The Arithmetic-Geometric Progression Abstract Domain*

Jérôme Feret

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

Abstract We present a new numerical abstract domain. This domain automatically detects and proves bounds on the values of program variables. For that purpose, it relates variable values to a clock counter. More precisely, it bounds these values with the i-th iterate of the function $[X \mapsto \alpha \times X + \beta]$ applied on M, where i denotes the clock counter and the floating-point numbers α, β, and M are discovered by the analysis. Such properties are especially useful to analyze loops in which a variable is iteratively assigned with a barycentric mean of the values that were associated with the same variable at some previous iterations. Because of rounding errors, the computation of this barycenter may diverge when the loop is iterated forever. Our domain provides a bound that depends on the execution time of the program.

Keywords: Abstract Interpretation, static analysis, numerical domains.

1 Introduction

A critical synchronous real-time system (as found in automotive, aeronautic, and aerospace applications) usually consists in iterating a huge loop. Because practical systems do not run forever, a bound on the maximum iteration number of this loop can be provided by the end-user or discovered automatically. The full certification of such a software may require relating variable values to the number of iterations of the main loop. It is especially true when using floating-point numbers. Some computations that are stable when carried out in the real field, may diverge because of the rounding errors. Rounding errors are accumulated at each iteration of the loop. When expressions are linear and when the evaluation of expressions does not overflow, the rounding errors at each loop iteration are usually proportional to the value of the variables. Thus the overall contribution of rounding errors can be obtained by iterating a function of the form $[X \mapsto \alpha \times X + \beta]$. Then by using the maximum number of iterations we can infer bounds on the values that would normally have diverged in the case of an infinite computation.

We propose a new numerical abstract domain that associates with each variable the corresponding coefficients α and β and the starting value M. This

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

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

Figure 1. Syntax.

framework was fully implemented in OCaml [7] and plugged into an existing analyzer [1,2]. We use this analyzer for certifying a family of critical embedded softwares, programs ranging from 70,000 to 379,000 lines of C.

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 numerical abstract predomain that relates arithmetic-geometric constraints with sets of real numbers. In Sect. 5, we enrich an existing analysis so that it can deal with arithmetic-geometric constraints. In Sect. 6, we refine our analysis to deal with more complex examples. In Sect. 7, we report the impact of the arithmetic-geometric progression domain 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 that are interpreted between the concrete and an abstract level. Data structures have been translated by using a finite set of abstract cells (see [2, Sect. 6.1]). Non-deterministic branching over-approximates all the memory accesses (array accesses, pointer dereferencing) that are not fully statically resolved. Furthermore, floating-point expressions have been conservatively approximated by linear forms with real interval coefficients. These linear forms include both the rounding errors and some expression approximations (see [9]). We also suppose that the occurrence of runtime errors (such as floating-point overflows) can be described by interval constraints on the memory state.

Let $\mathcal{V}$ be a finite set of variables. Let $\texttt{clock} \notin \mathcal{V}$ be an extra variable which is associated with the clock counter. The clock counter is explicitly incremented when a command **tick** is executed. The system stops when the clock counter overflows a maximum value which is defined by the end-user. 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* ($\rho \in \mathcal{V} \cup \{\texttt{clock}\} \to \mathbb{R}$) denotes a memory state. It maps each variable, including the clock variable, to a real number. We denote by *Env* the set of all environments. The semantics of an expression E is a function $(\!|E|\!) \in Env \to I$ mapping each environment to an interval. Given a maximum value $\texttt{mc}$ for the clock, the semantics of a program P is a function $[\![P]\!]_{\texttt{mc}} \in Env \to \wp(Env)$ mapping each environment ρ to the set of the environments that can be reached when applying the program P starting from the environment ρ. Returning a set of environments allows the description of both non-determinism and program halting (when the

$$(\!|I|\!)(\rho) = I,\ (\!|V|\!)(\rho) = \{\rho(V)\}$$
$$(\!|I \times V + E|\!)(\rho) = \{b \times \rho(V) + a \mid a \in (\!|E|\!)(\rho), b \in I\}$$
$$[\![V = E]\!]_{\texttt{mc}}(\rho) = \{\rho[V \mapsto x] \mid x \in (\!|E|\!)(\rho)\}$$
$$[\![\mathbf{skip}]\!]_{\texttt{mc}}(\rho) = \{\rho\}$$
$$[\![\mathbf{tick}]\!]_{\texttt{mc}}(\rho) = \begin{cases} \{\rho[\texttt{clock} \mapsto \rho(\texttt{clock}) + 1]\} & \text{if } \rho(\texttt{clock}) < \texttt{mc} \\ \emptyset & \text{otherwise} \end{cases}$$
$$[\![\mathbf{if}\ (V \geq 0)\ \{P_1\}\ \mathbf{else}\ \{P_2\}]\!]_{\texttt{mc}}(\rho) = \begin{cases} [\![P_1]\!]_{\texttt{mc}}(\rho) & \text{if } \rho(V) \geq 0 \\ [\![P_2]\!]_{\texttt{mc}}(\rho) & \text{otherwise} \end{cases}$$
$$[\![\mathbf{while}\ (V \geq 0)\ \{P\}]\!]_{\texttt{mc}}(\rho) = \{\rho' \in \mathit{Inv} \mid \rho'(V) < 0\}$$
$$\text{where } \mathit{Inv} = \text{lfp}\left(X \mapsto \{\rho\} \cup \left(\bigcup\{[\![P]\!]_{\texttt{mc}}(\rho') \mid \rho' \in X, \rho'(V) \geq 0\}\right)\right)$$
$$[\![P_1; P_2]\!]_{\texttt{mc}}(\rho) = \bigcup\{[\![P_2]\!]_{\texttt{mc}}(\rho') \mid \rho' \in [\![P_1]\!]_{\texttt{mc}}(\rho)\}$$

Figure 2. Concrete semantics.

clock has reached its maximum value). The functions $(\!|\_|\!)$ and $[\![\_]\!]_{\texttt{mc}}$ are 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[1] $f \in \wp(\mathit{Env}) \to \wp(\mathit{Env})$. Nevertheless, such a fixpoint is usually not computable, so we give a decidable approximate semantics in the next section.

