproximates $[\![f]\!]C$

- Starts with a summary (I, O) of a function f, and a less general input context C $(C \leq I)$
- Outputs $O' = [\![I == C]\!] O$ less general output context than O $(O' \leq O)$ which still overapproximates $[\![f]\!] C$

Principle:

- Write the constraint $I == C$ in the space of noise symbols:

$$(c_{0i}^I - c_{0i}^C) + \sum_{r=1}^{n} (c_{ri}^I - c_{ri}^C)\varepsilon_r + \sum_{r=1}^{m} (p_{ri}^I - p_{ri}^C)\eta_r = 0 \ (i = 1, \ldots, p)$$

- We derive relations of the form (by Gauss elimination):

$$\eta_{k_{i+1}} = R_i(\eta_{k_i}, \ldots, \eta_1, \varepsilon_n, \ldots, \varepsilon_1) \ (i = 0, \ldots, r-1)$$

- We eliminate $\eta_{k_1}, \ldots, \eta_{k_r}$ in O using the relations above

- Starts with a summary (I, O) of a function f, and a less general input context C $(C \leq I)$
- Outputs $O' = [\![I == C]\!] O$ less general output context than O $(O' \leq O)$ which still overapproximates $[\![f]\!] C$

Principle:

- Write the constraint $I == C$ in the space of noise symbols:

$$(c_{0i}^I - c_{0i}^C) + \sum_{r=1}^{n} (c_{ri}^I - c_{ri}^C)\varepsilon_r + \sum_{r=1}^{m} (p_{ri}^I - p_{ri}^C)\eta_r = 0 \ (i = 1, \ldots, p)$$

- We derive relations of the form (by Gauss elimination):

$$\eta_{k_{i+1}} = R_i(\eta_{k_i}, \ldots, \eta_1, \varepsilon_n, \ldots, \varepsilon_1) \ (i = 0, \ldots, r-1)$$

- We eliminate $\eta_{k_1}, \ldots, \eta_{k_r}$ in O using the relations above

In practice, with the componentwise join: all relations already in row-echelon form!

Example

```
(0) mult(real a, real b) { return a*(b-2); }
(1) x := [-1,1];   // x = eps_1
(2) real y1 = mult(x+1, x);
(3) real y2 = mult(x, 2*x);
```

if $!(C \leq I)$ **then**
$\quad I \leftarrow I \sqcup C$
$\quad S_f \leftarrow (I, [\![f]\!](I))$
end if
return $[\![I == C]\!]O$

$S_f = \emptyset$
$(2) : I_1 = C_1 = (\varepsilon_1 + 1, \varepsilon_1)$

```
(0) mult(real a, real b) { return a*(b-2); }
(1) x := [-1,1];   // x = eps_1
(2) real y1 = mult(x+1, x);
(3) real y2 = mult(x, 2*x);
```

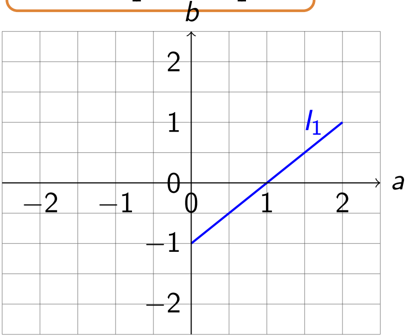
if $!(C \leq I)$ **then**
  $I \leftarrow I \sqcup C$
  $S_f \leftarrow (I, [\![f]\!](I))$
end if
return $[\![I == C]\!]O$

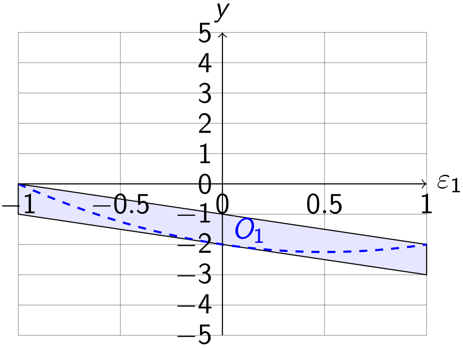
$(2) : I_1 = C_1 = (\varepsilon_1 + 1, \varepsilon_1)$
$O_1 = [\![mult]\!](I_1) = -1.5 - \varepsilon_1 + 0.5\eta_1$
$S = (I_1, O_1)$

```
(2) real y1 = mult(x+1, x);
(3) real y2 = mult(x, 2*x);
```

if $!(C \leq I)$ **then**
$\quad I \leftarrow I \sqcup C$
$\quad S_f \leftarrow (I, [\![f]\!](I))$
end if
return $[\![I == C]\!]O$

