MPRI

Static Analysis of Digital Filters

ESOP 2004, NSAD 2005

Jérôme Feret

Laboratoire d'Informatique de l'École Normale Supérieure

INRIA, ÉNS, CNRS

http://www.di.ens.fr/∼feret

January 2019.

Overview

Context

We want to prove run time error absence, in critical embedded software.
Filter behaviour is implemented at the software level, using hardware floating point numbers.

Full certification requires special care about these filters.

Issues

- Detection: to locate filter resets and filter iterations.

- Invariant inference: we are not interested in functional properties.

  We seek precise bounds on the output, using information inferred about the input.

  (Linear invariants do not yield accurate bounds).

- To take into account floating-point rounding:
  - in the semantics,
  - when implementing the abstract domain.

Overview

1. Introduction
2. Case studies
3. Concrete semantics
4. Generic aproximation
5. Filter domains
6. Post fixpoint inference of contracting function in floating-point arithmetics
7. Basic simplified filters
8. Higher order simplified filters
9. Bounded expansion
10. Filter detection
11. Conclusion

The high band-pass filter

$V \in \mathbb{R};$

$E_1 := 0;\ S := 0;$

while $(V \geq 0)$ {

  $V \in \mathbb{R};\ T \in \mathbb{R};$

  $E_0 \in [-1;1];$

  if $(T \geq 0)$ $\{S := 0\}$

  else $\{S := 0.999 \times S + E_0 - E_1\}$

  $E_1 := E_0;$

}

Interval approximation (simplified)

The analyzer infers the following sound counterpart $\mathbb{F}^\sharp$:

$$\mathbb{F}^\sharp(X) = \left\{0.999s + e_0 + e_1 \mid s \in X,\ e_0, e_1 \in [-1;1]\right\}$$

to the loop body.

Abstract iteration

1. The analyzer starts iterating $\mathbb{F}^\sharp$:
   $\mathbb{F}^\sharp(\{0\}) = [-2;2]$,
   $\mathbb{F}^\sharp([-2;2]) = [-3.998;3.998]$,
   $\ldots$;
2. then it widens the iterates:
   $\mathbb{F}^\sharp([-10;10]) \not\subseteq [-10;10]$,
   $\mathbb{F}^\sharp([-100;100]) \not\subseteq [-100;100]$,
   $\ldots$;
3. until it discovers a stable threshold:
   $\mathbb{F}^\sharp([-10000;10000]) = [-9992;9992]$;
4. finally, it keeps iterating to refine the solution:
   $\mathbb{F}^\sharp([-9992;9992]) = [-9984.008;9984.008]$.

Driving the analysis

Theorem 1 (High band-pass filter (history-insensitive))
Let $D \geq 0$, $m \geq 0$, a, X *and* Z *be real numbers such that:*

1. $|X| \leq D$;
2. $aX - m \leq Z \leq aX + m$;

then we have:

1. $|Z| \leq |a|D + m$;
2. $\left[|a| < 1 \text{ and } D \geq \frac{m}{1-|a|} \right] \implies |Z| \leq D$. $\square$

Theorem 1 implies that 2000 can be used as a widening threshold.

History sensitive approximation

Theorem 2 (High band-pass filter (history-sensitive version))
Let $\alpha \in [\frac{1}{2}; 1[$, i *and* $m > 0$ *be real numbers.*
Let E_n *be a real number sequence, such that* $\forall k \in \mathbb{N},\ E_k \in [-m; m]$.
Let S_n *be the following sequence:*

$$\begin{cases} S_0 = i \\ S_{n+1} = \alpha S_n + E_{n+1} - E_n. \end{cases}$$

We have:

1. $S_n = \alpha^n i + E_n - \alpha^n E_0 + \sum_{l=1}^{n-1} (\alpha - 1)\alpha^{l-1} E_{n-l}$
2. $|S_n| \leq |\alpha|^n |i| + (1 + |\alpha|^n + |1 - \alpha^{n-1}|)m$;
3. $|S_n| \leq 2m + |i|$. $\square$

Theorem 2 implies that 2 is a sound bound on $|S|$.

The second order filter

$V \in \mathbb{R};$

$E_1 := 0;\ E_2 := 0;\ S_0 := 0;\ S_1 := 0;\ S_2 := 0;$

while $(V \geq 0)$ $\{$

$\quad V \in \mathbb{R};\ T \in \mathbb{R};$

$\quad E_0 \in [-1;1];$

$\quad$ if $(T \geq 0)$ $\{S_0 := E_0;\ S_1 := E_0;\ E_1 := E_0\}$

$\quad$ else $\{S_0 := 1.5 \times S_1 - 0.7 \times S_2$

$$+ 0.5 \times E_0 - 0.7 \times E_1 + 0.4 \times E_2\};$$

$\quad E_2 := E_1;\ E_1 := E_0;$

$\quad S_2 := S_1;\ S_1 := S_0$

$\}$

Quadratic constraints

Theorem 3 (second order filter (history insensitive))
Let a, b, $K \geq 0$, $m \geq 0$, X, Y, Z *be real numbers such that:*

1. $a^2 + 4b < 0$,
2. $X^2 - aXY - bY^2 \leq K$,
3. $aX + bY - m \leq Z \leq aX + bY + m$.

We have:

1. $Z^2 - aZX - bX^2 \leq \left(\sqrt{-bK} + m\right)^2$;
2. $\begin{cases} \sqrt{-b} < 1 \\ K \geq \left(\frac{m}{1-\sqrt{-b}}\right)^2 \end{cases} \implies Z^2 - aZX - bX^2 \leq K$.

Proof

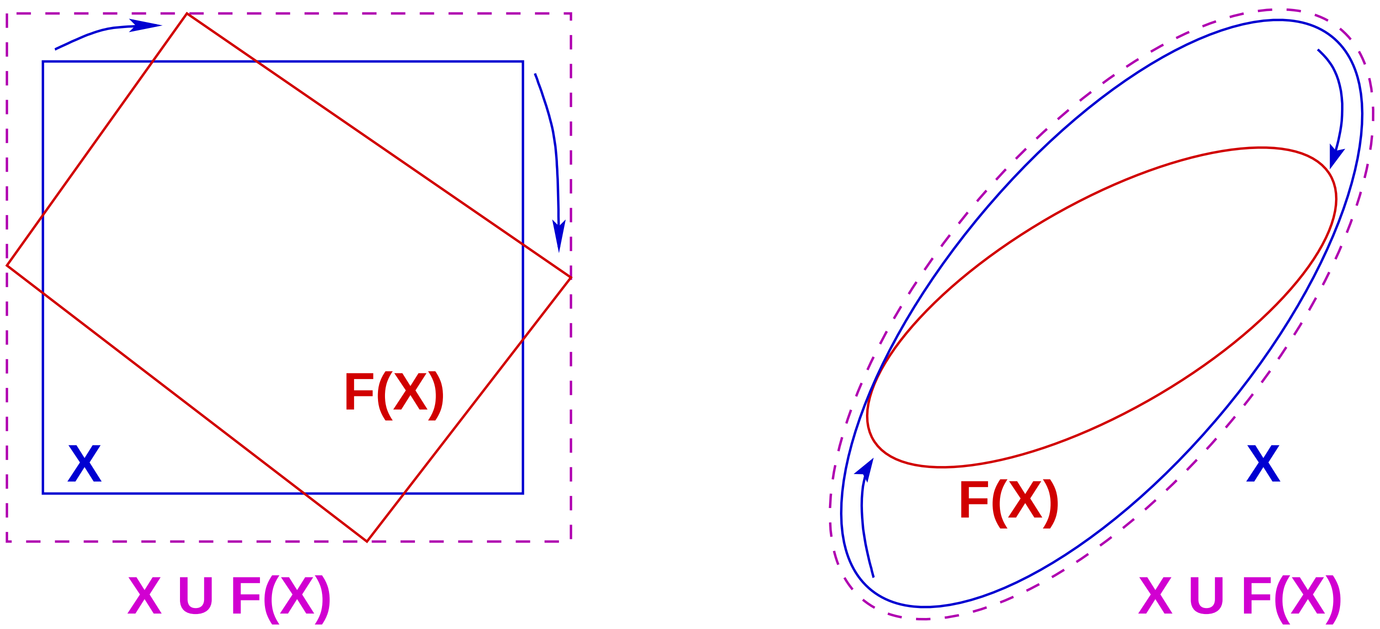
We define $Q(X,Y) \triangleq X^2 - aXY - bY^2$ and $Z \triangleq aX + bY + e$.
We have:

$$\begin{array}{rcll} Q(Z,X) & = & (aX+bY+e)^2 - a(aX+bY+e)X - bX^2 & \\ Q(Z,X) & = & -b(X^2 - aXY - bY^2) + e(aX+2bY+e) & \\ Q(Z,X) & = & -bQ(X,Y) + e(aX+2bY+e) & \\ Q(Z,X) & \leq & -bQ(X,Y) + m|aX+2bY| + m^2 & \text{since } |e| \leq m \end{array}$$

$$\begin{array}{rcll} (aX+2bY)^2 & = & -4b(\frac{a^2}{-4b}X^2 - aXY - bY^2) & \\ (aX+2bY)^2 & \leq & -4bQ(X,Y) & \text{since } a^2+4b<0 \\ |aX+2bY| & \leq & 2\sqrt{-bQ(X,Y)} & \end{array}$$

$$Q(Z,X) \leq \left(\sqrt{-bQ(X,Y)} + m\right)^2$$

Linear versus quadratic invariants



Second order filter approximation

1. without relational domain,
   we cannot limit $|S_2|$;
2. with quadratic constraints (history insensitive abstraction),
   we can infer that $|S_2| < 22.111$;
3. by formally expanding the output as a sum of all previous inputs,
   we can prove that $|S_2| < 1.41824$;

Overview

1. Introduction
2. Case studies
3. Concrete semantics
4. Generic aproximation
5. Filter domains
6. Post fixpoint inference of contracting function in floating-point arithmetics
7. Basic simplified filters
8. Higher order simplified filters
9. Bounded expansion
10. Filter detection
11. Conclusion

Syntax

Let $\mathcal{V}$ be a finite set of variables.
Let $\mathcal{I}$ be the set of real intervals (including $\mathbb{R}$).
Expressions $\mathcal{E}$ are affine forms of variables $\mathcal{V}$ with real interval coefficients:

$$E ::= I + \sum_{j \in J} I_j . V_j$$

Programs are given by the following grammar:

$$\begin{array}{rcl} P & :== & \text{skip} \\ & | & P;P \\ & | & V := E \\ & | & \text{if } (V \geq 0)\ \{P\} \text{ else } \{P\} \\ & | & \text{while } (V \geq 0)\ \{P\} \end{array}$$

Semantics

We define the semantics of a program P:

$$[\![P]\!] : (\mathcal{V} \rightarrow \mathbb{R}) \rightarrow \wp(\mathcal{V} \rightarrow \mathbb{R})$$

by induction over the syntax of P:

$[\![\text{skip}]\!](\rho) = \{\rho\}$,

$[\![P_1;P_2]\!](\rho) = \{\rho'' \mid \exists \rho' \in [\![P_1]\!](\rho),\ \rho'' \in [\![P_2]\!](\rho')\}$,

$[\![V := I + \sum_{j\in J} I_j.V_j]\!](\rho) = \left\{\rho\left[V \mapsto i + \sum_{j\in J} i_j.\rho(V_j)\right] \;\middle|\; i \in I,\ \forall j \in J, i_j \in I_j\right\}$,

$[\![\text{if } (V \geq 0)\ \{P_1\} \text{ else } \{P_2\}]\!](\rho) = \begin{cases} [\![P_1]\!](\rho) & \text{if } \rho(V) \geq 0 \\ [\![P_2]\!](\rho) & \text{otherwise,} \end{cases}$

$[\![\text{while } (V \geq 0)\ \{P\}]\!](\rho) = \{\rho' \in \mathit{Inv} \mid \rho'(V) < 0\}$
where $\mathit{Inv} = \text{lfp}\,(X \mapsto \{\rho\} \cup \{\rho'' \mid \exists \rho' \in X,\ \rho'(V) \geq 0 \text{ and } \rho'' \in [\![P]\!](\rho')\})$.

