Static Analysis of Digital Filters*

Jérôme Feret

DI, École Normale Supérieure, Paris, FRANCE
jerome.feret@ens.fr

Abstract. We present an Abstract Interpretation-based framework for automatically analyzing programs containing digital filters. Our framework allows refining existing analyses so that they can handle given classes of digital filters. We only have to design a class of symbolic properties that describe the invariants throughout filter iterations, and to describe how these properties are transformed by filter iterations. Then, the analysis allows both inference and proofs of the properties about the program variables that are tied to any such filter.

1 Introduction

Digital filters are widely used in real-time embedded systems (as found in automotive, aeronautic, and aerospace applications) since they allow modeling into software behaviors previously ensured by analogical filters. A filter transforms an input stream of floating-point values into an output stream. Existing analyses are very imprecise in bounding the range of the output stream, because of the lack of precise linear properties that would entail that the output is bounded. The lack of precise domains when analyzing digital filters was indeed the cause of almost all the remaining warnings (potential floating-point overflows) in the certification of a critical software family with the analyzer described in [1,2].

In this paper, we propose an Abstract Interpretation-based framework for designing new abstract domains which handle filter classes. Human intervention is required for discovering the general shape of the properties that are required in proving the stability of such a filter. Roughly speaking, filter properties are mainly an abstraction of the input stream, from which we deduce bounds on the output stream. Our framework can then be used to build the corresponding abstract domain. This domain propagates all these properties throughout the abstract computations of programs. Our approach is not syntactic, so that loop unrolling, filter reset, boolean control, and trace (or state) partitioning are dealt with for free and any filter of the class (for any setting) is analyzed precisely.

Moreover, in case of linear filters, we propose a general approach to build the corresponding class of properties. We first design a rough abstraction, in which at each filter iteration, we do not distinguish between the contributions of each input. Then, we design a precise abstraction: using linearity, we split the output between the global contribution of floating-point errors, and the contribution of

* This work was partially supported by the ASTRÉE RNTL project.

D.A. Schmidt (Ed.): ESOP 2004, LNCS 2986, pp. 33–48, 2004.

the ideal filter behavior. Global floating-point error contribution is then bounded using the rough abstraction, while ideal filter output is precisely described by formally expanding it, so that the contribution of each input is exactly described in the real field, and then approximated in floating-point arithmetics.

We have instantiated our framework for two kinds of widely used linear filters: the *high bandpass* and *second order* filters. The framework was fully implemented in OCaml [8] and plugged into an existing analyzer. We have obtained bounds that are very close to sample experimental results, which has allowed solving nearly all of our remaining warnings [2].

Previous works. To our knowledge, this is the first analysis that abstracts filter output invariants. Nevertheless, some work has been done in filter optimization. In [7], affine equality relationships [6] among variables at the beginning and at the end of loop iterations are used to factorize filters at compile time. In our case, because of floating-point rounding errors, there are no such affine equality relationships, so a more complex domain such as polyhedra [5] is required to perform the same task. Moreover, our programs involve complex boolean control flows. Thus, filter factorization cannot be performed without a highly expensive partitioning. Furthermore, our goal is just to prove the absence of error at runtime, and not to describe precisely the global behavior of filters.

Outline. In Sect. 2, we present the syntax and semantics of our language. In Sect. 3, we describe a generic abstraction for this language. In Sect. 4, we define a generic extension for refining existing abstractions. In Sect. 5, we give numerical abstract domains for describing sets of real numbers. In Sect. 6, we describe, on our examples, the general approach to building such generic extensions. In Sect. 7, we describe the impact of these extensions on the analysis results.

2 Language

We analyze a subset of C without dynamic memory allocation nor side-effect. Moreover, the use of pointer operations is restricted to call-by reference. For the sake of simplicity, we introduce an intermediate language to describe programs interpreted, between the concrete and an abstract level. We suppose that *lvalue* resolution has been partially solved (see [2, Sect. 6.1.3]). Furthermore, assignments (which use floating-point arithmetics at the concrete level) have been conservatively abstracted into non-deterministic assignments in the real field, i.e., floating-point expressions have been approximated by linear forms with real interval coefficients. These linear forms include both the rounding errors and some expression approximations (see [10]). We also suppose that runtime errors (such as floating-point overflows) can be described by interval constraints on the memory state after program computations.

Let $\mathcal{V}$ be a finite set of variables. We denote by $\mathcal{I}$ the set of all real number intervals (including $\mathbb{R}$ itself). We define inductively the syntax of programs in Fig. 1. We denote by $\mathcal{E}$ the set of expressions E. We describe the semantics of these programs in a denotational way. An *environment* ($\rho \in \mathcal{V} \to \mathbb{R}$) denotes a memory state. It maps each variable to a real number. We denote by *Env* the set

$V \in \mathcal{V}$, $I \in \mathcal{I}$
$E := I \mid V \mid (-V) + E \mid V + E \mid I \times V + E$
$P := V = E \mid \text{skip} \mid \text{if } (V \geq 0)\ \{P\} \text{ else } \{P\} \mid \text{while } (V \geq 0)\ \{P\} \mid P; P$

Fig. 1. Syntax.

$[\![I]\!]_E(\rho) = I$, $[\![V]\!]_E(\rho) = \{\rho(V)\}$
$[\![(-V) + E]\!]_E(\rho) = \{a - \rho(V) \mid a \in [\![E]\!]_E(\rho)\}$, $[\![V + E]\!]_E(\rho) = \{\rho(V) + a \mid a \in [\![E]\!]_E(\rho)\}$
$[\![I \times V + E]\!]_E(\rho) = \{b \times \rho(V) + a \mid a \in [\![E]\!]_E(\rho), b \in I\}$
$[\![V = E]\!]_P(\rho) = \{\rho[V \mapsto x] \mid x \in [\![E]\!]_E(\rho)\}$
$[\![\text{skip}]\!]_P(\rho) = \{\rho\}$
$$[\![\text{if } (V \geq 0)\ \{P_1\} \text{ else } \{P_2\}]\!]_P(\rho) = \begin{cases} [\![P_1]\!]_P(\rho) & \text{if } \rho(V) \geq 0 \\ [\![P_2]\!]_P(\rho) & \text{otherwise} \end{cases}$$
$[\![\text{while } (V \geq 0)\ \{P\}]\!]_P(\rho) = \{\rho' \in Inv \mid \rho'(V) < 0\}$
where $Inv = \text{lfp}\left(X \mapsto \{\rho\} \cup \left(\bigcup\{[\![P]\!]_P(\rho') \mid \rho' \in X, \rho'(V) \geq 0\}\right)\right)$
$[\![P_1; P_2]\!]_P(\rho) = \bigcup\{[\![P_2]\!]_P(\rho') \mid \rho' \in [\![P_1]\!]_P(\rho)\}$

Fig. 2. Concrete semantics.

of all environments. The semantics of a program P is a function ($[\![P]\!]_P \in Env \to \wp(Env)$) mapping each environment ρ to the set of the environments that can be reached when applying the program P starting from the environment ρ. The function $[\![\_]\!]_P$ is defined by induction on the syntax of programs in Fig. 2. Loop semantics requires the computation of a *loop invariant*, which is the set of all environments that can be reached just before the guard of this loop is tested. This invariant is well-defined as the least fixpoint of a $\cup$-complete endomorphism in the powerset $\wp(Env)$. Nevertheless, such a fixpoint is usually not computable, so we give a decidable approximate semantics in the next section.

We describe two filter examples, that we will use throughout the paper.