We describe two toy examples.

Example 1. The first example iterates the computation of a barycentric mean: at each loop iteration, a variable is updated with a barycentric mean among its current value and two previous values.

$$\begin{array}{l} V = \mathbb{R};\ B_1 = \mathbb{R};\ B_2 = \mathbb{R};\ X = 0;\ Y = 0;\ Z = 0; \\ \mathbf{while}\ (V \geq 0)\ \{ \\ \quad V = \mathbb{R};\ B_1 = \mathbb{R};\ B_2 = \mathbb{R}; \\ \quad \mathbf{if}\ (B_1 \geq 0)\ \{Z = Y;\ Y = X\}\ \mathbf{else}\ \{\mathbf{skip}\}; \\ \quad \mathbf{if}\ (B_2 \geq 0)\ \{ \\ \quad\quad X = I;\ Y = I;\ Z = I\} \\ \quad \mathbf{else}\ \{ \\ \quad\quad X = I_X \times X + I_Y \times Y + I_Z \times Z + I_\varepsilon\}; \\ \quad \mathbf{tick}\} \end{array}$$

where $I \in \mathcal{I},\ \varepsilon_i > 0$ for any $i \in \{X; Y; Z; 0\},\ 0 < \alpha < 0.5$,
$I_X = [1 - 2 \times \alpha - \varepsilon_X; 1 - 2 \times \alpha + \varepsilon_X],\ I_Y = [\alpha - \varepsilon_Y; \alpha + \varepsilon_Y]$,
$I_Z = [\alpha - \varepsilon_Z; \alpha + \varepsilon_Z]$, and $I_\varepsilon = [-\varepsilon_0; \varepsilon_0]$.

More precisely, initialization values range in the interval I. The parameter α is a coefficient of the barycentric mean. The parameters ε_X, ε_Y, and ε_Z encode the rounding errors relative respectively to the variables X, Y, and Z in the computation of the barycentric mean. The parameter ε_0 encodes the absolute rounding error. The three variables X, Y, and Z allow the recursion (X is associated with

[1] In fact, we only use the monotonicity of f.

the current value, Y is associated with the last selected value and Z is associated with the previous selected value) and the three variables V, B_1, and B_2 allow non-deterministic boolean control. The variable V allows stopping the loop iteration. The variable B_1 allows the selection of a recursive value which consists in shifting the variables X, Y, and Z. The variable B_2 allows the choice between a reinitialization or an iteration step: a reinitialization step consists in assigning the variables X, Y, and Z with some random values in the interval I, whereas an iteration step consists in updating the variable X with the barycentric mean between its current value and the last two selected values. Because of rounding errors, the value associated with the variable X cannot be bounded without considering the clock. Therefore, we can prove that this value is bounded by $[X \mapsto ((1+\varepsilon_X+\varepsilon_Y+\varepsilon_Z)\times X)+\varepsilon_0]^{(\mathtt{mc})}(M_I)$, where M_I is the least upper bound of the set $\{|x| \mid x \in I\}$. This bound can be discovered using the arithmetic-geometric domain presented in this paper. It is worth noting that the domains that deal with digital stream processing [6] do not help because the value of the variable Y is not necessarily the previous value of the variable X: such domains can only approximate relations of the form $o_n = f(o_{n-1},...,o_{n-p},i_{n-1},...,i_{n-q})$ where (i_n) is the input stream and (o_n) is the output stream.

Example 2. The second example iterates a loop where a floating point is first divided by a coefficient $\alpha > 0$ and then multiplied by the coefficient α.

$$\begin{array}{l}
V = \mathbb{R};\ B_1 = \mathbb{R};\ B_2 = \mathbb{R};\ X = 0;\\
\textbf{while}\ (V \geq 0)\ \{\\
\quad V = \mathbb{R};\ B_1 = \mathbb{R};\ B_2 = \mathbb{R};\\
\quad \textbf{if}\ (B_1 \geq 0)\ \{X = I_1\}\ \textbf{else}\ \{\textbf{skip}\};\\
\quad X = [\frac{1}{\alpha} - \varepsilon_1; \frac{1}{\alpha} + \varepsilon_1] \times X + [-\varepsilon_2; \varepsilon_2];\\
\quad \textbf{if}\ (B_2 \geq 0)\ \{X = I_2\}\ \textbf{else}\ \{\textbf{skip}\};\\
\quad X = [\alpha - \varepsilon_3; \alpha + \varepsilon_3] \times X + [-\varepsilon_4; \varepsilon_4];\\
\quad \textbf{tick}\}
\end{array}$$

where $\varepsilon_i > 0$, for any $i \in \{1; 2; 3; 4\}$, $\alpha > 0$, and $I_1, I_2 \in \mathcal{I}$.