Overview

1. Introduction
2. Case studies
3. Concrete semantics
4. Generic aproximation
5. Filter domains
6. Post fixpoint inference of contracting function in floating-point arithmetics
7. Basic simplified filters
8. Higher order simplified filters
9. Bounded expansion
10. Filter detection
11. Conclusion

Abstract domain

An abstract domain $\text{ENV}^\sharp$ is a set of environment properties.

A concretization map γ relates each property to the set of its solutions:

$$\gamma : \text{ENV}^\sharp \to \wp(\mathcal{V} \to \mathbb{R}).$$

Some primitives simulate concrete computation steps in the abstract:

- an abstract control path merge $\sqcup$;
- an abstract guard GUARD and an abstract assignment ASSIGN;
- an abstract least fixpoint $\text{lfp}^\sharp$ operator, which maps sound counterpart $f^\sharp$ to monotonic function f, to an abstraction of the least fixpoint of f.
  $\text{lfp}^\sharp$ is defined using extrapolation operators $(\perp, \nabla, \Delta)$.
  Soundness follows from the monotony of the concrete semantics.

Abstract semantics

$[\![\text{skip}]\!]^\sharp(a) = a$

$[\![P_1;P_2]\!]^\sharp(\rho^\sharp) = [\![P_2]\!]^\sharp([\![P_1]\!]^\sharp(\rho^\sharp))$

$[\![V := E]\!]^\sharp(a) = \text{ASSIGN}(V, E, a)$

$$[\![\text{if } (V \geq 0)\ \{P_1\} \text{ else } \{P_2\}]\!]^\sharp(a) = a_1 \sqcup a_2,$$
$$\text{with } \begin{cases} a_1 = [\![P_1]\!]^\sharp(\text{GUARD}(V, [0; +\infty[, a)) \\ a_2 = [\![P_2]\!]^\sharp(\text{GUARD}(V,]-\infty; 0[, a)) \end{cases}$$

$$[\![\text{while } (V \geq 0)\ \{P\}]\!]^\sharp(a) = \text{GUARD}(V,]-\infty; 0[, \mathit{Inv}^\sharp)$$
$$\text{where } \mathit{Inv}^\sharp = \text{lfp}^\sharp \left(X \mapsto a \sqcup [\![P]\!]^\sharp(\text{GUARD}(V, [0; +\infty[, X))\right)$$

Soundness

We prove by induction over the syntax:

Theorem 4 (Soundness) *For any program P, environment ρ, abstract element a, we have:*

$$\rho \in \gamma(a) \implies [\![P]\!](\rho) \subseteq \gamma\left([\![P]\!]^{\sharp}(a)\right).$$

□

Extrapolation operators

- iteration basis: $\bot \in \text{ENV}^\sharp$
- a widening operator $\triangledown$ such that:
  1. $\triangledown \in (\text{ENV}^\sharp \times \text{ENV}^\sharp) \to \text{ENV}^\sharp$,
  2. $\forall a, b \in \text{ENV}^\sharp,\ \gamma(a) \cup \gamma(b) \subseteq \gamma(a \triangledown b)$,
  3. $\forall\, (a_i) \in (\text{ENV}^\sharp)^{\mathbb{N}}$, the sequence $(a_i^\triangledown)$ defined by:
     $$a_0^\triangledown = a_0 \text{ and } a_{n+1}^\triangledown = a_n^\triangledown \triangledown a_{n+1}$$
     is eventually stationary;
- a narrowing operator $\vartriangle$ such that:
  1. $\vartriangle \in (\text{ENV}^\sharp \times \text{ENV}^\sharp) \to \text{ENV}^\sharp$,
  2. $\forall a, b \in \text{ENV}^\sharp,\ \gamma(a) \cap \gamma(b) \subseteq \gamma(a \vartriangle b)$,
  3. $(a_i) \in (\text{ENV}^\sharp)^{\mathbb{N}}$, the sequence $(a_i^\vartriangle)$ defined by:
     $$a_0^\vartriangle = a_0 \text{ and } a_{n+1}^\vartriangle = a_n^\vartriangle \vartriangle a_{n+1}$$
     is eventually stationary;

Abstract iterations

Let $f^\sharp$ be a map in $\text{ENV}^\sharp \to \text{ENV}^\sharp$.

Abstract upward-iterates:

$$\begin{cases} C_0^{\triangledown} = \bot, \\ C_{n+1}^{\triangledown} = C_n^{\triangledown} \triangledown f^\sharp(C_n^{\triangledown}), \end{cases}$$

is eventually stationary: We denote by $C_\omega^{\triangledown}$ its limit.

Abstract downward-iterates:

$$\begin{cases} D_0^{\vartriangle} = C_\omega^{\triangledown}, \\ D_{n+1}^{\vartriangle} = D_n^{\vartriangle} \vartriangle f^\sharp(D_n^{\vartriangle}), \end{cases}$$

is eventually stationary: We define $\text{lfp}^\sharp(f^\sharp)$ as this limit.

Soundness

Let f be a $\cup$-complete morphism such that:

$$\forall a \in \text{ENV}^\sharp,\ f(\gamma(a)) \subseteq \gamma(f^\sharp(a)).$$

We want to prove that $\text{lfp}(f) \subseteq \gamma(\text{lfp}^\sharp(f^\sharp))$.

We know that (kleenean iteration):

$$\forall a \in \wp(\mathcal{V} \to \mathbb{R}),\ f(a) \subseteq a \implies \text{lfp}(f) \subseteq a.$$

So, we only have to prove that:

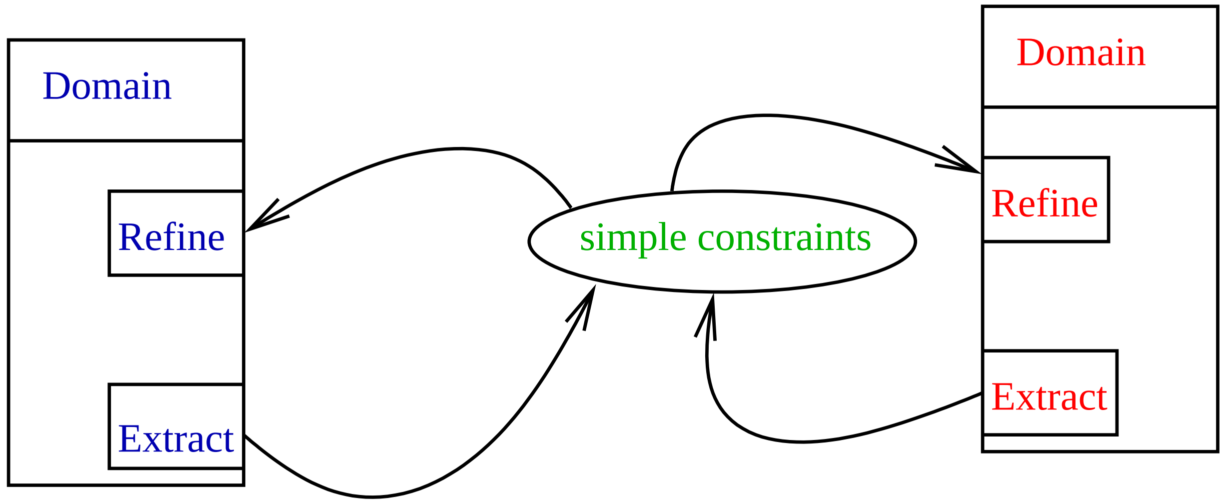
$$\exists b \in \wp(\mathcal{V} \to \mathbb{R}),\ f(b) \subseteq b \text{ and } b \subseteq \gamma(\text{lfp}^\sharp(f^\sharp)).$$

Soundness proof (continued)

1. $f(\gamma(C_\omega^\nabla)) \subseteq \gamma(C_\omega^\nabla)$ since:
   $f(\gamma(C_\omega^\nabla)) \subseteq \gamma(f^\sharp(C_\omega^\nabla))$, (soundness of $f^\sharp$)
   $\gamma(f^\sharp(C_\omega^\nabla)) \subseteq \gamma(C_\omega^\nabla \nabla f^\sharp(C_\omega^\nabla))$, (soundness of ∇)
   $C_\omega^\nabla \nabla f^\sharp(C_\omega^\nabla) = C_\omega^\nabla$, ($C_\omega^\nabla$ is a limit)
2. $\forall n \in \mathbb{N},\ \exists a \in \wp(\mathcal{V} \to \mathbb{R})$ such that $f(a) \subseteq a$ and $a \subseteq \gamma(D_n^\Delta)$:
   (a) $\gamma(D_0^\Delta) = \gamma(C_\omega^\nabla)$ and $f(\gamma(C_\omega^\nabla)) \subseteq \gamma(C_\omega^\nabla)$;
   (b) let $a \in \wp(\mathcal{V} \to \mathbb{R})$ such that $f(a) \subseteq a$ and $a \subseteq \gamma(D_n^\Delta)$,
   then
   - $f(f(a)) \subseteq f(a)$ (f is monotonic),
     $f(a) \subseteq f(\gamma(D_n^\Delta)) \subseteq \gamma(f^\sharp(D_n^\Delta))$,
   - $f(a \cap f(a)) \subseteq f(a) \cap f(f(a)) \subseteq a \cap f(a)$,
     $a \cap f(a) \subseteq \gamma(D_n^\Delta) \cap \gamma(f^\sharp(D_n^\Delta)) \subseteq \gamma(D_n^\Delta \Delta f^\sharp(D_n^\Delta)) \subseteq \gamma(D_{n+1}^\Delta)$

Approximated reduced product

Domains are refined by simple constraints computed in other domains:



Interface with other domains

We only use two kinds of simple constraints:

- $\gamma_= : \begin{cases} \wp(\mathcal{V}^2) & \to \wp(\mathcal{V} \to \mathbb{R}) \\ \mathcal{R} & \mapsto \{\rho \mid (X,Y) \in \mathcal{R} \implies \rho(X) = \rho(Y)\}; \end{cases}$
- $\gamma_{\mathcal{I}} : \begin{cases} (\mathcal{V} \to \mathcal{I}) & \to \wp(\mathcal{V} \to \mathbb{R}) \\ \rho^\sharp & \mapsto \{\rho \mid \forall X \in \mathcal{V},\ \rho(X) \in \rho^\sharp(X)\}. \end{cases}$

We can get such constraints by weakening of abstract properties:

1. $\mathsf{EQU} : \mathsf{ENV}^\sharp \to \wp(\mathcal{V}^2)$
   $\forall a \in \mathsf{ENV}^\sharp,\ \gamma(a) \subseteq \gamma_=(\mathsf{EQU}(a))$;
2. $\mathsf{RANGE} : \mathsf{ENV}^\sharp \to (\mathcal{V} \to \mathcal{I})$:
   $\forall a \in \mathsf{ENV}^\sharp,\ \gamma(a) \subseteq \gamma_{\mathcal{I}}(\mathsf{RANGE}(a))$.

We refine abstract properties by weakening range constraints:

- $\mathsf{REDUCE} : ((\mathcal{V} \to \mathcal{I}) \times \mathsf{ENV}^\sharp) \to \mathsf{ENV}^\sharp$
  $\forall a \in \mathsf{ENV}^\sharp,\ \rho^\sharp \in (\mathcal{V} \to \mathcal{I}),\ \gamma(a) \cap \gamma_{\mathcal{I}}(\rho^\sharp) \subseteq \gamma(\mathsf{REDUCE}(\rho^\sharp, a))$,

Reduction policy

We will refine abstract properties when it is necessary:

- after assignments
- after guards
- after extrapolation steps

To ensure termination, we forbid cyclic reductions after extrapolation steps: domains are ordered by the relation “is used to refine”.

Extrapolation (revisited)

We also require that:

- $\forall k \in \mathbb{N},\ \rho_1, ..., \rho_k \in (\mathcal{V} \to \mathcal{I})$, $(a_i) \in (\text{ENV}^\sharp)^{\mathbb{N}}$,
  the sequence $\left(a_i^\triangledown\right)$ defined by:

  $$a_0^\triangledown = \rho(a_0) \text{ and } a_{n+1}^\triangledown = \rho(a_n^\triangledown \triangledown a_{n+1})$$
  $$\text{with } \rho = [X \mapsto \text{REDUCE}(\rho_k, X)] \circ ... \circ [X \mapsto \text{REDUCE}(\rho_1, X)],$$

  is eventually stationary;
- $\forall k \in \mathbb{N},\ \rho_1, ..., \rho_k \in (\mathcal{V} \to \mathcal{I})$, $(a_i) \in (\text{ENV}^\sharp)^{\mathbb{N}}$,
  the sequence $\left(a_i^\vartriangle\right)$ defined by:

  $$a_0^\vartriangle = \rho(a_0) \text{ and } a_{n+1}^\vartriangle = \rho(a_n^\vartriangle \vartriangle a_{n+1}),$$
  $$\text{with } \rho = [X \mapsto \text{REDUCE}(\rho_k, X)] \circ ... \circ [X \mapsto \text{REDUCE}(\rho_1, X)],$$

  is eventually stationary.