Example 1. A *high bandpass* filter can be encoded by the following program:

$$\begin{array}{l} V = \mathbb{R};\ E_1 = [0; 0]; \\ \text{while } (V \geq 0)\ \{ \\ \quad V = \mathbb{R};\ T = \mathbb{R};\ E_0 = I; \\ \quad \text{if } (T \geq 0)\ \{S = 0\} \\ \quad \text{else } \{S = A \times S + E_0 + (-E_1) + F\}; \\ \quad E_1 = E_0; \\ \} \end{array}$$

Roughly speaking, the interval I denotes the range of filter entries. Floating-point rounding errors are captured by the range of both intervals A and F. The interval A describes the filter coefficient and satisfies $A \subseteq [\frac{1}{2}; 1[$. Variables V and T allow control flow enforcement. At each loop iteration, the variable S denotes the value of the current filter output, the variable E_0 denotes the value of the current filter input, and the variable E_1 denotes the value of the previous filter input. Depending on the value of T, the filter is either reset (i.e., the output is set to 0), or iterated (i.e., the value of the next output is calculated from the last output value and the two last input values). The analysis described in [2]

only discovers inaccurate bounds for the variable S. It works as if the expression $A \times S + E_0 - E_1 + F$ were approximated by $A \times S + (2 \times I + F)$. The analysis discovers the first widening threshold l (see [1, Sect. 2.1.2]) such that l is greater than $\frac{2 \times i + f}{1 - a}$, for any $(i, f, a) \in I \times F \times A$. It proves that l is stable, and then successive narrowing iterations refine the value l. □

Example 2. A *second order* digital filter can be encoded as follows:

$V = \mathbb{R}$; $E_1 = [0; 0]$; $E_2 = [0; 0]$;
while $(V \geq 0)$ {
  $V = \mathbb{R}$; $T = \mathbb{R}$; $E_0 = I$;
  if $(T \geq 0)$ $\{S_0 = E_0;\ S_1 = E_0\}$
  else $\{S_0 = A \times S_1 + B \times S_2 + C \times E_0 + D \times E_1 + E \times E_2 + F\}$;
  $E_2 = E_1$; $E_1 = E_0$; $S_2 = S_1$; $S_1 = S_0$
}

Roughly speaking, the interval I denotes the range of filter entries. Intervals A, B, C, D and E denote filter coefficients and satisfy $A \subseteq [0; \infty[$, $B \subseteq]-1; 0[$ and $\forall (a, b) \in A \times B,\ a^2 + 4 \times b < 0$. Floating-point rounding errors are captured by the range of intervals A, B, C, D, E and F. Variables V and T allow control flow enforcement. At each loop iteration, the variable S_0 denotes the value of current filter output, variables S_1 and S_2 denote the last two values of filter output, the variable E_0 denotes the value of the current filter input, and variables E_1 and E_2 denote the last two values of filter input. Depending on the value of T, either the filter is reset (i.e., both the current and the previous outputs are set to the same value), or iterated (i.e., the value of the next output is calculated from the last two output values and the last three input values). The analysis described in [2] fails to discover any bound for the variables S_0, S_1, S_2. □

3 Underlying Domain

We use the Abstract Interpretation framework [3,4] to derive a generic approximate semantics. An abstract domain $Env^\sharp$ is a set of properties about memory states. Each abstract property is related to the set of the environments which satisfy it via a concretization map γ. An operator $\sqcup$ allows the gathering of information about different control flow paths. Some primitives ASSIGN and GUARD are sound counterparts to concrete assignments and guards. To effectively compute an approximation of concrete fixpoints, we introduce an iteration basis $\bot$, a widening operator $\triangledown$ and a narrowing operator $\triangle$. Several abstract domains collaborate, and refine each others using very simple constraints: variable ranges and equality relations. These constraints are related to the concrete domains via two concretization functions $\gamma_{\mathcal{I}}$ and $\gamma_{=}$ that respectively map each function $\rho^\sharp \in \mathcal{V} \to \mathcal{I}$ to the set of the environments ρ such that $\forall X \in \mathcal{V},\ \rho(X) \in \rho^\sharp(X)$, and each relation $\mathcal{R} \subseteq \mathcal{V}^2$ to the set of environments ρ such that for any variables X and Y, $(X, Y) \in \mathcal{R}$ implies that $\rho(X) = \rho(Y)$. The primitives RANGE and EQU capture simple constraints about the filter input stream, and about

$[\![V = E]\!]^\sharp(a) = \text{ASSIGN}(V = E, a)$
$[\![\text{skip}]\!]^\sharp(a) = a$
$[\![\text{if } (V \geq 0)\ \{P_1\} \text{ else } \{P_2\}]\!]^\sharp(a) = a_1 \sqcup a_2$, with $\begin{cases} a_1 = [\![P_1]\!]^\sharp(\text{GUARD}(V, [0; +\infty[, a)) \\ a_2 = [\![P_2]\!]^\sharp(\text{GUARD}(V,]-\infty; 0[, a)) \end{cases}$
$[\![\text{while } (V \geq 0)\ \{P\}]\!]^\sharp(a) = \text{GUARD}(V,]-\infty; 0[, Inv^\sharp)$
where $Inv^\sharp = \text{lfp}^\sharp\left(X \mapsto a \sqcup [\![P]\!]^\sharp(\text{GUARD}(V, [0; +\infty[, X))\right)$
$[\![P_1; P_2]\!]^\sharp(\rho^\sharp) = [\![P_2]\!]^\sharp([\![P_1]\!]^\sharp(\rho^\sharp))$

Fig. 3. Abstract semantics.

the initialization state, to be passed to the filter domains. Conversely, a primitive REDUCE receives the constraints about the filter output range to refine the underlying domain.

Definition 1 (Generic abstraction). *An abstraction is defined by a tuple* $(Env^\sharp, \gamma, \sqcup, \text{ASSIGN}, \text{GUARD}, \bot, \nabla, \triangle, \text{RANGE}, \text{EQU}, \text{REDUCE})$ *such that:*

1. $Env^\sharp$ *is a set of properties;*
2. $\gamma \in Env^\sharp \to \wp(Env)$ *is a concretization map;*
3. $\forall a, b \in Env^\sharp, \gamma(a) \cup \gamma(b) \subseteq \gamma(a \sqcup b)$*;*
4. $\forall a \in Env^\sharp, \rho^\sharp \in (\mathcal{V} \to \mathcal{I})$, $\gamma(a) \cap \gamma_{\mathcal{I}}(\rho^\sharp) \subseteq \gamma(\text{REDUCE}(\rho^\sharp, a))$, *moreover* $\text{REDUCE}([X \mapsto \mathbb{R}], a) = a$*;*
5. ∇ *is a widening operator such that:* $\forall a, b \in Env^\sharp,\ \gamma(a) \cup \gamma(b) \subseteq \gamma(a \nabla b)$*; and* $\forall \rho^\sharp \in (\mathcal{V} \to \mathcal{I})$, $(a_i) \in (Env^\sharp)^{\mathbb{N}}$, *the sequence* $\left(a_i^\nabla\right)$ *defined by* $a_0^\nabla = \text{REDUCE}(\rho^\sharp, a_0)$ *and* $a_{n+1}^\nabla = \text{REDUCE}(\rho^\sharp, a_n^\nabla \nabla a_{n+1})$ *is ultimately stationary;*
6. $\triangle$ *is a narrowing operator such that:* $\forall a, b \in Env^\sharp,\ \gamma(a) \cap \gamma(b) \subseteq \gamma(a \triangle b)$*; and* $\forall \rho^\sharp \in (\mathcal{V} \to \mathcal{I})$, $(a_i) \in (Env^\sharp)^{\mathbb{N}}$, *the sequence* $\left(a_i^\triangle\right)$ *defined by* $a_0^\triangle = \text{REDUCE}(\rho^\sharp, a_0)$ *and* $a_{n+1}^\triangle = \text{REDUCE}(\rho^\sharp, a_n^\triangle \triangle a_{n+1})$*, is ultimately stationary;*
7. $\forall a \in Env^\sharp, X \in \mathcal{V}, E \in \mathcal{E}, \rho \in \gamma(a),\ [\![X = E]\!]_P(\rho) \subseteq \gamma(\text{ASSIGN}(X = E, a))$*;*
8. $\forall a \in Env^\sharp, X \in \mathcal{V}, I \in \mathcal{I},\ \{\rho \in \gamma(a) \mid \rho(X) \in I\} \subseteq \gamma(\text{GUARD}(X, I, a))$*;*
9. $\forall a \in Env^\sharp,\ \gamma(a) \subseteq \gamma_{\mathcal{I}}(\text{RANGE}(a))$ *and* $\gamma(a) \subseteq \gamma_=(\text{EQU}(a))$.