More precisely, initialization values range in the intervals I_1 and I_2. The parameter α is a coefficient of the example. The parameters ε_1 and ε_3 encode relative rounding errors and the parameters ε_2 and ε_4 encode absolute rounding errors. The variable X contains the value that is divided and multiplied. The three variables V, B_1, and B_2 allow boolean control. The variable V allows stopping the loop iteration. The variable B_1 allows the reinitialization of the variable X before the division, the variable B_2 allows its reinitialization before the multiplication. Because of rounding errors, the value associated with the variable X cannot be bounded without considering the clock. Therefore, we can prove that this value is bounded by $[X \mapsto (1+a)\times X + b]^{(\mathtt{mc})}(M_I)$ where $a = \alpha \times \varepsilon_1 + \frac{1}{\alpha} \times \varepsilon_3 + \varepsilon_1 \times \varepsilon_3$ and $b = \varepsilon_2 \times (\alpha + \varepsilon_3) + \varepsilon_4$, and M_I is the least upper bound of the set $\{|x| \mid x \in I_1 \cup A\}$ with $A = \{\frac{y}{\varepsilon_1 + \frac{1}{\alpha}} \mid y \in I_2\}$. This bound can be discovered using the arithmetic-geometric domain.

3 Underlying Domain

We use the Abstract Interpretation framework [3,4,5] 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. The primitives ASSIGN, GUARD, and TICK are sound counterparts to concrete assignments, guards, and clock ticks. To effectively compute an approximation of concrete fixpoints, we introduce an iteration basis $\perp$, a widening operator $\triangledown$, and a narrowing operator $\vartriangle$. Several abstract domains collaborate and use simple constraints to refine each other. We introduce two domains of simple constraints. The domain of interval $\mathcal{V} \cup \{\texttt{clock}\} \rightarrow \mathcal{I}$ and the domain of absolute value ordering $\wp(\mathcal{V}^2)$. The interval constraints encoded by a map $\rho^\sharp \in \mathcal{V} \cup \{\texttt{clock}\} \rightarrow \mathcal{I}$ are satisfied by the environment set $\gamma_{\mathcal{I}}(\rho^\sharp) = \{\rho \in Env \mid \rho(X) \in \rho^\sharp(X),\ \forall X \in \mathcal{V} \cup \{\texttt{clock}\}\}$. The constraints encoded by a subset $\mathcal{R} \subseteq \mathcal{V}^2$ are satisfied by the environment set $\gamma_{\text{ABS}}(\mathcal{R}) = \bigcap_{(X,Y)\in\mathcal{R}} \{\rho \in Env \mid |\rho(X)| \leq |\rho(Y)|\}$. The primitives RANGE and ABS capture simple constraints about the values that are associated with variables by weakening the abstract elements of $Env^\sharp$. These constraints are useful in refining the arithmetic-geometric progression domain. Conversely, a primitive REDUCE uses the range constraints that have been computed by the other domains in order to refine the underlying domain.

Definition 1 (Generic abstraction). *An abstraction is defined by a tuple* $(Env^\sharp, \gamma, \sqcup, \text{ASSIGN}, \text{GUARD}, \text{TICK}, \perp, \triangledown, \vartriangle, \text{RANGE}, \text{ABS}, \text{REDUCE})$ *such that:*

1. $Env^\sharp$ *is a set of properties;*
2. $\gamma \in Env^\sharp \rightarrow \wp(Env)$ *is a concretization map;*
3. $\forall a, b \in Env^\sharp, \gamma(a) \cup \gamma(b) \subseteq \gamma(a \sqcup b)$*;*
4. $\forall a \in Env^\sharp, X \in \mathcal{V}, E \in \mathcal{E}, \rho \in \gamma(a),\ [\![X = E]\!]_{\texttt{mc}}(\rho) \subseteq \gamma(\text{ASSIGN}(X = E, a))$*;*
5. $\forall a \in Env^\sharp, X \in \mathcal{V} \cup \{\texttt{clock}\}, I \in \mathcal{I}$,
   $\{\rho \in \gamma(a) \mid \rho(X) \in I\} \subseteq \gamma(\text{GUARD}(X, I, a))$*;*
6. $\forall a \in Env^\sharp, \{\rho[\texttt{clock} \mapsto \rho(\texttt{clock}) + 1] \mid \rho \in \gamma(a)\} \subseteq \gamma(\text{TICK}(a))$*;*
7. $\forall a \in Env^\sharp, \rho^\sharp \in (\mathcal{V} \cup \{\texttt{clock}\} \rightarrow \mathcal{I}),\ \gamma(a) \cap \gamma_{\mathcal{I}}(\rho^\sharp) \subseteq \gamma(\text{REDUCE}(\rho^\sharp, a))$*;*
8. $\triangledown$ *is a widening operator such that:* $\forall a, b \in Env^\sharp,\ \gamma(a) \cup \gamma(b) \subseteq \gamma(a \triangledown b)$*; and* $\forall k \in \mathbb{N},\ \rho_1, ..., \rho_k \in (\mathcal{V} \cup \{\texttt{clock}\} \rightarrow \mathcal{I}),\ (a_i) \in (Env^\sharp)^{\mathbb{N}}$*, the sequence* $\left(a_i^\triangledown\right)$ *defined by* $a_0^\triangledown = r(a_0)$ *and* $a_{n+1}^\triangledown = r(a_n^\triangledown \triangledown a_{n+1})$ *with* $r = [X \mapsto \text{REDUCE}(\rho_k, X)] \circ ... \circ [X \mapsto \text{REDUCE}(\rho_1, X)]$*, is ultimately stationary;*
9. $\vartriangle$ *is a narrowing operator such that:* $\forall a, b \in Env^\sharp,\ \gamma(a) \cap \gamma(b) \subseteq \gamma(a \vartriangle b)$*; and* $\forall k \in \mathbb{N},\ \rho_1, ..., \rho_k \in (\mathcal{V} \cup \{\texttt{clock}\} \rightarrow \mathcal{I}),\ (a_i) \in (Env^\sharp)^{\mathbb{N}}$*, the sequence* $\left(a_i^\vartriangle\right)$ *defined by* $a_0^\vartriangle = r(a_0)$ *and* $a_{n+1}^\vartriangle = r(a_n^\vartriangle \vartriangle a_{n+1})$*, with* $r = [X \mapsto \text{REDUCE}(\rho_k, X)] \circ ... \circ [X \mapsto \text{REDUCE}(\rho_1, X)]$*, is ultimately stationary;*
10. $\forall a \in Env^\sharp,\ \gamma(a) \subseteq \gamma_{\mathcal{I}}(\text{RANGE}(a))$ *and* $\gamma(a) \subseteq \gamma_{\text{ABS}}(\text{ABS}(a))$*.*