Overview

1. Introduction
2. Case studies
3. Concrete semantics
4. Generic aproximation
5. Filter domains
6. Post fixpoint inference of contracting function in floating-point arithmetics
7. Basic simplified filters
8. Higher order simplified filters
9. Bounded expansion
10. Filter detection
11. Conclusion

Filter family

A filter class is given by:

- the number p of outputs and the number q of inputs involved in the computation of the next output;
- a (generic/symbolic) description of F with parameters;
- some conditions over these parameters

In the case of the second order filter:

- $p = 2,\ \ q = 3$;
- $F(V, W, X, Y, Z) = a \times V + b \times W + c \times X + d \times Y + e \times Z$;
- $a^2 + 4b < 0$.

Filter domain

A filter constraint is a couple in $\mathcal{T} \times \mathcal{B}$ where:

- $\mathcal{T} \in \wp_{\text{finite}}(\mathcal{V}^m \times \mathbb{R}^n)$ with:
  - m, the number of variables that are involved in the computation of the next output. m depends on the abstraction;
  - n, the number of filter parameters;
- $\mathcal{B}$ is an abstract domain encoding some “ranges”.

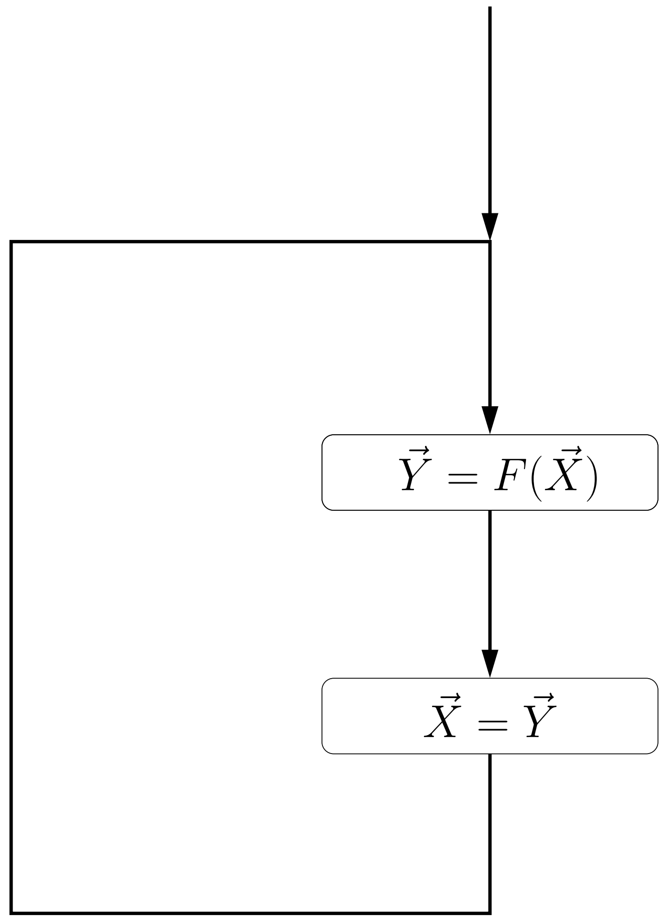
A constraint (t, d) is related to a set of environments:

$$\gamma_{\mathcal{B}} : \mathcal{T} \times \mathcal{B} \to \wp(\mathcal{V} \to \mathbb{R}).$$

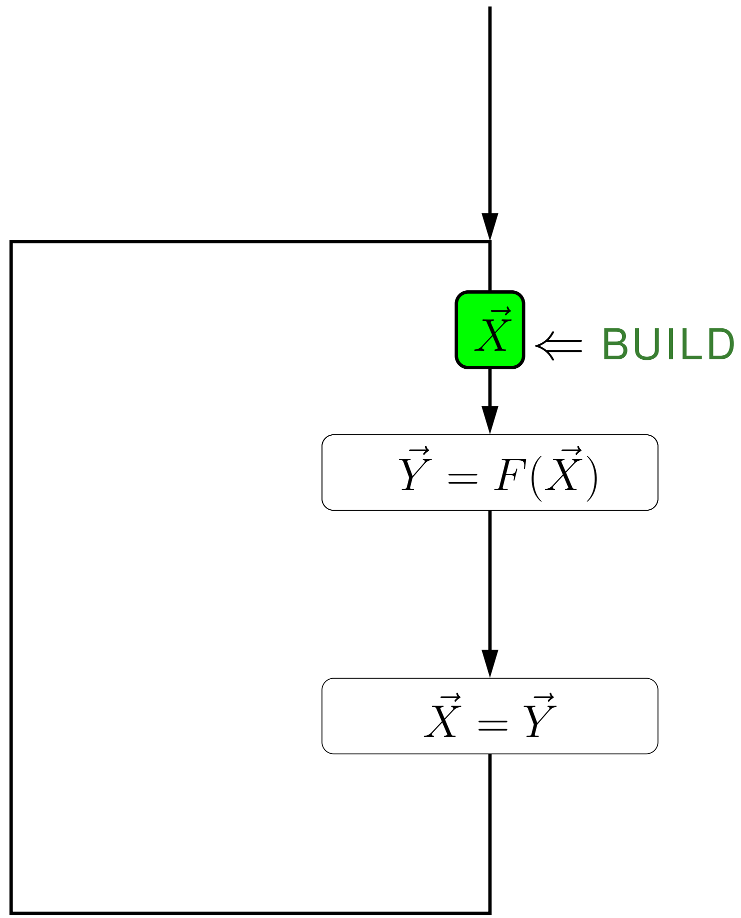
An approximation of second order filter may consist in relating:

- the last two outputs and the first two coefficients of the filter (a and b)
- to the ‘radius’ of an ellipsis.