Least fixpoint approximation is performed in two steps [3]: we first compute an approximation using the widening operator; then we refine it using the narrowing operator. More formally, let f be a $\cup$-complete endomorphism of $\wp(Env)$, and $(f^\sharp \in Env^\sharp \to Env^\sharp)$ be an abstract counterpart of f satisfying $\forall a \in Env^\sharp,\ (f \circ \gamma)(a) \subseteq (\gamma \circ f^\sharp)(a)$. The abstract upward iteration (C_n^∇) of $f^\sharp$ is defined by $C_0^\nabla = \bot$ and $C_{n+1}^\nabla = C_n^\nabla \nabla f^\sharp(C_n^\nabla)$. The sequence (C_n^∇) is ultimately stationary and its limit C_ω^∇ satisfies $\text{lfp}(f) \subseteq \gamma(C_\omega^\nabla)$. Then the abstract downward iteration $(D_n^\triangle)$ of $f^\sharp$ is defined by $D_0^\triangle = C_\omega^\nabla$ and $D_{n+1}^\triangle = D_n^\triangle \triangle f^\sharp(D_n^\triangle)$. The sequence $(D_n^\triangle)$ is ultimately stationary and its limit $D_\omega^\triangle$ satisfies $\text{lfp}(f) \subseteq \gamma(D_\omega^\triangle)$. So we define $\text{lfp}^\sharp(f^\sharp)$ by the limit of the abstract downward iteration of $f^\sharp$.

The abstract semantics of a program is given by a function $([\![\_]\!]^\sharp \in Env^\sharp \to Env^\sharp)$ in Fig. 3. Its soundness can be proved by induction on the syntax:

Theorem 1. *For any program* P*, environment* ρ*, abstract element* a*, we have:*

$$\rho \in \gamma(a) \implies [\![P]\!]_P(\rho) \subseteq \gamma\left([\![P]\!]^\sharp(a)\right). \qquad \square$$

4 Generic Extension

We now show how an analysis can be extended to take into account a given class of digital filters. We first introduce all the primitives that we need to build such a domain. Then we define a single domain. At last, we build an approximate reduced product between such a domain and an underlying domain.

4.1 Primitives

A filter domain collects constraints that relate some variables that will be used again at the next filter iteration with some filter parameters and a dynamic information. This provides approximation of both filter input and output. For instance, the variables may be the last outputs, the parameters some directing coefficients, and the dynamic information a radius inferred during the analysis. Let m be the numbers of variables to be used and n that of the filter parameters. The abstract domain maps tuples in $\mathcal{T} = \mathcal{V}^m \times \mathbb{R}^n$ to elements of the set $\mathcal{B}$ of dynamic information. The set of the environments that satisfies a constraint $c \in \mathcal{T} \times \mathcal{B}$ is given by its concretization $\gamma_{\mathcal{B}}(c)$. An operator $\sqcup_{\mathcal{B}}$ is used to merge constraints related to the same tuple, but coming from distinct flows. For each assignment $X = E$ interpreted in an environment satisfying the range constraints described by $\rho^\sharp \in \mathcal{V} \to \mathcal{I}$, the set $\textsc{rlvt}(X = E)$ denotes the set of tuples corresponding to the constraints that are modified by this assignment (usually the tuples containing some variables that occurs in the expression E). For each of these tuples t associated with the dynamic information a, the pair $\textsc{pt}(X = E, t, \rho^\sharp) = (t', \mathit{info})$ is such that t' is the tuple that is tied, after the assignment (obtained by shifting the variables), by the new constraint[1], and *info* contains all the parameters about both the filter and the input stream that are required to infer the dynamical information. This information is itself updated by $\delta(\mathit{info}, a)$. The element $\bot_{\mathcal{B}}$ provides the basis of iterations. Extrapolation operators $\triangledown_{\mathcal{B}}$ and $\triangle_{\mathcal{B}}$ allow the concrete post-fixpoint approximation. A primitive BUILD receives some range and equality constraints from the underlying domain to build filter constraints when the filter is reset. Conversely, the function TO extracts information about the range of the output stream to be passed to the underlying domain.

Definition 2 (Generic extension). *An abstract extension is given by a tuple* $(\mathcal{T}, \mathcal{B}, \gamma_{\mathcal{B}}, \sqcup_{\mathcal{B}}, \textsc{rlvt}, \textsc{info}, \textsc{pt}, \delta, \bot_{\mathcal{B}}, \triangledown_{\mathcal{B}}, \triangle_{\mathcal{B}}, \textsc{build}, \textsc{to})$ *which satisfies:*

1. $\mathcal{T} = \mathcal{V}^m \times \mathbb{R}^n$*, where* $m, n \in \mathbb{N}$*;* $\mathcal{B}$ *is a set of dynamical information;*
2. $\gamma_{\mathcal{B}} \in \mathcal{T} \times \mathcal{B} \to \wp(\mathit{Env})$ *is a concretization map;*
3. $\forall a, b \in \mathcal{B}, t \in \mathcal{T},\ \gamma_{\mathcal{B}}(t, a) \cup \gamma_{\mathcal{B}}(t, b) \subseteq \gamma_{\mathcal{B}}(t, a \sqcup_{\mathcal{B}} b)$*;*
4. ***assignments:*** $\textsc{rlvt}$ *maps an expression* $E \in \mathcal{E}$ *to a subset* T *of* $\mathcal{T}$*;* $\textsc{info}$ *is a set of filter parameters;* $\forall X \in \mathcal{V}, E \in \mathcal{E}, \rho^\sharp \in \mathcal{V} \to \mathcal{I}$*, we have* $\textsc{pt}(X = E, t, \rho^\sharp) \in \mathcal{T} \times \textsc{info}$*, the map* $[t \mapsto \mathit{fst}(\textsc{pt}(X = E, t, \rho^\sharp))]$ *is injective, and*

[1] $\rho^\sharp$ may be used to interpret some variables that occur in E. It also infers a bound to the filter input.

$\forall t \in \text{RLVT}(E), a \in \mathcal{B}$, *such that* $\rho \in \gamma_{\mathcal{B}}(t,a) \cap \gamma_{\mathcal{I}}(\rho^\sharp)$, *we have* $[\![X = E]\!]_P(\rho) \subseteq \gamma_{\mathcal{B}}(t', \delta(info, a))$, *with* $(t', info) = \text{PT}(X = E, t, \rho^\sharp)$;

5. $\nabla_{\mathcal{B}}$ *is a widening operator such that:* $\forall a, b \in \mathcal{B},\ \forall t \in \mathcal{T},\ \gamma_{\mathcal{B}}(t,a) \cup \gamma_{\mathcal{B}}(t,b) \subseteq \gamma_{\mathcal{B}}(t, (a \nabla_{\mathcal{B}} b))$; *and* $\forall (a_i) \in \mathcal{B}^{\mathbb{N}}$, *the sequence* $\left(a_i^\nabla\right)$ *defined by* $a_0^\nabla = a_0$ *and* $a_{n+1}^\nabla = a_n^\nabla \nabla_{\mathcal{B}} a_{n+1}$ *is ultimately stationary;*
6. $\Delta_{\mathcal{B}}$ *is a narrowing operator such that:* $\forall a, b \in \mathcal{B},\ \forall t \in \mathcal{T}, \gamma_{\mathcal{B}}(t,a) \cap \gamma_{\mathcal{B}}(t,b) \subseteq \gamma_{\mathcal{B}}(t, a \Delta_{\mathcal{B}} b)$; $\forall (a_i) \in \mathcal{B}^{\mathbb{N}}$, *the sequence* $\left(a_i^\Delta\right)$ *defined by* $a_0^\Delta = a_0$ *and* $a_{n+1}^\Delta = a_n^\Delta \Delta_{\mathcal{B}} a_{n+1}$, *is ultimately stationary;*
7. ***filter constraint synthesis*** *from the underlying domain:* $\forall \rho^\sharp \in \mathcal{V} \to \mathcal{I}, \mathcal{R} \in \wp(\mathcal{V}^2), t \in \mathcal{T},\ \gamma_{\mathcal{I}}(\rho^\sharp) \cap \gamma_{=}(\mathcal{R}) \subseteq \gamma_{\mathcal{B}}(t, \text{BUILD}(t, \rho^\sharp, \mathcal{R}))$;
8. ***interval constraint synthesis:*** $\text{TO} \in (\mathcal{T} \times \mathcal{B}) \to (\mathcal{V} \to \mathcal{I})$ *and* $\forall t \in \mathcal{T}, a \in \mathcal{B},\ \gamma_{\mathcal{B}}(t,a) \subseteq \gamma_{\mathcal{I}}(\text{TO}(t,a))$.