Least fixpoint approximation is performed in two steps [4]: we first compute an approximation using the widening operator; then we refine it using the narrowing operator. More formally, let f be a monotonic map in $\wp(Env) \rightarrow$

$\wp(Env)$ and $(f^\sharp \in Env^\sharp \to Env^\sharp)$ be an abstract counterpart of f satisfying $\forall a \in Env^\sharp,\ (f \circ \gamma)(a) \subseteq (\gamma \circ f^\sharp)(a)$. It is worth noting that the abstract counterpart $f^\sharp$ is usually not monotonic with respect to the partial order $\sqsubseteq^\sharp$ that is defined by $a \sqsubseteq^\sharp b \Longleftrightarrow \gamma(a) \subseteq \gamma(b)$. The abstract upward iteration $(C_n^\triangledown)$ of $f^\sharp$ is defined by $C_0^\triangledown = \bot$ and $C_{n+1}^\triangledown = C_n^\triangledown \triangledown f^\sharp(C_n^\triangledown)$. The sequence $(C_n^\triangledown)$ is ultimately stationary and we denote its limit by $C_\omega^\triangledown$. Then the abstract downward iteration $(D_n^\vartriangle)$ of $f^\sharp$ is defined by $D_0^\vartriangle = C_\omega^\triangledown$ and $D_{n+1}^\vartriangle = D_n^\vartriangle \vartriangle f^\sharp(D_n^\vartriangle)$. The sequence $(D_n^\vartriangle)$ is ultimately stationary and we denote its limit by $D_\omega^\vartriangle$. We define[2] $\mathrm{lfp}^\sharp(f^\sharp)$ by the limit $D_\omega^\vartriangle$ of the abstract downward iteration of $f^\sharp$. We introduce some lemmas in order to prove that $\mathrm{lfp}(f) \subseteq \gamma(D_\omega^\vartriangle)$:

Lemma 1. *We have* $f(\gamma(C_\omega^\triangledown)) \subseteq \gamma(C_\omega^\triangledown)$.

Proof. Since $C_\omega^\triangledown$ is the limit of the upward-iteration, we have $C_\omega^\triangledown = C_\omega^\triangledown \triangledown f^\sharp(C_\omega^\triangledown)$. By Def. 1.(8) of the widening, we obtain that $\gamma(f^\sharp(C_\omega^\triangledown)) \subseteq \gamma(C_\omega^\triangledown)$. By soundness of $f^\sharp$, we also have $f(\gamma(C_\omega^\triangledown)) \subseteq \gamma(f^\sharp(C_\omega^\triangledown))$. So $f(\gamma(C_\omega^\triangledown)) \subseteq \gamma(C_\omega^\triangledown)$. □

Lemma 2. *For all* $a \in \wp(Env)$ *and* $x \in Env^\sharp$, *we have:*

$$a \subseteq \gamma(x) \implies a \cap f(a) \subseteq \gamma(x \vartriangle f^\sharp(x)).$$

Proof. Let $a \in \wp(Env)$ and $x \in Env^\sharp$ such that $a \subseteq \gamma(x)$. Since f is monotonic, we have $f(a) \subseteq f(\gamma(x))$. Then by soundness of $f^\sharp$, we have $f(\gamma(x)) \subseteq \gamma(f^\sharp(x))$. Thus $f(a) \subseteq \gamma(f^\sharp(x))$. So $a \cap f(a) \subseteq \gamma(x) \cap \gamma(f^\sharp(x))$. By Def. 1.(9), we have $\gamma(x) \cap \gamma(f^\sharp(x)) \subseteq \gamma(x \vartriangle f^\sharp(x))$. We conclude that $a \cap f(a) \subseteq \gamma(x \vartriangle f^\sharp(x))$. □

Lemma 3. *For all* $a \in \wp(Env)$, *we have:*

$$f(a) \subseteq a \implies f(f(a) \cap a) \subseteq f(a) \cap a.$$

Proof. Let $a \in \wp(Env)$ such that $f(a) \subseteq a$. We have $f(a) \cap a = f(a)$. Since f is monotonic, we have $f(f(a)) \subseteq f(a)$. We conclude that $f(f(a) \cap a) \subseteq f(a) \cap a$. □

Lemma 4 (transfinite kleenean iteration). *For all* $a \in \wp(Env)$, *we have:*

$$f(a) \subseteq a \implies \mathit{lfp}(f) \subseteq a.$$

Theorem 1. *We have* $\mathit{lfp}(f) \subseteq \gamma(D_\omega^\vartriangle)$.

Proof. We introduce the sequence (u_n) that is defined by $u_0 = \gamma(C_\omega^\triangledown)$ and $u_{n+1} = u_n \cap f(u_n)$ for any $n \in \mathbb{N}$. We can prove by induction that $\forall n \in \mathbb{N}$, we have:

1. $u_n \subseteq \gamma(D_n^\vartriangle)$;
2. $f(u_n) \subseteq u_n$.