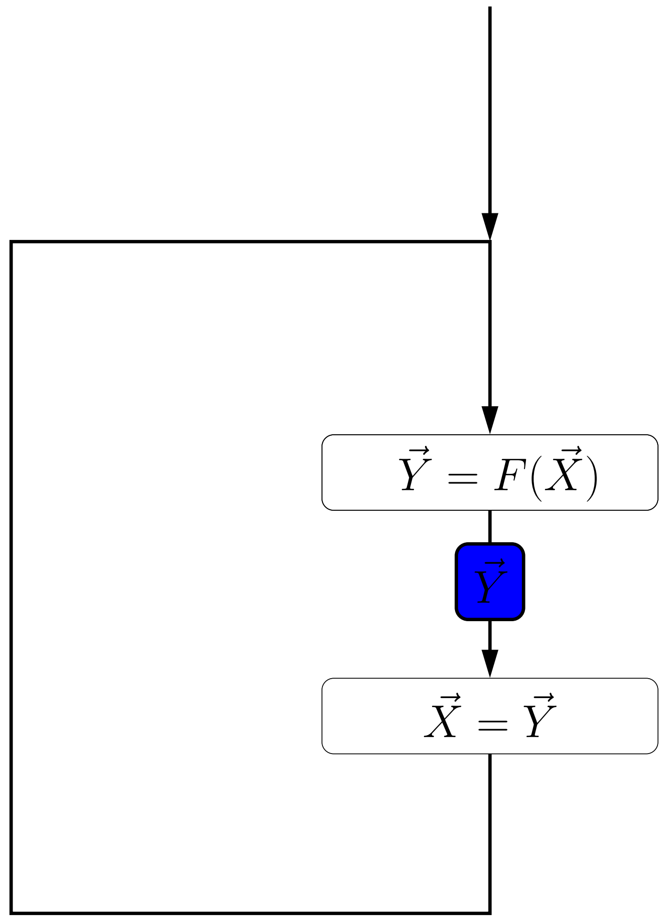
Iterations



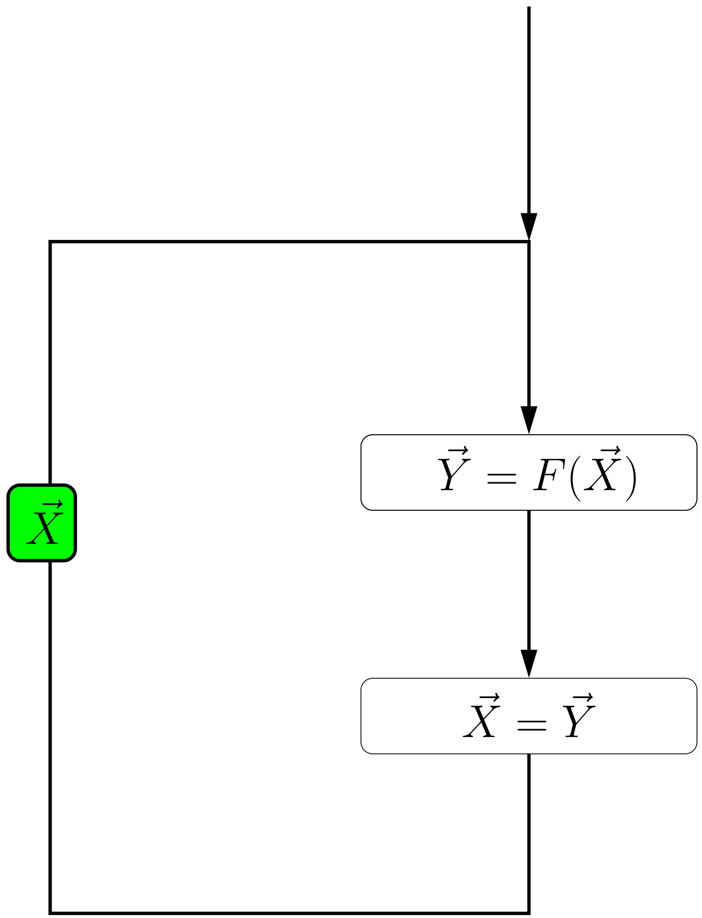
Iterations



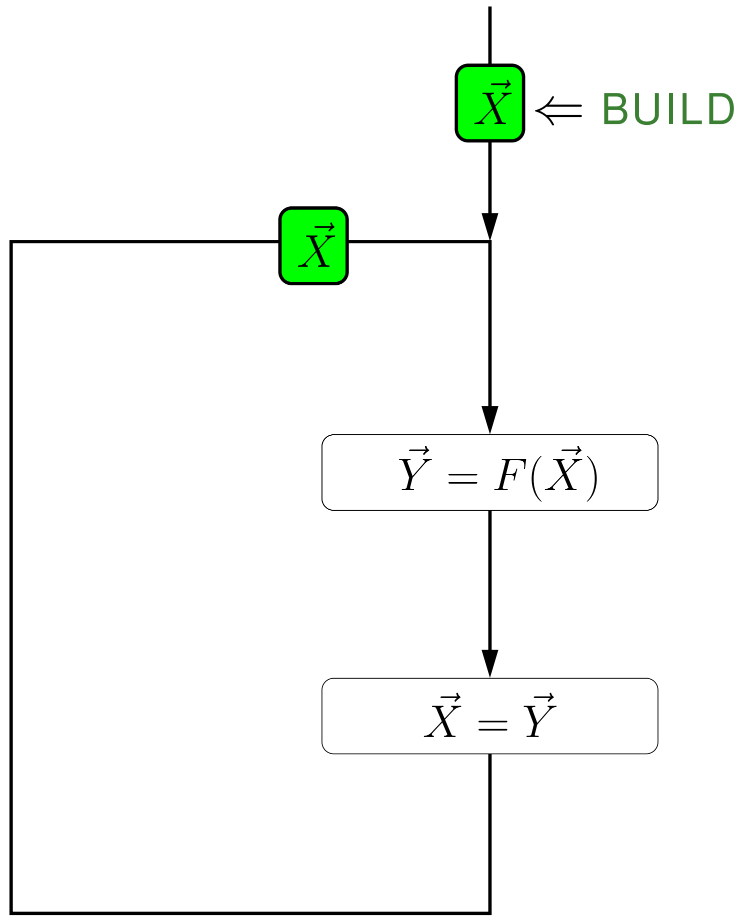
Iterations



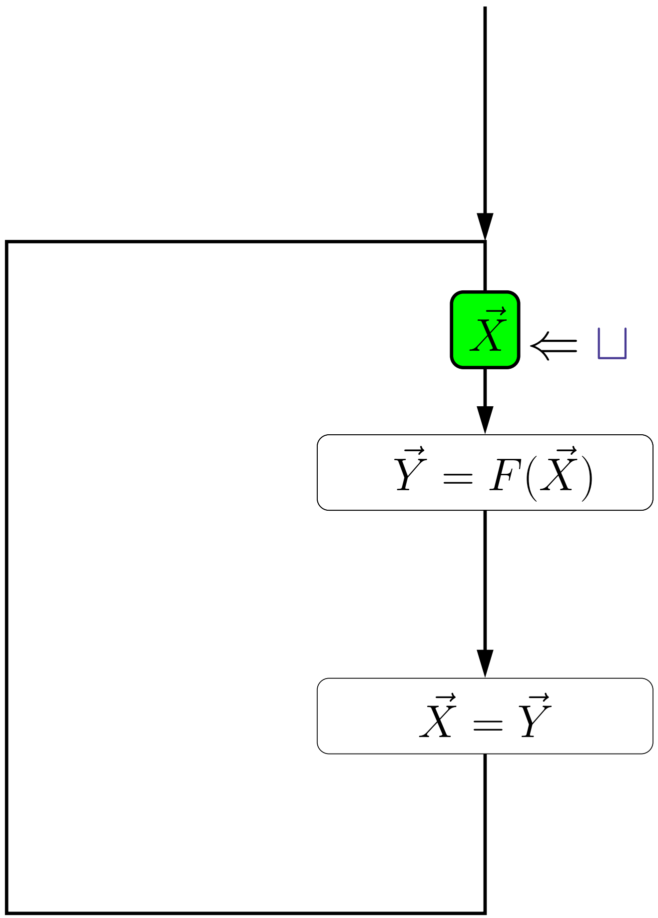
Iterations



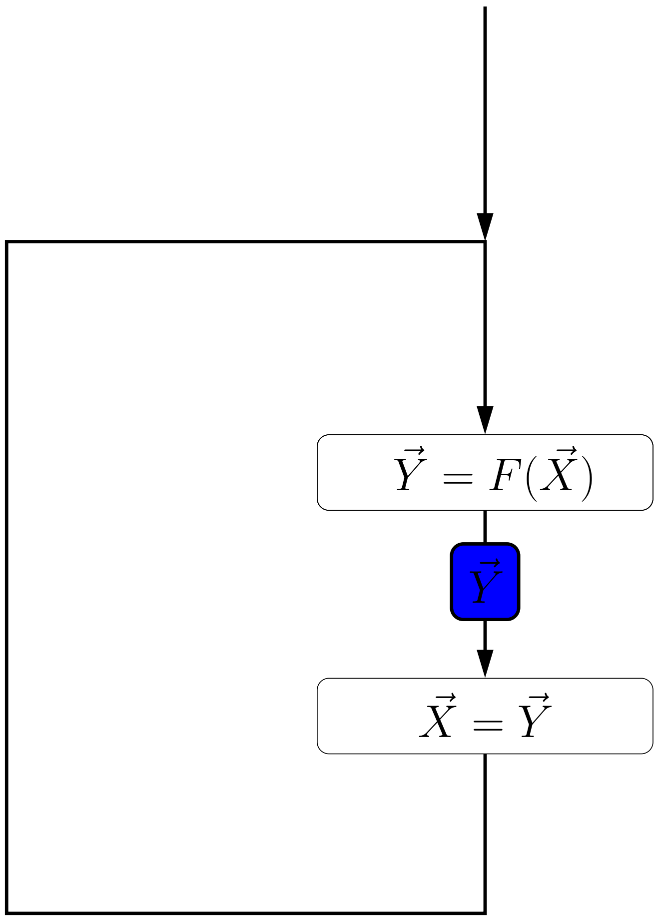
Iterations



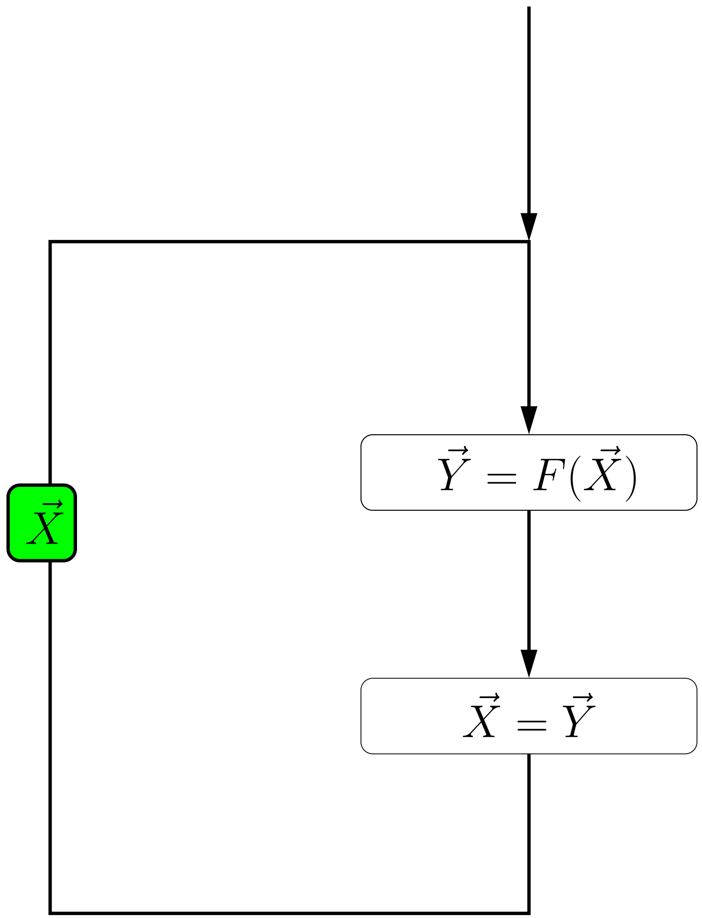
Iterations



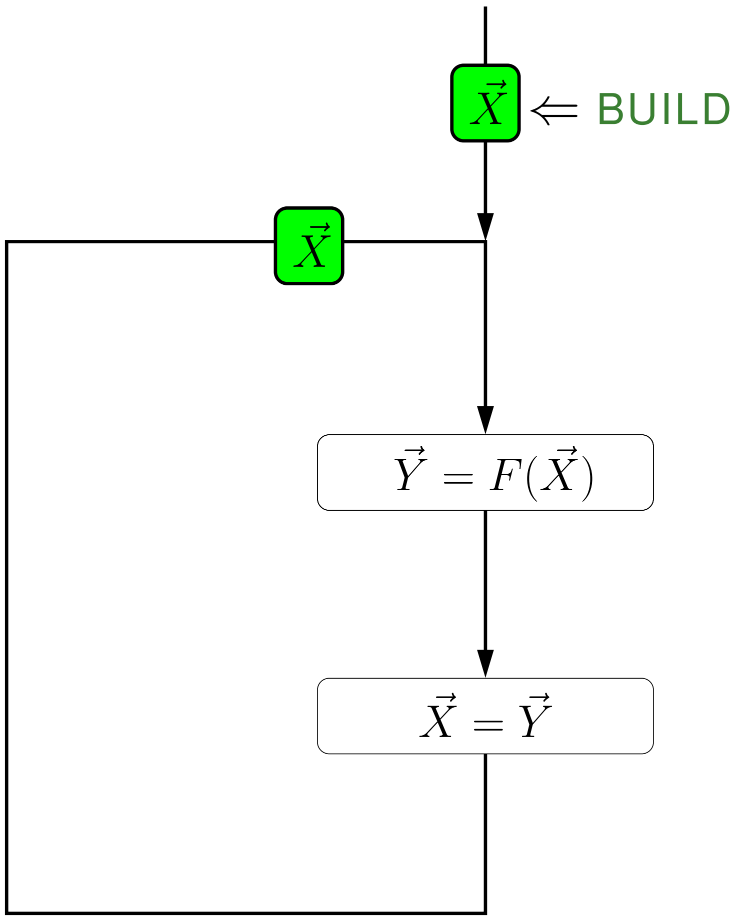
Iterations



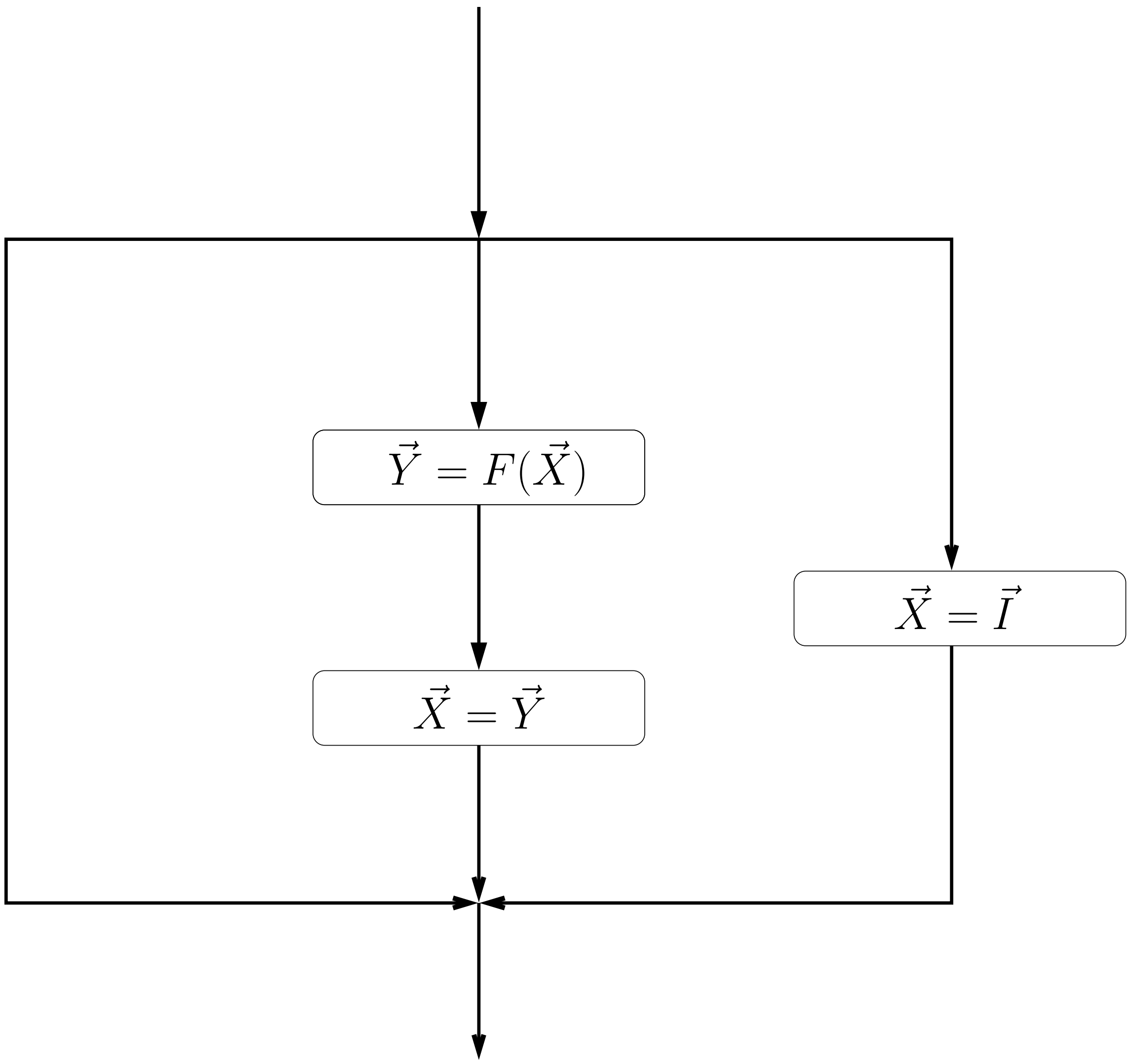
Iterations



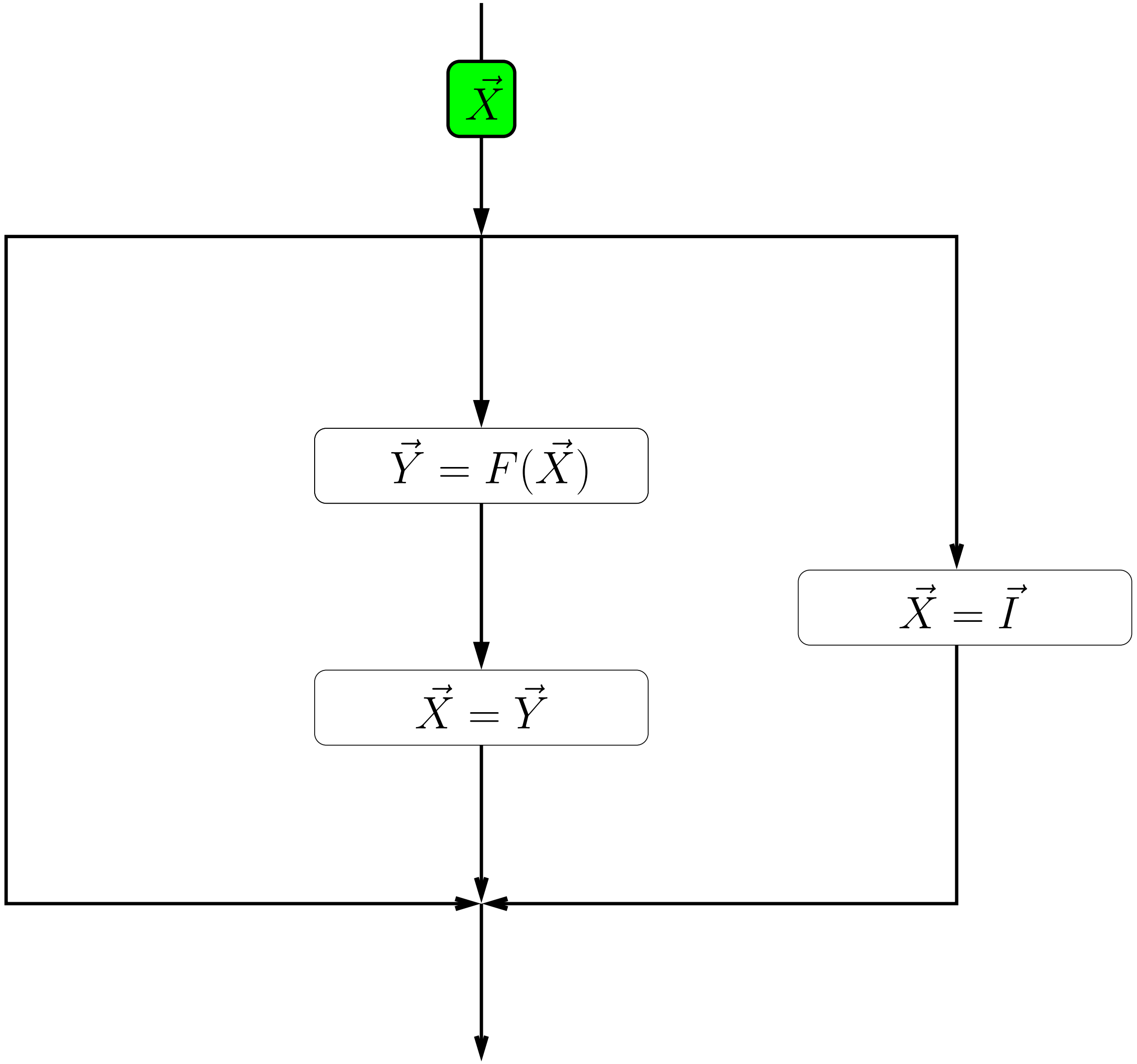
Iterations



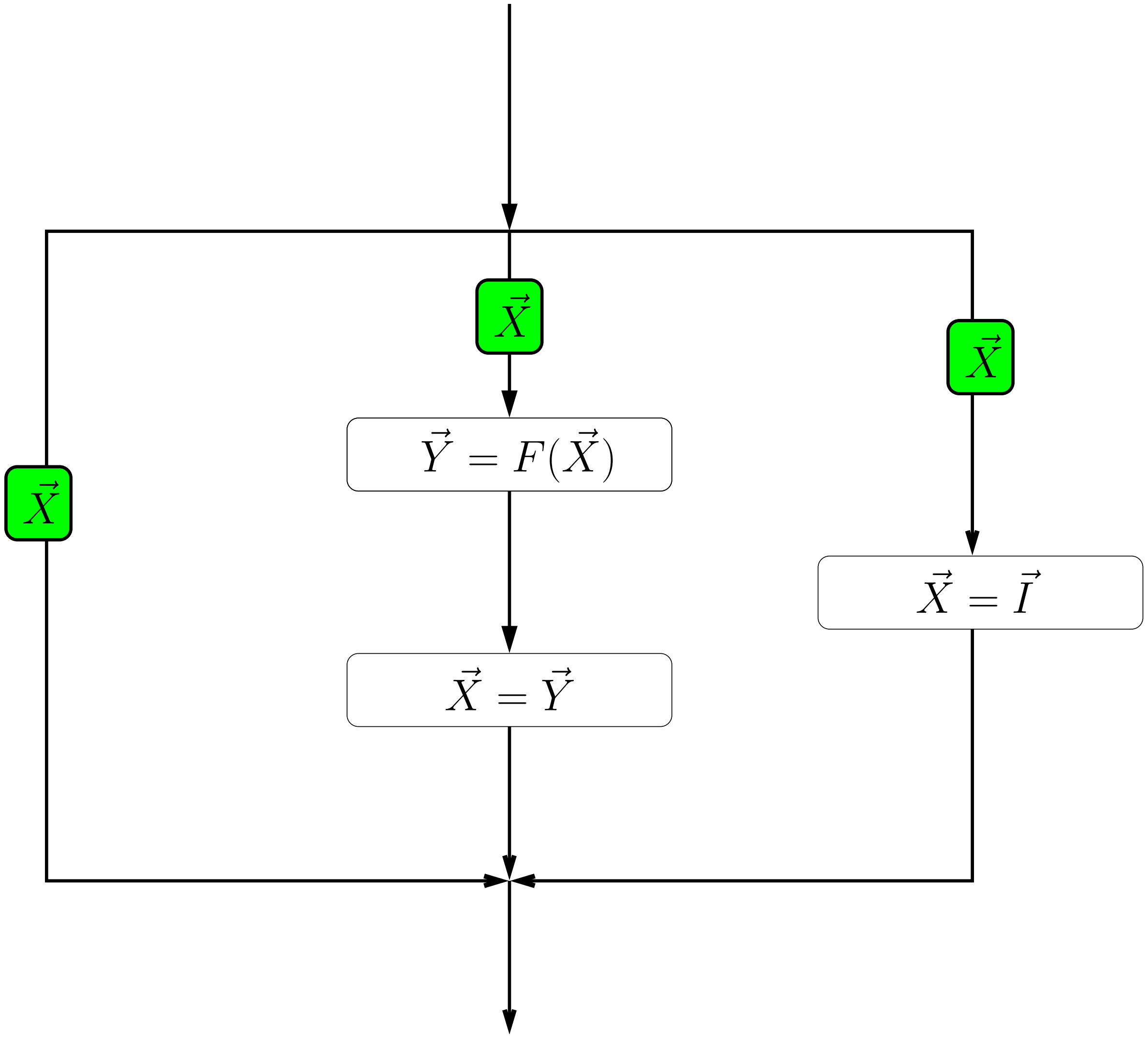
Merging computation paths



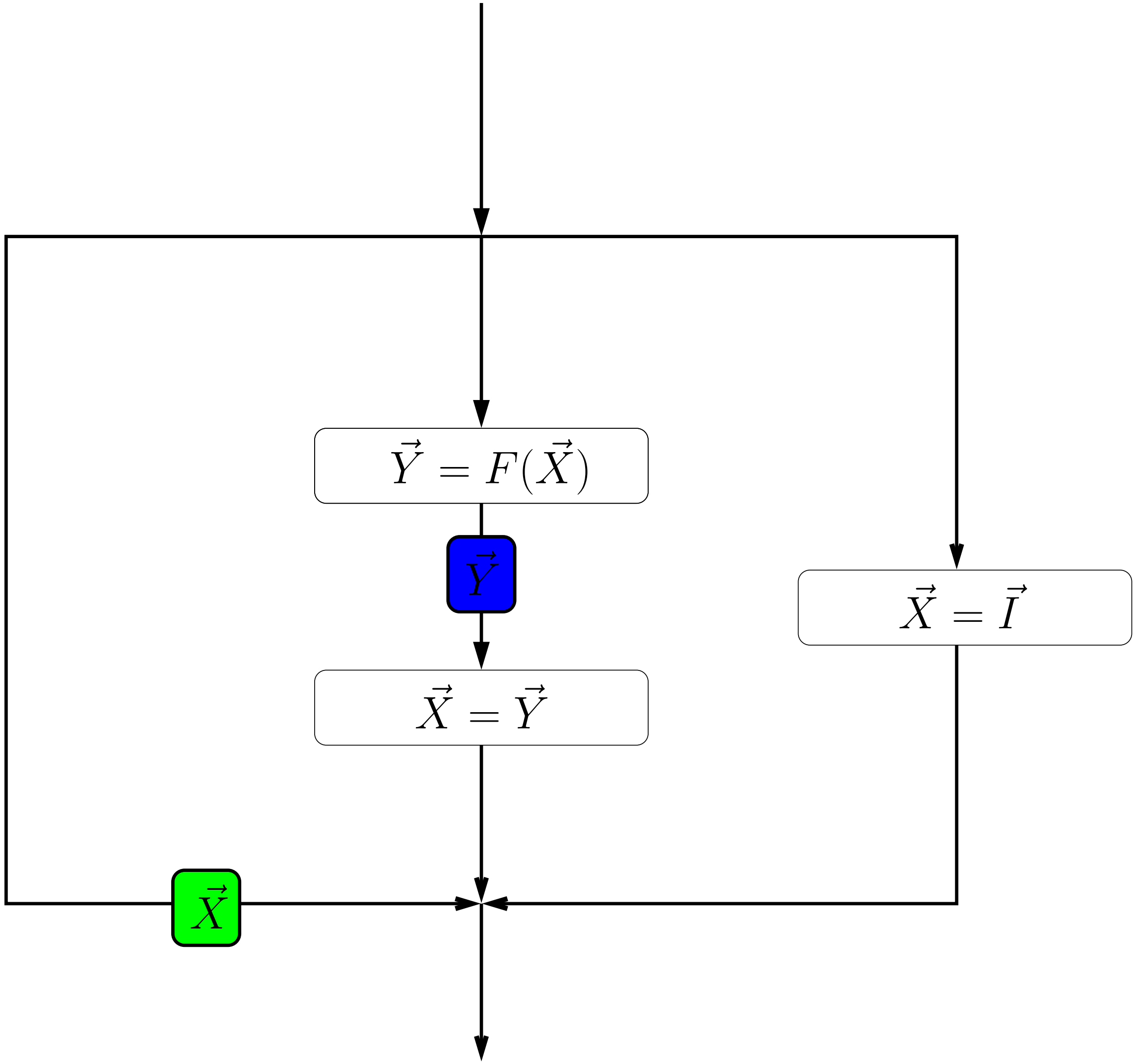
Merging computation paths



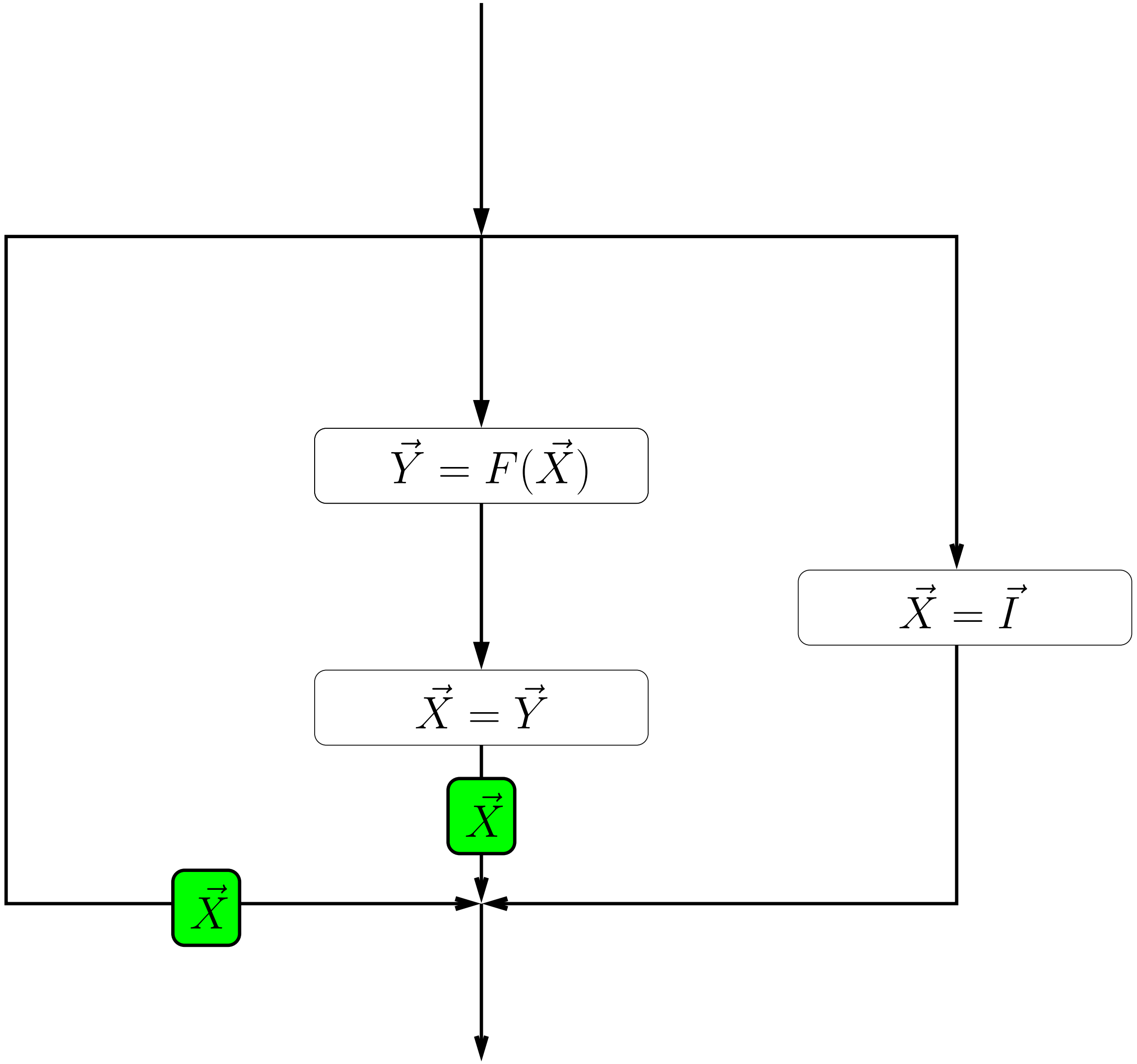