4.2 Abstract Domain

We now build an abstraction from a generic extension. We first enrich the set $\mathcal{B}$ with an extra element $\top_{\mathcal{B}} \notin \mathcal{B}$. That element will denote the fact that a tuple is tied to no constraint. We set $\gamma_{\mathcal{B}}(t, \top_{\mathcal{B}}) = Env^\sharp$ and $\text{TO}(t, \top_{\mathcal{B}}) = Env^\sharp$. We also lift other primitives of the domain so that they return $\top_{\mathcal{B}}$ as soon as one of their arguments is $\top_{\mathcal{B}}$. The abstract domain $Env_F^\sharp = (\mathcal{T} \to \mathcal{B})$ is related to $\wp(Env)$ by the concretization function γ_F which maps $f \in Env_F^\sharp$ to the set of environments $\left(\bigcap_{t \in \mathcal{T}} \gamma_{\mathcal{B}}(t, f(t))\right)$ that satisfy all the constraints encoded by f. The operator $\sqcup_F$ applies componentwise the operator $\sqcup_{\mathcal{B}}$. The abstract assignment may require information about variable ranges in order to extract filter parameters (in particular the input values). The abstract assignment $\text{ASSIGN}_F^{\rho^\sharp}(X = E, f)$ of an abstract element f under the interval constraints $\rho^\sharp \in \mathcal{V} \to \mathcal{I}$, is given by the abstract element f' where:

1. if E is a variable Y, each constraint containing Y gives a constraint for X: we take $f'(t) = f(\sigma t)$, where σ substitutes each occurrence of X by Y;
2. otherwise, we remove each constraint involving X, and add each constraint corresponding to some filter iteration, $f'(t)$ is given by:

$$\begin{cases} \delta(info, f(t_{-1})) & \text{if } \exists t_{-1} \in \text{RLVT}(E), (t, info) = \text{PT}(X = E, t_{-1}, \rho^\sharp), \\ f(t) & \text{if } t \in (\mathcal{V} \setminus \{X\})^m \times \mathbb{R}^n, \\ \top_{\mathcal{B}} & \text{otherwise;} \end{cases}$$

We also set $\text{ASSIGN}_F(X = E, f) = \text{ASSIGN}_F^{\text{RANGE}_F(f)}(X = E, f)$. Since filter invariants do not rely on guards, we set $\text{GUARD}_F(X, I, f) = f$. The function $\text{RANGE}_F(f)$ maps each variable X to the interval $\left(\bigcap_{t \in \mathcal{T}} \text{TO}_F(t, f(t))(X)\right)$; since filters do not generate syntactic equality constraints, we take $\text{EQU}_F = \emptyset$. We also ignore the constraints that are passed by the other domains, this way we take $\text{REDUCE}_F(\rho^\sharp, a) = a$. The $\bot_F$ element maps each tuple to the element $\top_{\mathcal{B}}$. The operators (∇_F, Δ_F) are defined componentwise. We define the tuple $\mathcal{A}_F$ by $(Env_F^\sharp, \gamma_F, \sqcup_F, \text{ASSIGN}_F, \text{GUARD}_F, \bot_F, \nabla_F, \Delta_F, \text{RANGE}_F, \text{EQU}_F, \text{REDUCE}_F)$.

Theorem 2. *The tuple* $\mathcal{A}_F$ *is an abstraction.* □

4.3 Product and Approximate Reduced Product

Unfortunately the abstraction $\mathcal{A}_F$ cannot compute any constraint, mainly because of inaccurate assignments and guards, and abstract iterations always return the top element. Hence we use a reduced product between this abstraction and an existing underlying abstraction to refine and improve the analysis.
Let $\mathcal{A}_0 = (\mathit{Env}_0^\sharp, \gamma_0, \sqcup_0, \textsc{assign}_0, \textsc{guard}_0, \bot_0, \triangledown_0, \vartriangle_0, \textsc{range}_0, \textsc{equ}_0, \textsc{reduce}_0)$ be an abstraction. We first introduce the product of the two abstractions. The abstract domain $\mathit{Env}^\sharp$ is the Cartesian product $\mathit{Env}_0^\sharp \times \mathit{Env}_F^\sharp$. The operator $\sqcup$, the transfer functions $(\textsc{assign}, \textsc{guard})$, the function $\textsc{reduce}$ and the extrapolation operators $(\bot, \triangledown, \vartriangle)$ are all defined pairwise. The concretization and the remaining refinement primitives are defined as follows:

$$\begin{cases} \gamma(a,f) = \gamma_0(a) \cap \gamma_F(f), \\ \textsc{equ}(a,f)(X,Y) \iff \textsc{equ}_0(a)(X,Y) \text{ or } \textsc{equ}_F(f)(X,Y), \\ \textsc{range}(X)(a,f) = \textsc{range}_0(X)(a) \cap \textsc{range}_F(X)(f), \end{cases}$$

which corresponds to taking the meet of collected information.

We refine abstract assignments and binary operators:

- before assignments, we synthesize the filter constraints that are relevant for the assignment; during assignments, we use the underlying constraints to interpret the expression; and after assignments, we use the filter constraints to refine the properties in the underlying domain. That is to say, we define $\textsc{assign}'(X = E, (a,f))$ by $(\textsc{reduce}_0(\textsc{range}_F(f''), a''), f'')$ where:
  - $(a'', f'') = (\textsc{assign}_0(X = E, a), \textsc{assign}_F^{\textsc{range}_0(a)}(X = E, f'))$
  - $f'(t) = \begin{cases} \textsc{build}(t, \textsc{range}_0(a), \textsc{equ}_0(a)) & \text{if } t \in \textsc{rlvt}(E) \text{ and } f(t) = \top_{\mathcal{B}}, \\ f(t) & \text{otherwise.} \end{cases}$
- before applying a binary operator, we refine the filter constraints so that both arguments constrain the same set of tuples; after applying a binary operator, we use the filter constraints to refine the underlying domain properties. That is to say, for any operator $\circledast \in \{\sqcup, \vartriangle, \triangledown\}$, we define $(a_1, f_1) \circledast' (a_2, f_2)$ by $(\textsc{reduce}_0(\textsc{range}_F(f''), a''), f'')$ where:
  - $(a'', f'') = (a_1 \circledast_0 a_2, f_1' \circledast_F f_2')$,
  - $f_i'(t) = \begin{cases} \textsc{build}(t, \textsc{range}_0(a_i), \textsc{equ}_0(a_i)) & \text{if } f_i(t) = \top_{\mathcal{B}} \text{ and } f_{2-i}(t) \neq \top_{\mathcal{B}}, \\ f_i(t) & \text{otherwise.} \end{cases}$

We set $\mathcal{A} = (\mathit{Env}^\sharp, \gamma, \sqcup', \textsc{assign}', \textsc{guard}, \bot, \triangledown', \vartriangle', \textsc{range}, \textsc{equ}, \textsc{reduce})$.

Theorem 3. *The tuple $\mathcal{A}$ is an abstraction.* □

5 Numerical Abstract Domains

Until now, we have only used real numbers. In order to implement numerical abstract domains, we use a finite subset $\mathbb{F}$ of real numbers (such as the floating-point numbers), that is closed under negation. The set $\overline{\mathbb{F}}$ is obtained by enriching the set $\mathbb{F}$ with two extra elements $+\infty$ and $-\infty$ that respectively describe the

reals that are greater (resp. smaller) than the greatest (resp. smallest) element of $\mathbb{F}$. Since the result of a computation on elements of $\overline{\mathbb{F}}$ may be not in $\overline{\mathbb{F}}$, we suppose we are given a function $\lceil\_\rceil$, that provides an upper bound in $\overline{\mathbb{F}}$ to real numbers. The set $\overline{\mathbb{F}}$ is also endowed with a finite widening ramp L [2, Sect. 7.1.2]. For any two elements a and b in $\overline{\mathbb{F}}$ we set $a\triangledown_{\overline{\mathbb{F}}}b = min\{l \in L \cup \{a; +\infty\} \mid max(a,b) \leq l\}$. We now design two families of abstract domains parametrized by the integers q and r that allow tuning the extrapolation strategy.