[2] $\mathrm{lfp}^\sharp(f^\sharp)$ is an approximation of the concrete least fixpoint; it may not be a least fixpoint of the abstract counterpart $f^\sharp$ which is not supposed to be monotonic.

$$
\begin{aligned}
&[\![V = E]\!]^\sharp_{\mathtt{mc}}(a) = \textsc{assign}(V = E, a)\\
&[\![\mathbf{skip}]\!]^\sharp_{\mathtt{mc}}(a) = a\\
&[\![\mathbf{tick}]\!]^\sharp_{\mathtt{mc}}(a) = \textsc{guard}(\mathtt{clock}, [0; \mathtt{mc}], \textsc{tick}(a))\\
&[\![\mathbf{if}\ (V \geq 0)\ \{P_1\}\ \mathbf{else}\ \{P_2\}]\!]^\sharp_{\mathtt{mc}}(a) = a_1 \sqcup a_2,\\
&\text{where } \begin{cases} a_1 = [\![P_1]\!]^\sharp_{\mathtt{mc}}(\textsc{guard}(V, [0; +\infty[, a))\\ a_2 = [\![P_2]\!]^\sharp_{\mathtt{mc}}(\textsc{guard}(V,]-\infty; 0[, a)) \end{cases}\\
&[\![\mathbf{while}\ (V \geq 0)\ \{P\}]\!]^\sharp_{\mathtt{mc}}(a) = \textsc{guard}(V,]-\infty; 0[, \mathit{Inv}^\sharp),\\
&\text{where } \mathit{Inv}^\sharp = \mathrm{lfp}^\sharp \left(X \mapsto a \sqcup [\![P]\!]^\sharp_{\mathtt{mc}}(\textsc{guard}(V, [0; +\infty[, X))\right)\\
&[\![P_1; P_2]\!]^\sharp_{\mathtt{mc}}(a) = [\![P_2]\!]^\sharp_{\mathtt{mc}}([\![P_1]\!]^\sharp_{\mathtt{mc}}(a))
\end{aligned}
$$

Figure 3. Abstract semantics.

- When $n = 0$: by definition, we have $u_0 = \gamma(C^\triangledown_\omega) = \gamma(D^\vartriangle_0)$ and thanks to Lemma 1, we have $f(u_0) \subseteq u_0$.
- We now suppose there exists $n \in \mathbb{N}$ such that $u_n \subseteq D^\vartriangle_n$ and $f(u_n) \subseteq u_n$.
  1. We have $u_{n+1} = u_n \cap f(u_n)$ and $u_n \subseteq \gamma(D^\vartriangle_n)$. By Lemma 2, we have $u_{n+1} \subseteq \gamma(D^\vartriangle_n \vartriangle f^\sharp(D^\vartriangle_n))$. By definition of $D^\vartriangle_{n+1}$, we obtain that $u_{n+1} \subseteq \gamma(D^\vartriangle_{n+1})$.
  2. We have $f(u_{n+1}) = f(u_n \cap f(u_n))$ and $f(u_n) \subseteq u_n$. By Lemma 3, we obtain that $f(u_{n+1}) \subseteq u_{n+1}$.

Then let $n \in \mathbb{N}$ be a natural such that $D^\vartriangle_\omega = D^\vartriangle_n$. We have $u_n \subseteq \gamma(D^\vartriangle_\omega)$ and $f(u_n) \subseteq u_n$. By lemma 4, we have $\mathrm{lfp}(f) \subseteq \gamma(D^\vartriangle_\omega)$. □

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

Theorem 2. *For any program P, environment ρ, abstract element a, and maximum clock value $\mathtt{mc}$, we have:*

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

4 Arithmetic-geometric progressions

4.1 Affine transformations

We introduce, as follows, the family of the affine transformations ($f[a, b]$) that is indexed by two non-negative real parameters a and b:

$$f[a, b] : \begin{cases} \mathbb{R}^+ \to \mathbb{R}^+\\ X \quad \mapsto a \times X + b \end{cases}$$

Lemma 5. *Let $a_1, a_2, b_1, b_2, X_1, X_2$ be non-negative real numbers in $\mathbb{R}^+$. If $a_1 \leq a_2,\ b_1 \leq b_2,$ and $X_1 \leq X_2$, then $f[a_1, b_1](X_1) \leq f[a_2, b_2](X_2)$.*

4.2 Arithmetic-geometric progression in the real field

We introduce the predomain $\mathcal{D}_{\mathbb{R}}$ of all the 5-tuples of non-negative real numbers. The predomain $\mathcal{D}_{\mathbb{R}}$ is ordered by the product order $\sqsubseteq_{\mathcal{D}_{\mathbb{R}}}$. Intuitively,[3] an element (M, a, b, a', b') of this predomain encodes an arithmetic-geometric progression. The real M is a bound on the initial value of the progression. The affine transformation $f[a', b']$ over-approximates the composition of all the affine transformations that can be applied to a value between two consecutive clock ticks. Finally, the affine transformation $f[a, b]$ over-approximates the composition of all the affine transformations that have been applied to a value since the last clock tick.

Thus, given a clock value $\mathtt{vc} \in \mathbb{N}$, we can define the concretization $\gamma^{\mathtt{vc}}_{\mathcal{D}_{\mathbb{R}}}(d)$ of such a tuple $d = (M, a, b, a', b') \in \mathcal{D}_{\mathbb{R}}$ by the set of all the elements $X \in \mathbb{R}$ such that $|X| \leq f[a,b]\left((f[a',b'])^{(\mathtt{vc})}(M)\right)$. We now define some primitives to handle the elements of $\mathcal{D}_{\mathbb{R}}$:

1. The join operator $\sqcup_{\mathcal{D}_{\mathbb{R}}}$ applies the maximum function component-wise. The soundness of the operator $\sqcup_{\mathcal{D}_{\mathbb{R}}}$ is established by Thm. 3, as follows:

   Theorem 3. *For any* $\mathtt{vc} \in \mathbb{N}$,
   $$\gamma^{\mathtt{vc}}_{\mathcal{D}_{\mathbb{R}}}(d_1) \cup \gamma^{\mathtt{vc}}_{\mathcal{D}_{\mathbb{R}}}(d_2) \subseteq \gamma^{\mathtt{vc}}_{\mathcal{D}_{\mathbb{R}}}(d_1 \sqcup_{\mathcal{D}_{\mathbb{R}}} d_2).$$

   Proof. By Lem. 5. □

2. The primitive $\mathit{affine}_{\mathcal{D}_{\mathbb{R}}}$ computes a linear combination among some elements of $\mathcal{D}_{\mathbb{R}}$. Let $n \in \mathbb{N}^*$ be a positive natural,[4] let $(d_i) = (M_i, a_i, b_i, a'_i, b'_i) \in \mathcal{D}^n_{\mathbb{R}}$ be a family of elements in $\mathcal{D}_{\mathbb{R}}$, let $(\alpha_i) \in (\mathbb{R} \setminus \{0\})^n$ be a family of real coefficients that are all distinct from 0, and let $\beta \in \mathbb{R}$ be a real coefficient. We define the element $\mathit{affine}_{\mathcal{D}_{\mathbb{R}}}((\alpha_i, d_i), \beta) \in \mathcal{D}_{\mathbb{R}}$ by $(g(M_i), a_\infty \times \alpha', |\beta| + g(b_i), a'_\infty, g(b'_i))$ where:
   - $a_\infty = \mathit{max}\{|a_i| \mid 1 \leq i \leq n\}$, $a'_\infty = \mathit{max}\{|a'_i| \mid 1 \leq i \leq n\}$,
   - $\alpha' = \sum\limits_{1 \leq i \leq n} |\alpha_i|$ and
   - the function $g : \mathbb{R}^n \to \mathbb{R}$ maps each family (x_i) of n real numbers into the real number defined by $\sum\limits_{1 \leq i \leq n} \frac{|\alpha_i \times x_i|}{\alpha'}$.

   The soundness of the primitive $\mathit{affine}_{\mathcal{D}_{\mathbb{R}}}$ is established by Thm. 4, as follows:

   Theorem 4. *Let* $\mathtt{vc} \in \mathbb{N}$ *be a natural and* $(X_i) \in \mathbb{R}^n$ *be a non-empty family of reals such that for any* i *such that* $1 \leq i \leq n$, *we have* $X_i \in \gamma^{\mathtt{vc}}_{\mathcal{D}_{\mathbb{R}}}(d_i)$. *Then we have:*
   $$\left(\sum_{1 \leq i \leq n} \alpha_i \times X_i + \beta\right) \in \gamma^{\mathtt{vc}}_{\mathcal{D}_{\mathbb{R}}}\left(\mathit{affine}_{\mathcal{D}_{\mathbb{R}}}((\alpha_i, d_i), \beta)\right).$$

[3] In Sect. 6 we forget this intuition to get a more expressive domain.

[4] The approximation of affine constants has an ad-hoc handling (Cf. Sect. 5.3).

Proof. By Lemma 5, by replacing α_i with $\frac{\alpha' \times \alpha_i}{\alpha'}$, by expanding $f[a,b]^{(\mathtt{vc})}$, and by applying the triangular inequality. $\square$

3. The primitive $\text{TICK}_{\mathcal{D}_{\mathbb{R}}} \in \mathcal{D}_{\mathbb{R}} \to \mathcal{D}_{\mathbb{R}}$ simulates clock ticks. It maps any element $d = (M, a, b, a', b') \in \mathcal{D}_{\mathbb{R}}$ into the element $(M, 1, 0, \mathit{max}(a, a'), \mathit{max}(b, b')) \in \mathcal{D}_{\mathbb{R}}$. Thus, just after the clock tick, the arithmetic-geometric progression that has been applied since the last clock tick is the identity. The progression between two clock ticks is chosen by applying the worst case among the progression between the last two clock ticks, and the progression between any other two consecutive clock ticks. The soundness of this operator is established by Thm. 5 as follows:

   Theorem 5 (clock tick). *Let* $\mathtt{vc} \in \mathbb{N}$ *be a natural. Then we have:*

   $$\gamma^{\mathtt{vc}}_{\mathcal{D}_{\mathbb{R}}}(d) \subseteq \gamma^{\mathtt{vc}+1}_{\mathcal{D}_{\mathbb{R}}}(\text{TICK}_{\mathcal{D}_{\mathbb{R}}}(d))$$

   Proof. By Lemma 5. $\square$

   For the sake of accuracy, we get a more precise definition of the primitive $\text{TICK}_{\mathcal{D}_{\mathbb{R}}}$ in Sect. 6, by forgetting the intuitive meaning of the elements of $\mathcal{D}_{\mathbb{R}}$.

4. The primitive $\mathit{range}_{\mathcal{D}_{\mathbb{R}}} \in (\mathcal{D}_{\mathbb{R}} \times \{[a;b] \mid a, b \in \mathbb{N},\ a \leq b\}) \to \mathcal{I}$ associates an element of $\mathcal{D}_{\mathbb{R}}$ and an interval for the clock counter with an interval range: we define $\mathit{range}_{\mathcal{D}_{\mathbb{R}}}((M, a, b, a', b'), [m_{\mathtt{vc}}; M_{\mathtt{vc}}])$ by $[-l; l]$ where $l = \mathit{max}(u_{m_{\mathtt{clock}}}, u_{M_{\mathtt{clock}}})$ and for any $\mathtt{vc} \in \mathbb{N}$,

   $$u_{\mathtt{vc}} = \begin{cases} a \times (M + \mathtt{vc} \times b') + b & \text{if } a' = 1, \\ a \times \left(a'^{\mathtt{vc}} \times \left(M - \frac{b'}{1-a'}\right) + \frac{b'}{1-a'}\right) + b & \text{otherwise.} \end{cases}$$