Merging computation paths



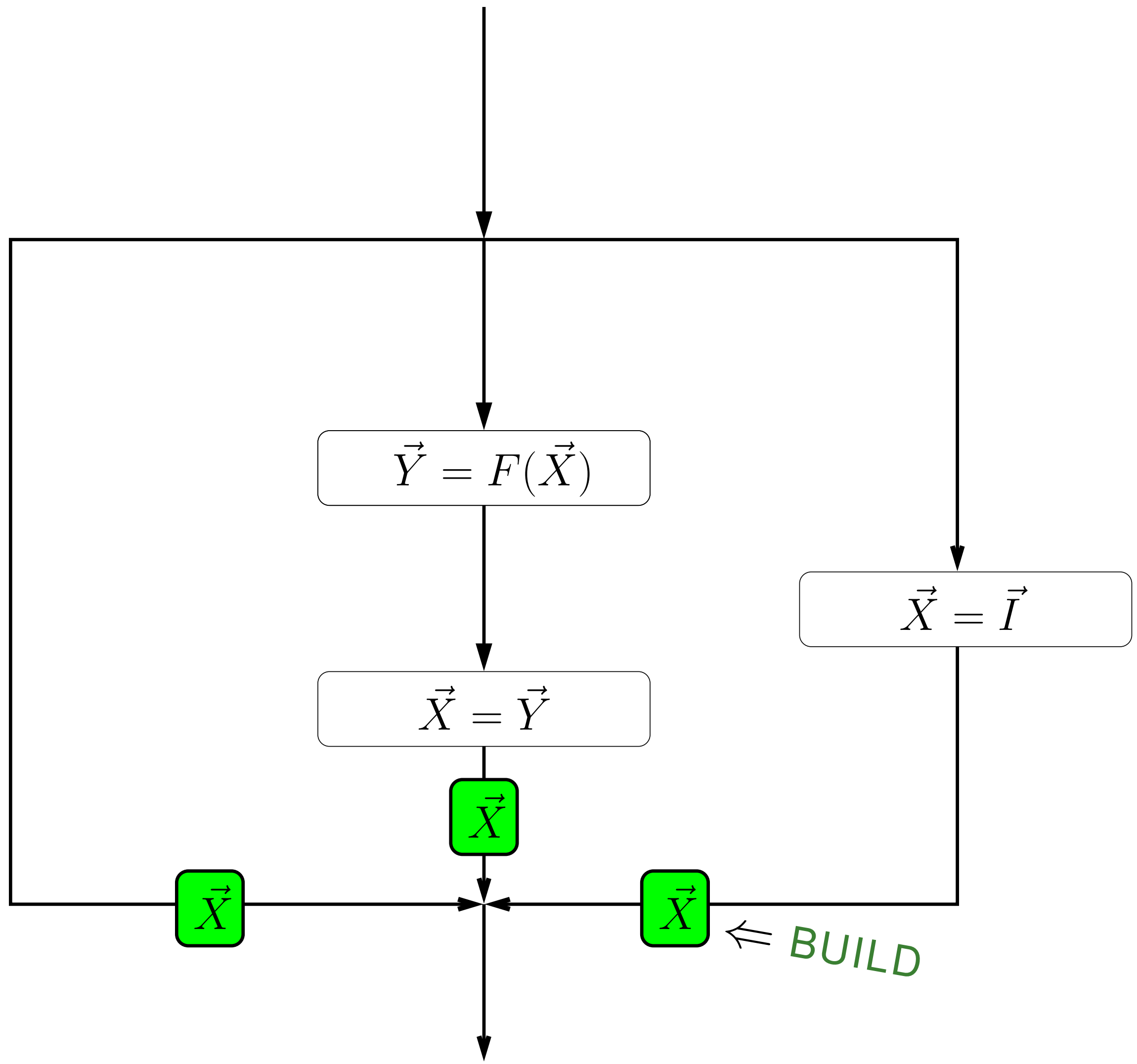
Merging computation paths



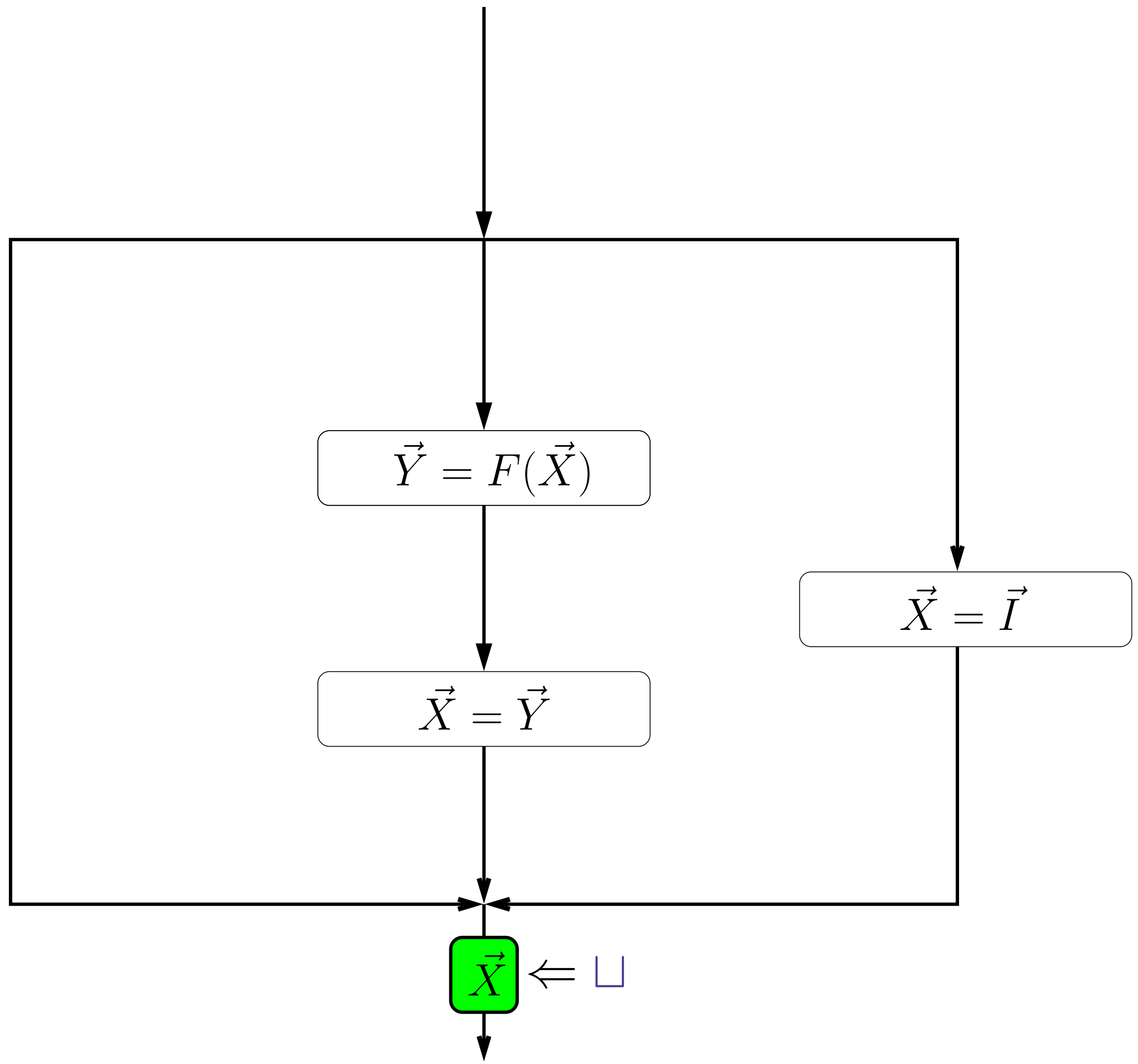
Merging computation paths



Merging computation paths



Merging computation paths



Overview

1. Introduction
2. Case studies
3. Concrete semantics
4. Generic aproximation
5. Filter domains
6. Post fixpoint inference of contracting function in floating-point arithmetics
7. Basic simplified filters
8. Higher order simplified filters
9. Bounded expansion
10. Filter detection
11. Conclusion

Floating point domain

Let:

- $\mathbb{F}$ be a finite subset of $\mathbb{R}$ closed upon opposite,
- L is a finite subset of $\mathbb{F}$;
- q, r two natural parameters for setting extrapolation strategy.

We define $\mathcal{F}_{q,r}$ as follows:

- $\mathcal{F}_{q,r} = \overline{\mathbb{F}} = \mathbb{F} \cup \{-\infty; +\infty\}$;
- $\gamma_{\overline{\mathbb{F}}} : \begin{cases} \overline{\mathbb{F}} & \mapsto \wp(\mathbb{R}) \\ a & \rightarrow \begin{cases} [-a; a] & \text{if } a \in \mathbb{F} \\ \mathbb{R} & \text{otherwise}; \end{cases} \end{cases}$
- $\lceil \_ \rceil : \begin{cases} \mathbb{R} \rightarrow \overline{\mathbb{F}} \\ x \rightarrow \mathit{min}(\{f \in \overline{\mathbb{F}} \mid f \geq x\}); \end{cases}$
- $a \nabla_{\overline{\mathbb{F}}} b = \mathit{min}(\{l \in L \cup \{a; +\infty\} \mid l \geq b\})$.