- The domain $\mathcal{F}_{q,r}$ is the set $\overline{\mathbb{F}} \times \mathbb{Z}$. It is related to $\wp(\mathbb{R})$ via the concretization $\gamma_{\mathcal{F}_{q,r}}$ that maps any pair (e,k) into the set of the reals r such that $|r| \leq e$. The bottom element $\bot_{\mathcal{F}_{q,r}}$ is $(-\infty, 0)$. The binary operators $\sqcup_{\mathcal{F}_{q,r}}$, $\triangledown_{\mathcal{F}_{q,r}}$, and $\vartriangle_{\mathcal{F}_{q,r}}$ are defined as follows:
  - $(a_1,k_1) \sqcup_{\mathcal{F}_{q,r}} (a_2,k_2) = (max(a_1,a_2), 0)$;
  - $(a_1,k_1)\triangledown_{\mathcal{F}_{q,r}}(a_2,k_2) = \begin{cases} (a_1,k_1) & \text{if } a_1 \geq a_2 \\ (a_2,k_1+1) & \text{if } a_2 > a_1 \text{ and } k_1 < q \\ (a_1\triangledown_{\overline{\mathbb{F}}}a_2, 0) & \text{otherwise;} \end{cases}$
  - $(a_1,k_1)\vartriangle_{\mathcal{F}_{q,r}}(a_2,k_2) = \begin{cases} (a_1,k_1) & \text{if } a_1 \leq a_2 \text{ or } k_1 \leq (-r) \\ (a_2, min(k_1,0)-1) & \text{if } a_2 < a_1 \text{ and } k_1 > (-r); \end{cases}$

  A constraint is only widened if it has been unstable q times since its last widening. On the other side, narrowing stops after r iterations.
- The domain $\mathcal{F}^2_{q,r}$ is the Cartesian product between $\mathcal{F}_{0,r}$ and $\mathcal{F}_{q,0}$. It is related to $\wp(\mathbb{R})$ via a concretization map $\gamma_{\mathcal{F}^2_{q,r}}$ that maps elements (a,b) to $\gamma_{\mathcal{F}_{0,r}}(a) \cap \gamma_{\mathcal{F}_{q,0}}(b)$. The element $\bot_{\mathcal{F}^2_{q,r}}$ is the pair $(\bot_{\mathcal{F}_{0,r}}, \bot_{\mathcal{F}_{q,0}})$. Binary operators are defined pairwise, except the widening, that is defined as follows:

  $$((a_1,k_1),(b_1,l_1))\triangledown_{\mathcal{F}^2_{q,r}}((a_2,k_2),(b_2,l_2)) = ((min(a_3,b_3),0),(b_3,l_3)),$$

  where $(a_3,k_3) = (a_1,k_1)\triangledown_{\mathcal{F}_{0,r}}(a_2,k_2)$ and $(b_3,l_3) = (b_1,l_1)\triangledown_{\mathcal{F}_{q,0}}(b_2,l_2)$.

  The elements a and b in the pair (a,b) are intended to describe the same set of reals. Nevertheless, they will be iterated using distinct extrapolation strategies and distinct transfer functions. The element a will be computed via a transfer function $f \in \overline{\mathbb{F}} \to \overline{\mathbb{F}}$, whereas b will be computed via a relaxed transfer function that try to guess the fixpoint of f. The element b is used to refine the element a after widening, in case it becomes more precise. This allows computing arbitrary thresholds during iterations. To ensure termination, it is necessary also to widen b, but this is not done at each iteration.

6 Applications

We propose a general approach to build accurate instantiations for generic extensions in case of linear filtering. We first design a rough abstraction ignoring the historical properties of the input. That way, filters will be considered as if the output only depends on the previous outputs and on only the last global contributions of the last inputs. Then, we formally expand filter equations, so that the overall contribution of each input is exactly described. We use the rough abstraction to approximate the contribution of the floating-point rounding errors

during filter iterations. In the whole section, we use a function $\text{EVAL}^\sharp$ that maps each pair $(E, \rho^\sharp)$, where E is an expression and $\rho^\sharp$ is an abstract environment in $\mathcal{V} \to \mathcal{I}$, to $m \in \overline{\mathbb{F}}$ such that $\forall \rho \in \gamma_{\mathcal{I}}(\rho^\sharp),\ [\![E]\!]_E(\rho) \subseteq [-m; m]$. We also use two natural number parameters q and r, and a real number parameter $\epsilon > 0$.

6.1 Rough Abstraction

To get the rough abstraction, we forget any historical information about the inputs during the filter iterations, so that each output is computed as an affine combination of the previous outputs. The constant term is an approximation of the contributions of both the previous inputs and the floating-point rounding errors. Human intervention is only required during the design of the domain to discover the form of the properties that are relevant, and the way these properties are propagated throughout a filter iteration.

Simplified High Passband Filter. A simplified high passband filter relates an input stream E_n to an output stream defined by $S_{n+1} = aS_n + E_n$.

Theorem 4. *Let $\varepsilon_a \geq 0$, $D \geq 0$, $m \geq 0$, a, X and Z be real numbers such that $|X| \leq D$ and $aX - (m + \varepsilon_a|X|) \leq Z \leq aX + (m + \varepsilon_a|X|)$.*
Then we have:

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

We define two maps $\phi_1 \in \mathbb{F}^2 \times \overline{\mathbb{F}}^2 \to \overline{\mathbb{F}}$ and $\phi_2 \in \mathbb{F}^2 \times \overline{\mathbb{F}} \to \overline{\mathbb{F}}$ by:

$$\begin{cases} \phi_1(a, \varepsilon_a, m, D) = \lceil (|a| + |\varepsilon_a|)D + m \rceil, \\ \phi_2(a, \varepsilon_a, m) = \begin{cases} \left\lceil \frac{m(1+\epsilon)}{1-(|a|+|\varepsilon_a|)} \right\rceil & \text{if } |a| + |\varepsilon_a| < 1 \\ +\infty & \text{otherwise.} \end{cases} \end{cases}$$

We derive the following extension:

- $\mathcal{T}_{r_1} = (\mathcal{V} \times \mathbb{F}^2)$ and $\mathit{info} = \mathbb{F}^2 \times \overline{\mathbb{F}}$;
- $(\mathcal{B}_{r_1}, \bot_{r_1}, \sqcup_{\mathcal{B}_{r_1}}, \nabla_{\mathcal{B}_{r_1}}, \triangle_{\mathcal{B}_{r_1}}) = (\mathcal{F}^2_{q,r}, \bot_{\mathcal{F}^2_{q,r}}, \sqcup_{\mathcal{F}^2_{q,r}}, \nabla_{\mathcal{F}^2_{q,r}}, \triangle_{\mathcal{F}^2_{q,r}})$;
- $\gamma_{\mathcal{B}_{r_1}}((X, a, \varepsilon_a), e) = \{\rho \in \mathit{Env} \mid \rho(X) \in \gamma_{\mathcal{F}^2_{q,r}}(e)\}$;
- RLVT_{r_1} maps each expression E to the set of the tuples (X, a, ε_a), such that E matches $I \times X + E'$ with $I \subseteq [\frac{1}{2}; 1[$ and $I = [a - \varepsilon_a; a + \varepsilon_a]$;
- $\text{PT}_{r_1}(Z = I \times X + E', (X, a, \varepsilon_a), \rho^\sharp) = ((Z, a, \varepsilon_a), (a, \varepsilon_a, m))$ where $m = \text{EVAL}^\sharp(E', \rho^\sharp)$;[2]
- $\delta_{r_1}((a, \varepsilon_a, m), ((D_1, k_1), (D_2, k_2))) = ((r_1, 0), (r_2, 0))$ where $r_1 = \phi_1(a, \varepsilon_a, m, D_1)$ and $r_2 = \mathit{max}(\phi_1(a, \varepsilon_a, m, D_2), \phi_2(a, \varepsilon_a, m))$;
- $\text{BUILD}_{r_1}((X, a, \varepsilon_a), \rho^\sharp, \mathcal{R}) = ((m, 0), (m, 0))$, with $m = \text{EVAL}^\sharp(X, \rho^\sharp)$;
- $\text{TO}_{r_1}((X, a, \varepsilon_a), e) = \begin{cases} Y \mapsto \gamma_{\mathcal{F}^2_{q,r}}(e) & \text{if } X = Y \\ Y \mapsto \mathbb{R} & \text{otherwise.} \end{cases}$

A simplified second order filter relates an input stream E_n to an output stream defined by:

$$S_{n+2} = aS_{n+1} + bS_n + E_{n+2}.$$

Thus we experimentally observe, in Fig. 4, that starting with $S_0 = S_1 = 0$ and provided that the input stream is bounded, the pair (S_{n+2}, S_{n+1}) lies in an ellipsoid. Moreover, this ellipsoid is attractive, which means that an orbit starting out of this ellipsoid, will get closer of it. This behavior is explained by Thm. 5.