$(2) : I_1 = (\varepsilon_1 + 1, \varepsilon_1)$
$O_1 = (-1.5 - \varepsilon_1 + 0.5\eta_1), S = (I_1, O_1)$

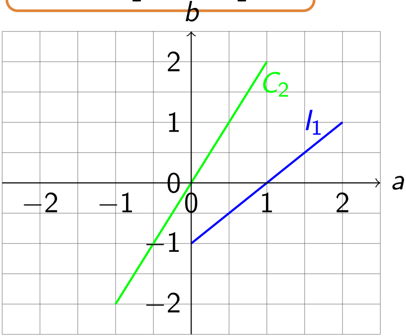


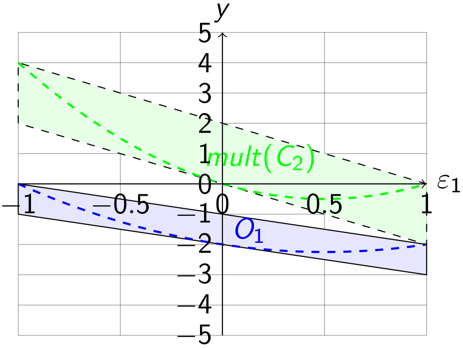


```
(2)  real  y1  =  mult(x+1,  x);
(3)  real  y2  =  mult(x,  2*x);
```

if $!(C \leq I)$ **then**
$\quad I \leftarrow I \sqcup C$
$\quad S_f \leftarrow (I, [\![f]\!](I))$
end if
return $[\![I == C]\!]O$

$(2) : I_1 = (\varepsilon_1 + 1, \varepsilon_1)$
$(3) : C_2 = (\varepsilon_1, 2\varepsilon_1)$

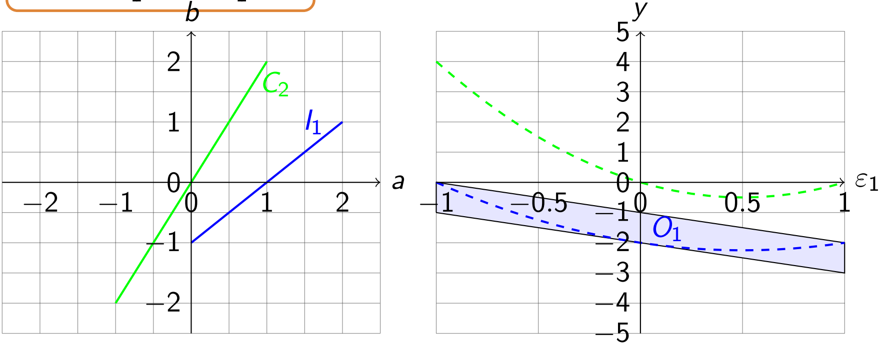




```
(2) real y1 = mult(x+1, x);
(3) real y2 = mult(x, 2*x);
```

if $!(C \leq I)$ **then**
  $I \leftarrow I \sqcup C$
  $S_f \leftarrow (I, [\![f]\!](I))$
end if
return $[\![I == C]\!]O$

$(2) : I_1 = (\varepsilon_1 + 1, \varepsilon_1)$
$(3) : C_2 = (\varepsilon_1, 2\varepsilon_1)$



```
(2) real y1 = mult(x+1, x);
(3) real y2 = mult(x, 2*x);
```

if $!(C \leq I)$ **then**

$I \leftarrow I \sqcup C$

$S_f \leftarrow (I, [\![f]\!](I))$

end if

return $[\![I == C]\!]O$

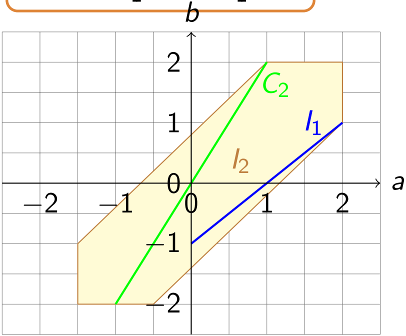
$I_1 = (\varepsilon_1 + 1, \varepsilon_1)$