Extrapolation strategy

- Delayed widening:

$$(a_1,k_1)\nabla_{\mathcal{F}_{q,r}}(a_2,k_2)=\begin{cases}(a_1,k_1) & \text{if } a_1\geq a_2\\ (a_2,k_1+1) & \text{if } a_2>a_1 \text{ and } k_1<q\\ (a_1\nabla_{\overline{\mathbb{F}}}a_2,0) & \text{otherwise};\end{cases}$$

  Constraints are only widened when they have been unstable (not necessarily successively) q times, since their last widening.

- Bounded narrowing:

$$(a_1,k_1)\Delta_{\mathcal{F}_{q,r}}(a_2,k_2)=\begin{cases}(a_1,k_1) & \text{if } a_1\leq a_2 \text{ or } k_1\leq(-r)\\ (a_2,\mathit{min}(k_1,0)-1) & \text{if } a_2<a_1 \text{ and } k_1>(-r);\end{cases}$$

  Constraints are only narrowed r times.

Approximating contracting functions

When analyzing filter, we iterate functions f such that:

- $f : I \times \mathbb{F} \rightarrow \mathbb{F}$
- $\forall i \in I$, the map $[x \rightarrow f(i,x)]$ is contracting;
- we can compute $f_l : I \rightarrow \mathbb{F}$ such that $\forall i \in I,\ f(i, f_l(i)) \leq f_l(i)$;

where I is a set of inputs.

Since $[x \rightarrow f(i,x)]$ is contracting, we have:

- $\forall i \in I,\ \forall x \geq f_l(i),\ f(i,x) \leq x$.

Our goal

We want to find a iterating strategy which ensures:

- soundness (even if f_l is unsound)
- accuracy (if f_l is sound):
  - do not jump directly at the limit f_l: (to analyze not iterated filter, loop unrolling. . .)
  - do not jump higher than the limit when the input is constant;
  - do not jump higher than the limit in most cases.
- termination (even if the input depend on the output).

Reduced product

We use an approximation of the reduced product of two domains:
Let q,r be two natural parameters.

1. the first domain iterates f in $\mathcal{F}_{0,r}$

   $\Longrightarrow$ widened at each step;

2. the second domain iterates $[(i,x) \rightarrow \textit{max}(f(i,x), f_l(i))]$ in $\mathcal{F}_{q,0}$

   $\Longrightarrow$ soundness does not depend on f_l

   $\Longrightarrow$ not widened at each step to wait until input are stables.

We use the reduction:

$$\rho : \begin{cases} \mathcal{F}_{0,r} \times \mathcal{F}_{q,0} & \mapsto \mathcal{F}_{0,r} \times \mathcal{F}_{q,0} \\ (x_0, m_0), (x_1, m_1) & \rightarrow (\textit{min}(x_0, x_1), m_0), (x_1, m_1) \end{cases}$$

after each computation step.

$\Longrightarrow$ The second domain is used to reduce the first one, when it is not accurate.

Unstable filters

In case the iterated function is not contracting, filters are very likely to diverge.
In case of linear filters, the iterated function is linear.
We may use the arithmetic-geometric progression domain [VMCAI'2005].
We require an external clock to relate the divergence to the value of the clock.

Overview

1. Introduction
2. Case studies
3. Concrete semantics
4. Generic aproximation
5. Filter domains
6. Post fixpoint inference of contracting function in floating-point arithmetics
7. Basic simplified filters
8. Higher order simplified filters
9. Bounded expansion
10. Filter detection
11. Conclusion

Simplified second order filter

Theorem 5 (Including rounding errors)

Let a, b, $\varepsilon_a \geq 0$, $\varepsilon_b \geq 0$, $K \geq 0$, $m \geq 0$, X, Y, Z be real numbers, such that:

1. $a^2 + 4b < 0$,
2. $X^2 - aXY - bY^2 \leq K$,
3. $aX + bY - (m + \varepsilon_a|X| + \varepsilon_b|Y|) \leq Z \leq aX + bY + (m + \varepsilon_a|X| + \varepsilon_b|Y|)$.

We have

1. $Z^2 - aZX - bX^2 \leq \left((\sqrt{-b} + \delta)\sqrt{K} + m\right)^2$;
2. $\begin{cases} \sqrt{-b} + \delta < 1 \\ K \geq \left(\frac{m}{1-\sqrt{-b}-\delta}\right)^2 \end{cases} \implies Z^2 - aZX - bX^2 \leq K$,

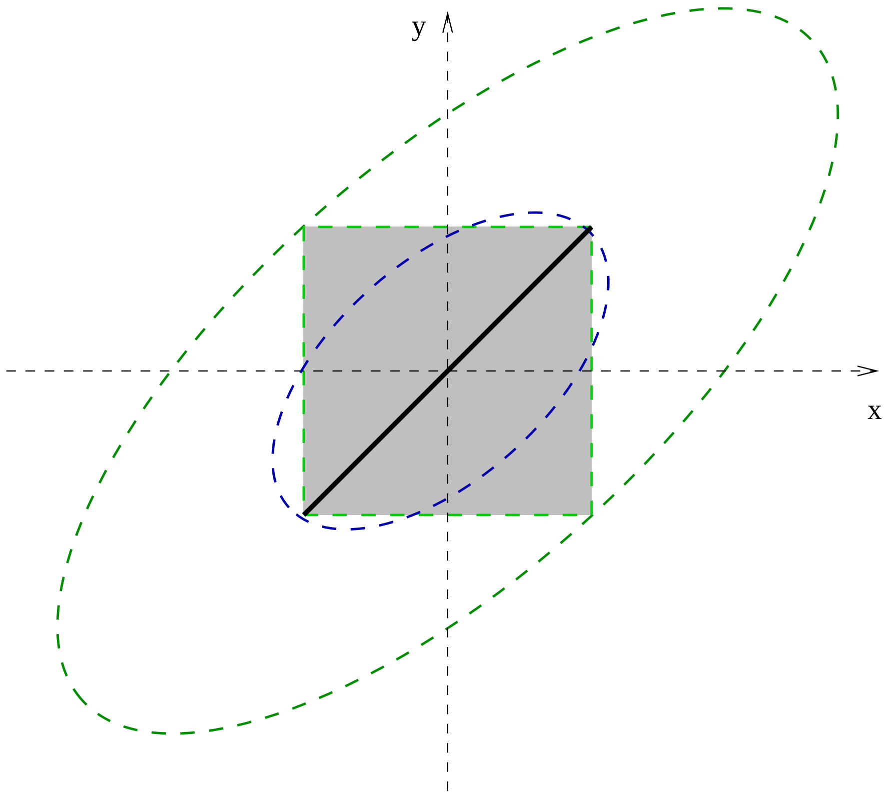
where $\delta = 2\frac{\varepsilon_b + \varepsilon_a\sqrt{-b}}{\sqrt{-(a^2+4b)}}$.

□

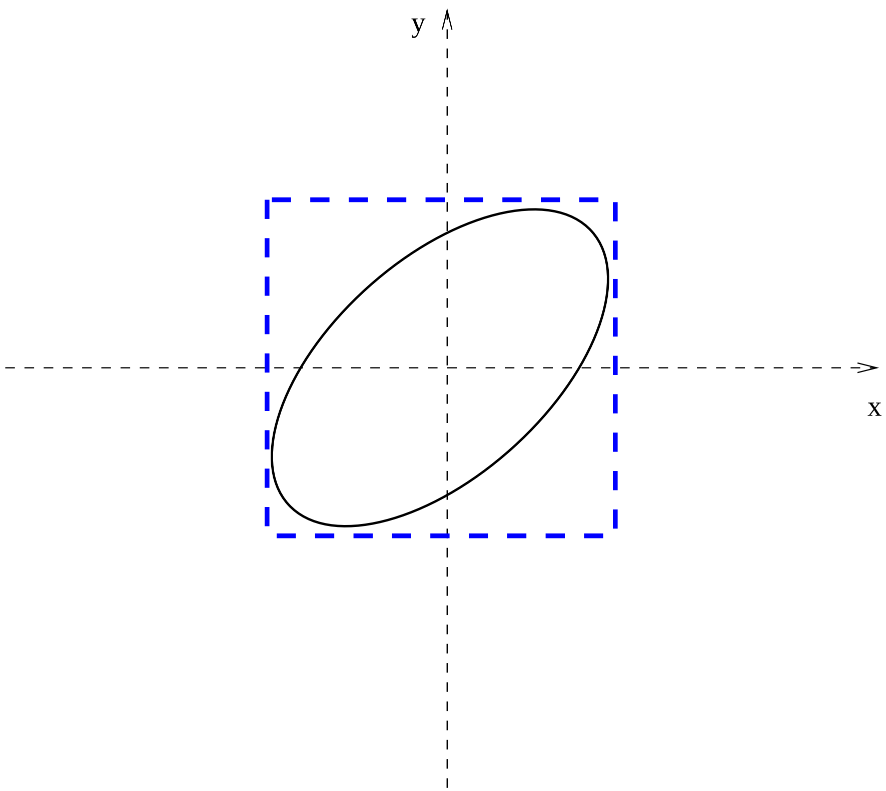
Domain

- The domain relates the variables describing the last two outputs and the four filter parameters to the square root of the ellipsis 'radius':
  $\gamma_{\mathcal{B}_1}((X, Y, a, \varepsilon_a, b, \varepsilon_b), k)$ is given by the set of environments ρ that satisfy:
  $$(\rho(X))^2 - a\rho(X)\rho(Y) - b(\rho(Y))^2 \leq k^2;$$
- in order to interpret assignment $Z = E$ under range constraints $\rho^\sharp$, we test whether E matches:
  $$[a - \varepsilon_a; a + \varepsilon_a] \times X + [b - \varepsilon_b; b + \varepsilon_b] \times Y + E'$$
  with $a^2 + 4b < 0$,
  and capture:
  - filter parameters: $(a, \varepsilon_a, b, \varepsilon_b)$;
  - variables tied before (X, Y) and after the iteration (Z, X),
  - an approximation of the current input: EVAL$^\sharp(E', \rho^\sharp)$.

Approximated reduced product



Initial conditions



Output refinement

Overview

1. Introduction
2. Case studies
3. Concrete semantics
4. Generic aproximation
5. Filter domains
6. Post fixpoint inference of contracting function in floating-point arithmetics
7. Basic simplified filters
8. Higher order simplified filters
9. Bounded expansion
10. Filter detection
11. Conclusion

Higher order simplified filters

A simplified filter of class (k, l) is defined as a sequence:

$$S_{n+p} = a_1 S_n + \ldots + a_p S_{n+p-1} + E_{n+p},$$

where the polynomial $P = X^p - a_p X^{p-1} - \ldots - a_1 X^0$ has no multiple roots (in $\mathbb{C}$) and can be factored into the product of k second order irreducible polynomials $X^2 - \alpha_i.X - \beta_i$ and l first order polynomials $X - \delta_j$.
Then, there exists sequences $(x_n^i)_{n\in\mathbb{N}}$ and $(y_n^j)_{n\in\mathbb{N}}$ such that:

$$\begin{cases} S_n = \left(\sum_{i=1}^{k} x_n^i\right) + \left(\sum_{j=1}^{l} y_n^j\right) \\ x_{n+2}^i = \alpha_i.x_{n+1}^i + \beta_i.x_n^i + F^i(E_{n+2}, E_{n+1}) \\ y_{n+1}^j = \delta_j.y_n^j + G^j(E_{n+1}). \end{cases}$$

The initial outputs ($x_0^i,\ x_1^i,\ y_0^j$) and filter inputs F^i, G^j are given by solving symbolic linear systems, they only depend on the roots of P.

Higher order simplified filters

Whenever we meet an assignment $V_{n+p} = E_{n+p} + \sum_{k=1}^{p} I_k \times V_{n+k-1}$,

1. we consider the characteristic polynomial $P = X^p - \sum_{k=1}^{p} I_k.X^{p-k}$,
2. we take a polynomial Q of the form $\prod_{i=1}^{k}(X^2 - A_i X - B_i)\prod_{j=1}^{l}(X - D_j)$ with $2k + l = p + 1$.
3. we expand Q into $X^p - \sum_{k=1}^{p} J_k.X^{p-k}$.
4. we bound the expression $|\sum_{k=1}^{p}(I_k - J_k) \times V_{n+k-1}| \leq \textit{err}(V_n, \ldots, V_{n+p-1})$;
5. we take the following assignment:

$$V_{n+p} = E_{n+p} + [-\textit{err}(V_n, \ldots, V_{n+p-1}), +\textit{err}(V_n, \ldots, V_{n+p-1})] + \sum_{k=1}^{p} J_k \times V_{n+k-1}$$

instead.