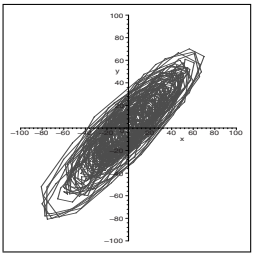

Fig. 4. Orbit.

Simplified Second Order Filter.

Theorem 5. *Let a, b, $\varepsilon_a \geq 0$, $\varepsilon_b \geq 0$, $K \geq 0$, $m \geq 0$, X, Y be real numbers, such that $a^2 + 4b < 0$, and $X^2 - aXY - bY^2 \leq K$. Let Z be a real number such that: $aX + bY - (m + \varepsilon_a|X| + \varepsilon_b|Y|) \leq Z \leq aX + bY + (m + \varepsilon_a|X| + \varepsilon_b|Y|)$. Let δ be $2\frac{\varepsilon_b + \varepsilon_a\sqrt{-b}}{\sqrt{-(a^2+4b)}}$, we have:*

1. $|X| \leq 2\sqrt{\frac{bK}{a^2+4b}}$ *and* $|Y| \leq 2\sqrt{\frac{-K}{a^2+4b}}$;
2. $Z^2 - aZX - bX^2 \leq \left((\sqrt{-b} + \delta)\sqrt{K} + m\right)^2$;
3. $\begin{cases} \sqrt{-b} + \delta < 1 \\ K \geq \left(\frac{m}{1-\sqrt{-b}-\delta}\right)^2 \end{cases} \implies Z^2 - aZX - bX^2 \leq K.$

□

We define two maps $\psi_1 \in \mathbb{F}^4 \times \overline{\mathbb{F}}^2 \to \overline{\mathbb{F}}$ and $\psi_2 \in \mathbb{F}^4 \times \overline{\mathbb{F}} \to \overline{\mathbb{F}}$ by:

$$\begin{cases} \psi_1(a, \varepsilon_a, b, \varepsilon_b, m, D) = \begin{cases} \left\lceil \left(\left(\sqrt{-b} + \delta(a, \varepsilon_a, b, \varepsilon_b)\right)\sqrt{D} + m \right)^2 \right\rceil & \text{if } a^2 + 4b < 0 \\ +\infty & \text{otherwise} \end{cases} \\ \psi_2(a, \varepsilon_a, b, \varepsilon_b, m) = \begin{cases} \left\lceil \left(\frac{(1+\epsilon)m}{1-\sqrt{-b}-\delta(a,\varepsilon_a,b,\varepsilon_b)} \right)^2 \right\rceil & \text{if } \begin{cases} a^2 + 4b < 0 \\ \sqrt{-b} + \delta(a, \varepsilon_a, b, \varepsilon_b) < 1 \end{cases} \\ +\infty & \text{otherwise.} \end{cases} \\ \text{where } \delta(a, \varepsilon_a, b, \varepsilon_b) = 2\frac{|\varepsilon_b| + |\varepsilon_a|\sqrt{-b}}{\sqrt{-(a^2+4b)}} \end{cases}$$

We derive the following abstract extension:

- $\mathcal{T}_{r_2} = (\mathcal{V}^2 \times \mathbb{F}^4)$ and $info = \mathbb{F}^4 \times \overline{\mathbb{F}}$;
- $(\mathcal{B}_{r_2}, \bot_{r_2}, \sqcup_{\mathcal{B}_{r_2}}, \nabla_{\mathcal{B}_{r_2}}, \triangle_{\mathcal{B}_{r_2}}) = (\mathcal{F}^2_{q,r}, \bot_{\mathcal{F}^2_{q,r}}, \sqcup_{\mathcal{F}^2_{q,r}}, \nabla_{\mathcal{F}^2_{q,r}}, \triangle_{\mathcal{F}^2_{q,r}})$;
- $\gamma_{\mathcal{B}_{r_2}}((X, Y, a, \varepsilon_a, b, \varepsilon_b), e)$ is given by the set of environments ρ that satisfies:
$$(\rho(X))^2 - a\rho(X)\rho(Y) - b(\rho(Y))^2 \leq max(\gamma_{\mathcal{F}^2_{q,r}}(e));$$
- RLVT$_{r_2}$ maps each expression E to the set of tuples $(X, Y, a, \varepsilon_a, b, \varepsilon_b)$, such that E matches $[a-\varepsilon_a; a+\varepsilon_a] \times X + [b-\varepsilon_b; b+\varepsilon_b] \times Y + E'$ with $a^2 + 4b < 0$;

- $\text{PT}_{r_2}(Z = [a-\varepsilon_a; a+\varepsilon_a] \times X + [b-\varepsilon_b; b+\varepsilon_b] \times Y + E', (X, Y, a, \varepsilon_a, b, \varepsilon_b), \rho^\sharp))$ is given[2] by $((Z, X, a, \varepsilon_a, b, \varepsilon_b), (a, \varepsilon_a, b, \varepsilon_b, m))$, where $m = \text{EVAL}^\sharp(E', \rho^\sharp)$;
- $\delta_{r_2}((a, \varepsilon_a, b, \varepsilon_b, m), ((D_1, k_1), (D_2, k_2))) = ((r_1, 0), (r_2, 0))$ where $r_1 = \psi_1(a, \varepsilon_a, b, \varepsilon_b, m, D_1)$ and $r_2 = max(\psi_1(a, \varepsilon_a, b, m, D_2), \psi_2(a, \varepsilon_a, b, \varepsilon_b, m))$;
- $\text{BUILD}_{r_2}((X, Y, a, \varepsilon_a, b, \varepsilon_b), \rho^\sharp, \mathcal{R})(X) = ((m, 0), (m, 0))$
  where $m = \begin{cases} \lceil |1 - a - b|x^2 \rceil & \text{if } (X, Y) \in \mathcal{R} \\ \lceil x^2 + |a|xy + |b|y^2 \rceil & \text{otherwise} \\ \text{with } x = \text{EVAL}^\sharp(X, \rho^\sharp) \text{ and } y = \text{EVAL}^\sharp(Y, \rho^\sharp); \end{cases}$
- $\text{TO}_{r_2}(X, Y, a, \varepsilon_a, b, \varepsilon_b)(e) = \begin{cases} V \mapsto [-m; m] & \text{if } X = V \text{ and } a^2 + 4b < 0 \\ V \mapsto \mathbb{R} & \text{otherwise} \end{cases}$
  where $m = \left\lceil 2\sqrt{\frac{max(\gamma_{\mathcal{F}^2_{q,r}}(e)) \times b}{a^2+4b}} \right\rceil$.

6.2 Formal Expansion

We design a more accurate abstraction by formally expanding the definition of the output in the real field. The overall contribution of rounding errors is then over-approximated using the rough abstraction, while the contribution of each input is exactly described. The obtained domain is history-aware: concretization uses an existential quantification over past inputs, which forbids precise abstract intersection and narrowing.