   The soundness of the primitive $\mathit{range}_{\mathcal{D}_{\mathbb{R}}}$ is established in Thm. 6 as follows:

   Theorem 6. *For any* $\mathtt{vc} \in \mathbb{N}$ *such that* $m_{\mathtt{vc}} \leq \mathtt{vc} \leq M_{\mathtt{vc}}$*, we have:*

   $$\gamma^{\mathtt{vc}}_{\mathcal{D}_{\mathbb{R}}}(d) \subseteq \mathit{range}_{\mathcal{D}_{\mathbb{R}}}(d, [m_{\mathtt{vc}}; M_{\mathtt{vc}}]).$$

   Proof. By studying the sign of $(u_{n+1} - u_n)$, for any $n \in \mathbb{N}$. $\square$

4.3 Representable real numbers

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 contains the set of numbers $\{0, 1\}$ and 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 $\mathbb{F}$. We denote the set $\{x \in \mathbb{F} \mid x \geq 0\}$ by $\mathbb{F}^+$ and the set $\mathbb{F}^+ \cup \{+\infty\}$ by $\overline{\mathbb{F}}^+$. The result of a computation on elements of $\overline{\mathbb{F}}$ may be not in $\overline{\mathbb{F}}$. So we suppose that we are given a function $\lceil \_ \rceil \in \mathbb{R} \to \overline{\mathbb{F}}$ such that $\lceil x \rceil \geq x$, for any $x \in \mathbb{R}$ and such that $\lceil x \rceil \leq 0$, for any $x \leq 0$. The domain $\overline{\mathbb{F}}$ is related to $\wp(\mathbb{R})$ via the concretization $\gamma_{\overline{\mathbb{F}}}$ that maps any representable number e into the set of the reals $r \in \mathbb{R}$ such that $|r| < e$, moreover we set $\gamma_{\overline{\mathbb{F}}}(+\infty) = \mathbb{R}$ and $\gamma_{\overline{\mathbb{F}}}(-\infty) = \emptyset$.

4.4 Representable arithmetic-geometric constraints

We introduce the predomain $\mathcal{D}_{\mathbb{F}}$ as the set of the 5-tuples $(M,a,b,a',b') \in \left(\overline{\mathbb{F}}^{+}\right)^{5}$. The order $\sqsubseteq_{\mathcal{D}_{\mathbb{F}}}$, the concretizations $\gamma^{\mathtt{vc}}_{\mathcal{D}_{\mathbb{F}}}$ (for any $\mathtt{vc} \in \mathbb{N}$), the join operator $\sqcup_{\mathcal{D}_{\mathbb{F}}}$, and the clock tick primitive $\text{TICK}_{\mathcal{D}_{\mathbb{F}}}$ are respectively defined as restrictions of the order $\sqsubseteq_{\mathcal{D}_{\mathbb{R}}}$, the concretizations $\gamma^{\mathtt{vc}}_{\mathcal{D}_{\mathbb{R}}}$, the join operator $\sqcup_{\mathcal{D}_{\mathbb{R}}}$, and the primitive $\text{TICK}_{\mathcal{D}_{\mathbb{R}}}$.

We now update the definition of the linear combination primitive $\mathit{affine}_{\mathcal{D}_{\mathbb{F}}}$ and of the reduction primitive $\mathit{range}_{\mathcal{D}_{\mathbb{F}}}$:

1. The primitive $\mathit{affine}_{\mathcal{D}_{\mathbb{F}}}$ maps each pair $((\alpha_i,d_i),\beta)$ where $(\alpha_i) \in \mathbb{R}^n$ (where $n \in \mathbb{N}^*$ is a positive natural), $(d_i) = ((M_i,a_i,b_i,a'_i,b'_i)) \in \mathcal{D}_{\mathbb{F}}^n$, and $\beta \in \mathbb{R}$ to the element:
$$(g(M_i), \lceil a_\infty \times \alpha'_M \rceil, |\beta| + g(b_i), a'_\infty, g(b'_i))$$
where:
   - $a_\infty = \mathit{max}\{|a_i| \mid 1 \leq i \leq n\}$, $a'_\infty = \mathit{max}\{|a'_i| \mid 1 \leq i \leq n\}$,
   - $s_n : \mathbb{R}^n \to \overline{\mathbb{F}}^{+}$ is defined by $s_0(()) = 0$ and $s_{n+1}((a_i)_{1\leq i\leq n+1}) = \lceil s_n((a_i)_{1\leq i\leq n}) + \lceil a_{n+1} \rceil \rceil$.
   - $\alpha'_m = -s_n((-|\alpha_i|)_{1\leq i\leq n})$, $\alpha'_M = s_n((|\alpha_i|)_{1\leq i\leq n})$,
   - $g : \mathbb{R}^n \to \overline{\mathbb{F}}^{+}$ maps each family (x_i) of real numbers into the real number that is defined by
$$\mathit{min}\left(\mathit{max}\left(\{|x_i| \mid 1 \leq i \leq n\}\right), s_n\left(\frac{\lceil |\alpha_i| \times |x_i| \rceil}{\alpha'_m}\right)\right).$$

   Theorem 7. *For any clock value* $\mathtt{vc} \in \mathbb{N}$*, we have:*
$$\left\{ \sum_{1\leq i\leq n} \alpha_i \times X_i + \beta \;\middle|\; X_i \in \gamma^{\mathtt{vc}}_{\mathcal{D}_{\mathbb{F}}}(d_i) \right\} \subseteq \gamma^{\mathtt{vc}}_{\mathcal{D}_{\mathbb{F}}}\left(\mathit{affine}_{\mathcal{D}_{\mathbb{F}}}((\alpha_i,d_i),\beta)\right).$$

   Proof. By Lemma 5 and Thm. 4. □

   Remark 1. We define g as the minimum of two sound results. In the real field, the second one is more precise. However, it may become less precise when computing with rounding errors.