A sound factoring algorithm is not required !

Overview

1. Introduction
2. Case studies
3. Concrete semantics
4. Generic aproximation
5. Filter domains
6. Post fixpoint inference of contracting function in floating-point arithmetics
7. Basic simplified filters
8. Higher order simplified filters
9. Bounded expansion
10. Filter detection
11. Conclusion

Other filters

We consider sequences of the following form:

$$\begin{cases} S_k = i_k, 0 \leq k < p \\ S_{n+p} = \overline{F}(S_n, \ldots, S_{n+p-1}) \mp \overline{G}(E_{n+p+1-q}, \ldots, E_{n+p}) \end{cases}$$

Having bounds:

- on the input sequence (E_n),
- and on the initial outputs $(i_k)_{0 \leq k < p}$;

we want to infer a bound on the output sequence (S_n).

Splitting S_n

We split the output sequence $S_n = R_n + \varepsilon_n$ into

- the contribution of the errors (ε_n);

$$\begin{cases} \varepsilon_k = 0, \ 0 \leq k < p; \\ \varepsilon_{n+p} = F(\varepsilon_n, \ldots, \varepsilon_{n+p-1}) + \textit{err}_{n+p} \end{cases}$$

- the ideal sequence (R_n) (in the real field);

$$\begin{cases} R_k = i_k, \ 0 \leq k < p \\ R_{n+p} = F(R_n, \ldots, R_{n+p-1}) + G(E_{n+p+1-q}, \ldots, E_{n+p}) \end{cases}$$

Bounding R_n

To refine the output, we need to bound the sequence R_n:

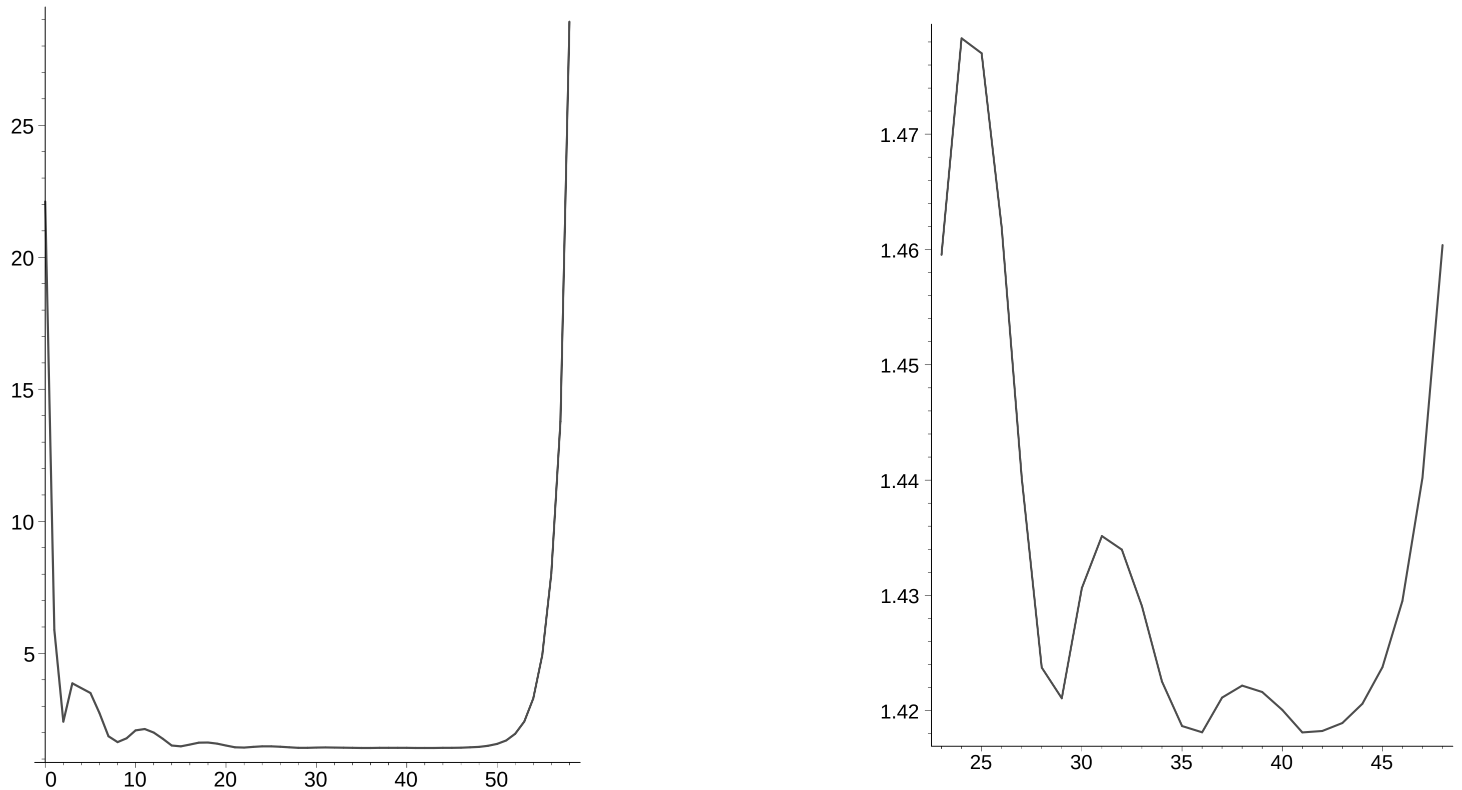
1. We isolate the contribution of the N last inputs:

$$R_n = \textit{last}_n^N(E_n, \ldots, E_{n+1-N}) + \textit{res}_n^N.$$

2. Since the filter is linear, we have, for $n > N + p$:
   - $\textit{last}_n^N(X_1, \ldots, X_N) = \textit{last}_{N+p}^N(X_1, \ldots, X_N)$;
   - $\textit{res}_n^N$ satisfies:

$$\textit{res}_{n+p}^N = F(\textit{res}_n^N, \ldots, \textit{res}_{n+p-1}^N) + G'_{[F,G]}(E_{n+p-N+1-q}, \ldots, E_{n+p-N})$$

Abstract gain with respect to N



Overview

Patterns

We use patterns to detect filter iterations:

$$P \stackrel{\Delta}{=} (P \oplus P) \mid (P \ominus P) \mid (P \otimes P) \mid (\ominus P) \mid c \in \mathit{Var}_{\mathit{cste}} \mid V \in \mathit{Var}_{\mathit{var}}$$

Patterns are seen up to the following congruence relation:

$$\begin{array}{rcll}
(P_1 \odot P_2) & \equiv_P & (P_2 \odot P_1) & \text{for } \odot \in \{\oplus, \otimes\} \\
((P_1 \odot P_2) \odot P_3) & \equiv_P & (P_1 \odot (P_2 \odot P_3)) & \text{for } \odot \in \{\oplus, \otimes\} \\
P_1 & \equiv_P & (\ominus(\ominus P_1)) & \\
(P_1 \ominus P_2) & \equiv_P & (P_1 \oplus (\ominus P_2)) & \\
\ominus(P_1 \odot P_2) & \equiv_P & ((\ominus P_1) \odot (\ominus P_2)) & \text{for } \odot \in \{\oplus, \ominus\} \\
\ominus(P_1 \otimes P_2) & \equiv_P & ((\ominus P_1) \otimes P_2) & \\
\ominus(P_1 \otimes P_2) & \equiv_P & (P_1 \otimes (\ominus P_2)) &
\end{array}$$

Expressions

We consider:

1. interval constraints:

$$\rho_I \ : \ \mathcal{V} \rightarrow \textit{Interval}$$

2. symbolic constraints [Miné: VMCAI'06]:

$$\rho_C \ : \ \mathcal{V} \rightarrow \textit{Expression} \cup \{\top\}$$

Expressions in assignments are seen up the following congruence:

$$E \equiv_E \tilde{\rho}_I(E)$$
$$V \equiv_E \rho_C(V) \text{ if } \rho_C(V) \neq \top$$

Pattern matching

Given $\rho_I : \mathcal{V} \rightarrow \textit{Interval}$ and $\rho_C : \mathcal{V} \rightarrow \textit{Expression} \cup \{\top\}$,
we define the relation $\models_{\rho_{cste},\rho_{var}}$ by induction as follows:

If: $E_1 \models_{\rho_{cste},\rho_{var}} P_1$ and $E_2 \models_{\rho_{cste},\rho_{var}} P_2$
then:

$$\begin{array}{rll}
(E_1 \overline{+} E_2) \models_{\rho_{cste},\rho_{var}} & (P_1 \oplus P_2) & \\
(E_1 \overline{-} E_2) \models_{\rho_{cste},\rho_{var}} & (P_1 \ominus P_2) & \\
(E_1 \overline{\times} E_1) \models_{\rho_{cste},\rho_{var}} & (P_1 \otimes P_2) & \\
E \models_{\rho_{cste},\rho_{var}} & c & \text{if } \rho_{cste}(c) = \tilde{\rho}_I(E) \\
E \models_{\rho_{cste},\rho_{var}} & (\ominus c) & \text{if } \rho_{cste}(c) = \tilde{\rho}_I(\overline{-} E) \\
X \models_{\rho_{cste},\rho_{var}} & V & \text{if } \rho_{var}(V) = X
\end{array}$$

When $E \models_{\rho_{cste},\rho_{var}} P$, we say that the expression E matches the pattern P under the environments ρ_I and ρ_c.

Abstract pattern matching

Given an expression E and a pattern P,
find a set of tuples $(E', P', \rho_{\mathit{cste}}, \rho_{\mathit{var}})$ such that:

1. $E \equiv_E E'$;
2. $P \equiv_P P'$;
3. $E \models_{\rho_{\mathit{cste}}, \rho_{\mathit{var}}} P$.

We explore E and P in parallel, when necessary:

1. we reorder terms and factors in P;
2. we introduce unary negations in P;
3. we push negations toward the leaves of P;
4. we replace variables in E with their symbolic constraint;

Memoization / Certificate

Exploration is costly (exponential in the size of P).
We use memoization to amortize this cost.

1. After each exploration, we memoize:
   - successful tuples (just E' and P' indead);
     (they can be used as certificate for *a posteriori* checks)
   - symbolic constraints that have been used;
2. At next iterations:
   - when these symbolic constraints have changed,
     we redo the exploration;
   - otherwise we check which tuples are still valid.

We deal with rounding errors the usual way.

Overview

1. Introduction
2. Case studies
3. Concrete semantics
4. Generic aproximation
5. Filter domains
6. Post fixpoint inference of contracting function in floating-point arithmetics
7. Basic simplified filters
8. Higher order simplified filters
9. Bounded expansion
10. Filter detection
11. Conclusion

Benchmarks

We analyze three programs in the same family on a AMD Opteron 248, 8 Gb of RAM (analyses use only 2 Gb of RAM).

lines of C	70,000			216,000			379,000		
global variables	13,400			7,500			9,000		
iterations	72	41	**37**	161	75	**53**	151	187	**74**
time/iteration	52s	1mn18s	**1mn16s**	3mn07s	5mn08s	**4mn40s**	4mn35s	9mn25s	**8mn17s**
analysis time	1h02mn	53mn	**47mn**	8h23mn	6h25mn	**4h08mn**	11h34mn	30h26mn	**10h14mn**
false alarms	574	3	**0**	207	0	**0**	790	0	**0**

1. without filter domains;
2. with simplified filter domains;
3. with expanded filter domains.

Conclusion

- a highly generic framework to analyze programs with digital filtering:
  a technical knowledge of used filters allows the design of the adequate abstract domain;
- the case of linear filters is fully handled:
  we need to solve a symbolic linear system for each filter family;
  we need an (not necessarily sound) polynomial reduction algorithm for each filter instance.
- filters are detected up to:
  - term recombination
  - and some laws of the real fields;

This framework has been used and was necessary in the full certification of the absence of run-time error in industrial critical embedded software.

http://www.astree.ens.fr