High Passband Filter

Theorem 6. *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. $S_n = \alpha^n i + E_n - \alpha^n E_0 + \Sigma_{l=1}^{n-1}(\alpha - 1)\alpha^{l-1} E_{n-l}$
2. $|S_n| \leq |\alpha|^n |i| + (1 + |\alpha|^n + |1 - \alpha^{n-1}|)m$;
3. $|S_n| \leq 2m + |i|$. □

We derive the following abstract extension:

- $\mathcal{T}_{d_1} = \mathcal{V}^2 \times \mathbb{F}$, $\mathcal{B}_{d_1} = \mathcal{F}_{q,r} \times \mathcal{F}_{q,r} \times \mathcal{F}^2_{q,r}$, and $\text{INFO}_{d_1} = \mathbb{F} \times \overline{\mathbb{F}} \times \overline{\mathbb{F}}$;
- $(\bot_{d_1}, \sqcup_{\mathcal{B}_{d_1}}, \triangledown_{\mathcal{B}_{d_1}})$ are defined componentwise;

[2] If $[t \to fst(\text{PT}_{r_i}(Z = E, t, \rho^\sharp))]$ is not injective, we take a subset of $\text{RLVT}_{r_i}(E)$.

- $\gamma_{\mathcal{B}_{d_1}}((S',E',\alpha),(a,b,c))$ is given by
$$\left\{ \rho \in Env \;\middle|\; \begin{array}{l} \exists \varepsilon \in \gamma_{\mathcal{F}^2_{q,r}}(c),\ \exists S_0 \in \gamma_{\mathcal{F}_{q,r}}(a), \\ \exists n \in \mathbb{N}, \exists (E_i)_{0\le i\le n} \in (\gamma_{\mathcal{F}_{q,r}}(b))^{n+1} \text{ such that} \\ \rho(E') = E_n \text{ and} \\ \rho(S') = \varepsilon + \alpha^n S_0 + E_n - \alpha^n E_0 + \Sigma_{l=1}^{n-1}(\alpha-1)\alpha^{l-1}E_{n-l} \end{array} \right\};$$
- RLVT_{d_1} maps each expression E to the set of the tuples (X_α, X_-, α), such that E matches
$[\alpha-\varepsilon_\alpha;\alpha+\varepsilon_\alpha]\times X_\alpha + [1-\varepsilon_+;1+\varepsilon_+]\times X_+ + [-1+\varepsilon_-;\varepsilon_- -1]\times X_- + [-\varepsilon,\varepsilon]$
where $\varepsilon_\alpha \ge 0$, $\varepsilon_+ \ge 0$, $\varepsilon_- \ge 0$, $\varepsilon \ge 0$ and $[\alpha-\varepsilon_\alpha;\alpha+\varepsilon_\alpha] \subseteq [\frac{1}{2};1[$; the pair $\text{PT}_{r_1}(Z = E, (X_\alpha, X_-, \alpha), \rho^\sharp)$ is given by $((Z, X_+, \alpha), (\alpha, m_+, m'))$, where $m_+ = \text{EVAL}^\sharp(X_+, \rho^\sharp)$ and $m' = \text{EVAL}^\sharp(E + \{-\alpha\}\times X_\alpha + (-X_+) + X_-, \rho^\sharp)$;
- $\delta_{d_1}((\alpha,m,\varepsilon),(a,b,c)) = (a, b \sqcup_{\mathcal{F}_{q,r}} (m,0), \delta_{r_1}((\alpha,0,\varepsilon),c))$;
- $\text{BUILD}_{d_1}((X_\alpha, X_+, \alpha), \rho^\sharp, \mathcal{R}) = ((m_\alpha, 0), (m_+, 0), ((0,0),(0,0)))$ where $m_\alpha = \text{EVAL}^\sharp(X_\alpha, \rho^\sharp)$ and $m_+ = \text{EVAL}^\sharp(X_+, \rho^\sharp)$;
- $\text{TO}_{d_1}((X_\alpha, X_+, \alpha),(a,b,c)) = \begin{cases} Y \mapsto [-m;m] & \text{if } X = Y \text{ and } \alpha \in [\frac{1}{2};1[\\ Y \mapsto \mathbb{R} & \text{otherwise,} \end{cases}$
where $m = \left\lceil max(\gamma_{\mathcal{F}_{q,r}}(a)) + 2max(\gamma_{\mathcal{F}_{q,r}}(b)) + max(\gamma_{\mathcal{F}^2_{q,r}}(c)) \right\rceil$;
- $\forall a, b \in \mathcal{B}_{d_1},\ a \triangle_{\mathcal{B}_{d_1}} b = a$.

Second Order Filter

Theorem 7. *Let a, b, c, d, e, i_0, and i_1 be real numbers, such that $-1 < b < 0$ and $a^2 + 4b < 0$. Let (E_n) be a sequence of real numbers. We assume there exists $m \in \mathbb{F}$ such that $\forall k \in \mathbb{N}$, we have $E_k \in [-m;m]$, and $\forall k \in \{1;2\}$, we have $i_k \in [-m;m]$. We define the sequence (C_n) as follows:*

$$\begin{cases} S_0 = i_0,\ S_1 = i_1, \\ S_{n+2} = aS_{n+1} + bS_n + cE_{n+2} + dE_{n+1} + eE_n \end{cases}$$

Let $(A_i)_{i\in\mathbb{N}}$, $(B_i)_{i\in\mathbb{N}}$, and $(C_i^j)_{i,j\in\mathbb{N}}$ be real coefficient families such that[3]*:*

$$\forall n \in \mathbb{N},\ S_n = A_n i_0 + B_n i_1 + \Sigma_{i=0}^n C_i^n E_i.$$

For any pair $(n,N) \in \mathbb{N}$ such that $N > 3$ and $n \ge N$, we define the residue R_n^N by $S_n - \Sigma_{i=n-N+1}^n C_i^n E_i$. We have:

1. $\forall N > 3, n \ge N+2,\ R_n^N = aR_{n-1}^N + bR_{n-2}^N + \varepsilon_n^N$
where $\varepsilon_n^N = aC_{n-N}^{n-1}E_{n-N} + bC_{n-1-N}^{n-2}E_{n-1-N} + bC_{n-N}^{n-2}E_{n-N}$;
2. $\forall N > 3, n \in \mathbb{N},\ |S_n| \le (max(S_\le^N(i_1 = i_2), S_\infty^N(i_1 = i_2))).m$
where
   - $S_\le^N(\text{B}) = max\{(\Sigma_{k=0}^p |C_k^p|) + init(p,\text{B}) \mid p \in [\![0; N+2]\!]\}$
   - $S_\infty^N(\text{B}) = \left(\Sigma_{i=0}^{N-1} |C_2^{i+2}| + 2\sqrt{\frac{-bmax(K_\infty^N, K_0^N(\text{B}))}{a^2+4b}} \right)$

[3] We omit the explicit recursive definition of these coefficients, for conciseness.

– $\forall n \geq 2,\ \mathit{init}(n, \mathrm{B}) = \begin{cases} |A_n| + |B_n| & \mathit{if}\ (\mathit{not}(\mathrm{B})) \\ |A_n + B_n| & \mathit{otherwise,} \end{cases}$

– $K^N_\infty = \left(\frac{|aC_2^{1+N} + bC_2^N| + |bC_2^{1+N}|}{1-\sqrt{-b}} \right)^2$ *and* $K^N_0(\mathrm{B}) = X^2 + |a|XY - bY^2$,

where $\begin{cases} X = (\mathit{init}(N+2, \mathrm{B})) + \Sigma_{i=0}^{N+2} \left| C_i^{N+2} \right| \\ Y = (\mathit{init}(N+1, \mathrm{B})) + \Sigma_{i=0}^{N+1} \left| C_i^{N+1} \right|. \end{cases}$ □

We set $N \in \mathbb{N}$ and define the map ψ_∞ that maps each tuple $(\mathrm{B}, a, b, c, d, e) \in \mathcal{B} \times \mathbb{F}^5$ into $\left\lceil \mathit{max}(S^N_\leq, S^N_\infty(\mathrm{B})) \right\rceil$ where $S^N_\leq$ and S^N_∞ are defined as in Thm. 7.