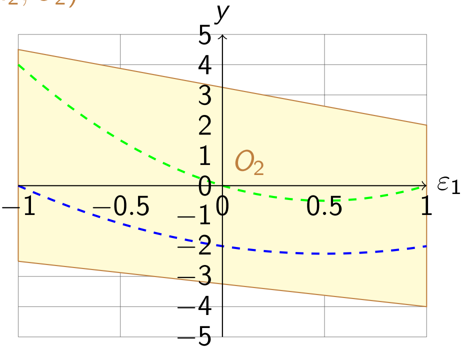
$C_2 = (\varepsilon_1, 2\varepsilon_1)$

$I_2 = I_1 \sqcup C_2 = (\frac{1}{2} + \varepsilon_1 + \frac{1}{2}\eta_2, \frac{3}{2}\varepsilon_1 + \frac{1}{2}\eta_3)$

$O_2 = -\frac{1}{4} - \frac{5}{4}\varepsilon_1 - \eta_2 + \frac{1}{4}\eta_3 + \frac{9}{4}\eta_4$

$S = (I_2, O_2)$





Instantiation to 2nd calling context C_2

return $[\![I_2 == C_2]\!] O_2$

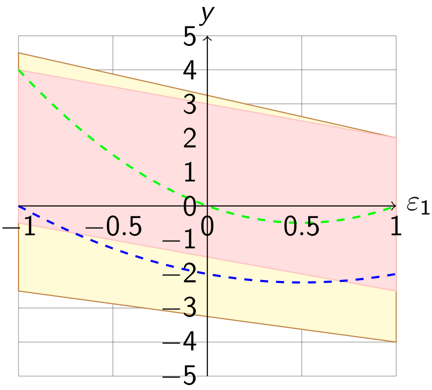
$$C_2 = (\varepsilon_1, 2\varepsilon_1)$$
$$I_2 = (\tfrac{1}{2} + \varepsilon_1 + \tfrac{1}{2}\eta_2, \tfrac{3}{2}\varepsilon_1 + \tfrac{1}{2}\eta_3)$$

$I_2 == C_2$ yields constraints

$$\varepsilon_1 = \tfrac{1}{2} + \varepsilon_1 + \tfrac{1}{2}\eta_2 \Rightarrow \eta_2 = -1$$
$$2\varepsilon_1 = \tfrac{3}{2}\varepsilon_1 + \tfrac{1}{2}\eta_3 \Rightarrow \eta_3 = \varepsilon_1$$

Substitute η_2 and η_3 in
$O_2 = -\frac{1}{4} - \frac{5}{4}\varepsilon_1 - \eta_2 + \frac{1}{4}\eta_3 + \frac{9}{4}\eta_4$
gives

$$[\![I_2 == C_2]\!] O_2 = \frac{3}{4} - \varepsilon_1 + \frac{9}{4}\eta_4.$$



Instantiation to 2nd calling context C_2

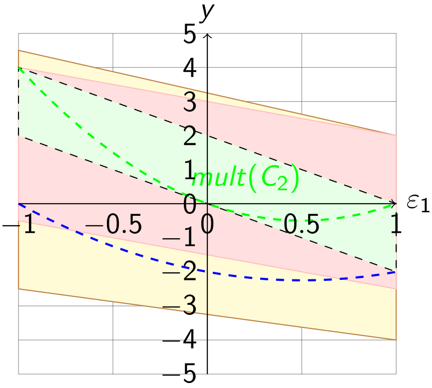
$$\text{return } [\![I_2 == C_2]\!] O_2$$

$C_2 = (\varepsilon_1, 2\varepsilon_1)$
$I_2 = (\frac{1}{2} + \varepsilon_1 + \frac{1}{2}\eta_2, \frac{3}{2}\varepsilon_1 + \frac{1}{2}\eta_3)$

$I_2 == C_2$ yields constraints

$\varepsilon_1 = \frac{1}{2} + \varepsilon_1 + \frac{1}{2}\eta_2 \Rightarrow \eta_2 = -1$
$2\varepsilon_1 = \frac{3}{2}\varepsilon_1 + \frac{1}{2}\eta_3 \Rightarrow \eta_3 = \varepsilon_1$