We derive the following abstract extension:

– $\mathcal{T}_{d_2} = \mathcal{V}^4 \times \mathbb{F}$, $\mathcal{B}_{d_2} = \mathcal{B} \times \mathcal{F}_{q,r} \times \mathcal{F}^2_{q,r}$, and $\mathrm{INFO}_{d_2} = \overline{\mathbb{F}} \times \overline{\mathbb{F}}$;
– $(\bot_{d_2}, \sqcup_{\mathcal{B}_{d_2}}, \triangledown_{\mathcal{B}_{d_2}})$ are defined componentwise.
– $\gamma_{\mathcal{B}_{d_2}}((S'', S', E'', E', a, b, c, d, e), (\mathrm{B}, m, K_\varepsilon))$ is given by:

$$\left\{ \rho \in \mathit{Env} \;\middle|\; \begin{array}{l} \exists \varepsilon_X, \varepsilon_Y, \text{ such that } \varepsilon_X^2 - a\varepsilon_X\varepsilon_Y - b\varepsilon_Y^2 \leq \mathit{max}(\gamma_{\mathcal{F}^2_{q,r}}(K_\varepsilon)) \\ \exists n \in \mathbb{N}, \exists (E_i)_{-2 \leq i \leq n+1} \in (\gamma_{\mathcal{F}_{q,r}}(m))^{n+4} \text{ such that} \\ \rho(S'') = \varepsilon_X + A_{n+1}E_{-2} + B_{n+1}E_{-1} + \Sigma_{i=0}^{n+1} C_i^{n+1} E_i \\ \rho(S') = \varepsilon_Y + A_n E_{-2} + B_n E_{-1} + \Sigma_{i=0}^{n} C_i^{n} E_i \\ \rho(E'') = E_{n+1},\ \rho(E') = E_n,\ [\mathrm{B} \implies E_{-1} = E_{-2}] \end{array} \right\}$$

– $\mathrm{RLVT}_{d_2}(E)$ is the set of tuples $t = (X_a, X_b, X_d, X_e, a, b, c, d, e)$, such that there exists $X_c \in \mathcal{V}$ such that E matches $\Sigma_{i \in \{a;b;c;d;e\}} [i - \varepsilon_i; i + \varepsilon_i] \times X_i + [-\varepsilon, \varepsilon]$ where $\forall i \in \{a; b; c; d; e\}, \varepsilon_i \geq 0$, $\varepsilon > 0$, $a^2 + 4b < 0$, and $-1 < b < 0$, then the pair $\mathrm{PT}_{r_1}(Z = E, t, \rho^\sharp)$ is given by $(t', (\mathrm{EVAL}^\sharp(X_c, \rho^\sharp), \mathrm{EVAL}^\sharp(E', \rho^\sharp)))$ where $t' = (Z, X_a, X_c, X_d, a, b, c, d, e)$ and $E' = E + \Sigma_{i \in \{a;b;c;d;e\}} \{-i\} \times X_i$;
– $\delta_{d_2}((m, \varepsilon), (\mathrm{A}, i_E, i_\varepsilon)) = (\mathrm{A}, i_E \sqcup_{\mathcal{F}_{q,r}} (m, 0), \delta_{r_2}((a, 0, b, 0, \varepsilon), i_\varepsilon))$;
– $\mathrm{BUILD}_{d_2}((X_a, X_b, X_d, X_e, a, b, c, d, e), \rho^\sharp, \mathcal{R}) = (\mathrm{B}, (m, 0), ((0, 0), (0, 0)))$ with $\mathrm{B} = ((X_a, X_b) \in \mathcal{R})$ and $m = \mathit{max}\{\mathrm{EVAL}^\sharp(X, \rho^\sharp) \mid \forall X \in \{X_a, X_b, X_d, X_e\}\}$;
– $\mathrm{TO}_{d_2}((X_a, X_b, X_d, X_e, a, b, c, d, e), (\mathrm{A}, i_E, i_\varepsilon)) = \begin{cases} Y \mapsto [-m; m] & \text{if } \begin{cases} X = Y \\ \frac{a^2}{4} < -b \end{cases} \\ Y \mapsto \mathbb{R} & \text{otherwise} \end{cases}$

where $m = \left\lceil \mathit{max}(\gamma_{\mathcal{F}_{q,r}}(i_E))\psi_\infty(\mathrm{A}, a, b, c, d, e) + 2\sqrt{\frac{\mathit{max}(\gamma_{\mathcal{F}^2_{q,r}}(i_\varepsilon))b}{a^2+4b}} \right\rceil$

– $\forall a, b \in \mathcal{B}_{d_2},\ a \triangle_{\mathcal{B}_{d_2}} b = a$.

7 Benchmarks

We tested our framework with the same program as in [2]. This program is 132,000 lines of C long with macros (75 kLOC after preprocessing) and has about 10,000 global variables. The underlying domain is the same as in [2] (i.e., we use intervals [3], octagons [9], decision trees, except the ellipsoid domain which corresponds to the rough abstraction of second order digital filters). We perform three analyses with several levels of accuracy for filter abstraction. First, we use no filter abstraction; then we only use the rough abstraction; finally we

	no filter abstraction	rough filter abstraction	accurate filter abstraction
iteration number	69	120	72
average time by iteration	48.0 s.	58.4 s.	61.7 s.
analysis time	3313 s.	7002 s.	4444 s.
warnings	639	10	6

Fig. 5. Some statistics.

use the most accurate abstraction. For each of these analyses, we report in Fig. 5 the analysis time, the number of iterations for the main loop, and the number of warnings (in polyvariant function calls). These results have been obtained on a 2.8 GHz, 4 Gb RAM PC.

8 Conclusion

We have proposed a highly generic framework to analyze programs with digital filtering. We have also given a general approach to instantiate this framework in the case of linear filtering. We have enriched an existing analyzer, and obtained results that were far beyond our expectations. As a result, we solved nearly all remaining warnings when analyzing an industrial program of 132,000 lines of C: we obtain a sufficiently small number of warnings to allow manual inspection, and we discovered they could be eliminated without altering the functionality of the application by changing only three lines of code.

Linear filtering is widely used in the context of critical embedded software, so we believe that this framework is crucial to achieve full certification of the absence of runtime error in such programs.

Acknowledgments. We deeply thank the anonymous referees. We also thank Charles Hymans, Francesco Logozzo and each member of the magic team: Bruno Blanchet, Patrick Cousot, Radhia Cousot, Laurent Mauborgne, Antoine Miné, David Monniaux, and Xavier Rival.

References

1. B. Blanchet, P. Cousot, R. Cousot, J. Feret, L. Mauborgne, A. Miné, D. Monniaux, and X. Rival. Design and implementation of a special-purpose static program analyzer for safety-critical real-time embedded software, invited chapter. In *The Essence of Computation: Complexity, Analysis, Transformation. Essays Dedicated to Neil D. Jones*, LNCS 2566. Springer-Verlag, 2002.
2. B. Blanchet, P. Cousot, R. Cousot, J. Feret, L. Mauborgne, A. Miné, D. Monniaux, and X. Rival. A static analyzer for large safety-critical software. In *Proc. PLDI'03*. ACM Press, 2003.
3. P. Cousot and R. Cousot. Abstract interpretation: a unified lattice model for static analysis of programs by construction or approximation of fixpoints. In *Proc. POPL'77*. ACM Press, 1977.

4. P. Cousot and R. Cousot. Abstract interpretation frameworks. *Journal of logic and computation*, 2(4), August 1992.
5. P. Cousot and N. Halbwachs. Automatic discovery of linear restraints among variables of a program. In *Proc. POPL'78*. ACM Press, 1978.
6. M. Karr. Affine relationships among variables of a program. *Acta Inf.*, 1976.
7. A. A. Lamb, W. Thies, and S.P.Amarasinghe. Linear analysis and optimisation of stream programs. In *Proc. PLDI'03*. ACM Press, 2003.
8. X. Leroy, D. Doligez, J. Garrigue, D. Rémy, and J. Vouillon. The Objective Caml system, documentation and user's manual. Technical report, INRIA, 2002.
9. A. Miné. The octagon abstract domain. In *Proc. WCRE'01(AST'01)*, IEEE, 2001.
10. A. Miné. Relational abstract domains for the detection of floating-point run-time errors. In *Proc. ESOP'04*, LNCS. Springer, 2004. © Springer.