Substitute η_2 and η_3 in
$O_2 = -\frac{1}{4} - \frac{5}{4}\varepsilon_1 - \eta_2 + \frac{1}{4}\eta_3 + \frac{9}{4}\eta_4$
gives
$[\![I_2 == C_2]\!] O_2 = \dfrac{3}{4} - \varepsilon_1 + \dfrac{9}{4}\eta_4.$



Outline of the talk

- A zonotopic abstract domain
- Summary-based modular analysis
- Experiments

Benchmarks

- `sincos`: sine and cosine approximations by high order polynomials:
  - iterates on 65 distinct overlapping intervals some computation f involving polynomial approximations of sine and sqrt
  - function f is analyzed twice, and instantiated 63 times
- `img_filter`: edge detection on a small image:
- `filter`: recursive order 2 filter with uncertain parameters

	example	*sincos*	*img_filter*	*filter*
	#lines/#vars	208/135	194/55	96/19
Non-modular	time (s)	3.84	11.8	49
Modular	time(s)	0.79	1.7	10
	instantiations	63/65	18/20	6/8
Ratio	time gain	4.9	6.9	4.9
	accuracy loss	2.23	1	1.27

- `sincos`: sine and cosine approximations
- `img_filter`: edge detection on a small image:
  - iteration of Sobel filters and normalized Gaussian smoothing on a 1D signal of dimension 20
  - called on on 20 classes of signals
  - functions are linear: the summary is exact (accuracy loss=1)
  - only 2 full analyzes of f and 18 instantiations
  - instantiation is fast: time gain of 6.9
- `filter`: recursive order 2 filter with uncertain parameters

	example	*sincos*	*img_filter*	*filter*
	#lines/#vars	208/135	194/55	96/19
Non-modular	time (s)	3.84	11.8	49
Modular	time(s)	0.79	1.7	10
	instantiations	63/65	18/20	6/8
Ratio	time gain	4.9	6.9	4.9
	accuracy loss	2.23	1	1.27

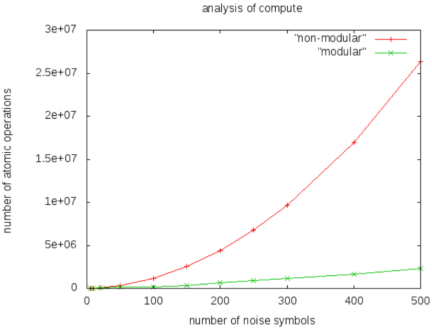
- `sincos`: sine and cosine approximations
- `img_filter`: edge detection on a small image:
- `filter`: recursive order 2 filter with uncertain parameters
  - slight non-linearity: small accuracy loss

	example	*sincos*	*img_filter*	*filter*
	#lines/#vars	208/135	194/55	96/19
Non-modular	time (s)	3.84	11.8	49
Modular	time(s)	0.79	1.7	10
	instantiations	63/65	18/20	6/8
Ratio	time gain	4.9	6.9	4.9
	accuracy loss	2.23	1	1.27

Performance

Possibly many noise symbols (though artificial here)

```
real x := [-1,1];
real y1, y2;
for(int i=0;i<100;i++)
{
   y1 = mult(x+1,x);
   y2 = mult(x, 2*x);
   x /= 2;
}
```



$\#op = f(\#symbols)$

- Instantiations of linear cost (vs quadratic monolithic analysis here)
- Accuracy: $y1 \in [-6.5, 2.5]$ (modular) vs $[-3, 0]$ (non-modular), and $y2 \in [-4, 4]$ vs $[-2, 4]$.

- Zonotopic abstraction of input-output relationships naturally well suited to modular analysis
- Simple and efficient algorithm: promising results with our prototype on small examples
- Experiments on large industrial test cases:
  - first partly automated experiments (see paper for details)
  - achieved using the interactive version of Fluctuat (see TAPAS'12 presentation)
  - fully automated experiments require combination with modular alias analyses
  - another difficulty is to determine the right level of modularity (many possible levels in some instrumentation software)
  - heuristics to create several summaries (as hinted in the paper) to be developed/experimented