2. The interval $\mathit{range}_{\mathcal{D}_{\mathbb{F}}}((M,a,b,a',b'),[m_{\mathtt{vc}};M_{\mathtt{vc}}])$ is given by $[-l;l]$ where:
   - $l = \mathit{max}(u_{m_{\mathtt{vc}}}, u_{M_{\mathtt{vc}}})$;
   - $u_{\mathtt{vc}} = \lceil \lceil a \times v_{\mathtt{vc}} \rceil + b \rceil$;
   - $v_{\mathtt{vc}} = \begin{cases} \lceil M + \lceil \mathtt{vc} \times b' \rceil \rceil & \text{if } a' = 1, \\ \lceil c_1^+ + c_2^+ \rceil & \text{otherwise;} \end{cases}$
   - $\begin{cases} \exp_0^- = 1,\ \exp_{2\times n}^- = -\lceil \exp_n^- \times (-\exp_n^-) \rceil, \\ \exp_{2\times n+1}^- = -\lceil \lceil \exp_n^- \times (-\exp_n^-) \rceil \times a' \rceil; \end{cases}$
   - $\begin{cases} \exp_0^+ = 1,\ \exp_{2\times n}^+ = \lceil \exp_n^+ \times \exp_n^+ \rceil, \\ \exp_{2\times n+1}^+ = \lceil \lceil \exp_n^+ \times \exp_n^+ \rceil \times a' \rceil; \end{cases}$

– $c_1^+ = \begin{cases} \lceil \exp_{\mathtt{vc}}^+ \times \lceil M - c_2^- \rceil \rceil & \text{if } M \geq c_2^- \\ \lceil \exp_{\mathtt{vc}}^- \times \lceil M - c_2^- \rceil \rceil & \text{otherwise;} \end{cases}$
– $c_2^- = -\left\lceil \frac{-b'}{\lceil 1-a' \rceil} \right\rceil$ and $c_2^+ = \left\lceil \frac{-b'}{\lceil a'-1 \rceil} \right\rceil$.

Theorem 8. *For any clock value* $\mathtt{vc} \in [m_{\mathtt{vc}}; M_{\mathtt{vc}}]$*, we have:*

$$\gamma_{\mathcal{D}_{\mathbb{F}}}^{\mathtt{vc}}(d) \subseteq range_{\mathcal{D}_{\mathbb{F}}}(d, \mathtt{vc}).$$

Proof. Because $\forall \mathtt{vc} \in \mathbb{N}$, $\exp_{\mathtt{vc}}^- \leq a'^{\mathtt{vc}} \leq \exp_{\mathtt{vc}}^+$, $c_1^+ \geq a'^{\mathtt{vc}} \times \left(M - \frac{b'}{1-a'}\right)$ and $c_2^- \leq \frac{b'}{1-a'} \leq c_2^+$, and by applying Thm. 6. □

Remark 2. In the implementation, we use memoization to avoid computing the same exponential twice.

4.5 Tuning the extrapolation strategy

Although $\overline{\mathbb{F}}$ is height-bounded, we introduce some extrapolation operators in order to accelerate the convergence of abstract iterations. A widening step consists in applying an extensive map to each unstable components of the 5-tuples. In order to let constraints stabilize, we only widen a component when it has been unstable a given number of times since its last widening. For that purpose, we associate each representable number in $\overline{\mathbb{F}}$ with a natural that denotes the number of times it has been unstable without being widened. We suppose that we are given a natural parameter n and an extensive function f over $\overline{\mathbb{F}}$. We first define the widening ∇_f^n of two annotated representable numbers $(x_1, n_1), (x_2, n_2) \in \overline{\mathbb{F}} \times \mathbb{N}$ by:

$$(x_1, n_1) \nabla_f^n (x_2, n_2) = \begin{cases} (x_1, n_1) & \text{if } x_1 \geq x_2 \\ (x_2, n_1 + 1) & \text{if } x_1 < x_2 \text{ and } n_1 < n \\ (f(x_2), 0) & \text{otherwise.} \end{cases}$$

A narrowing step refines an arithmetic-geometric constraint with another one if the last one is smaller component-wise (so that we are sure that this refinement does not locally lose accuracy). To avoid too long decreasing sequences, we count the number of times such a refinement has been applied with each constraint. Thus we associate each constraint with an extra counter.

We then introduce the predomain $\mathcal{D}_{\mathbb{F}}^{\mathcal{L}} = \left(\overline{\mathbb{F}}^+ \times \mathbb{N}\right)^5 \times \mathbb{N}$ of annotated constraints. The function *annotate* maps each element $d = (M, a, b, a', b') \in \mathcal{D}_{\mathbb{F}}$ to the annotated element that is defined by $(((M, 0), (a, 0), (b, 0), (a', 0), (b', 0)), 0) \in \mathcal{D}_{\mathbb{F}}^{\mathcal{L}}$, where all counters are initialized with the value 0. Conversely the function *remove* maps each element $(((M, n_M), (a, n_a), (b, n_b), (a', n_{a'}), (b', n_{b'})), n) \in \mathcal{D}_{\mathbb{F}}^{\mathcal{L}}$ to the annotation-free element $(M, a, b, a', b') \in \mathcal{D}_{\mathbb{F}}$. We can define the preorder $\sqsubseteq_{\mathcal{D}_{\mathbb{F}}^{\mathcal{L}}}$ by $a \sqsubseteq_{\mathcal{D}_{\mathbb{F}}^{\mathcal{L}}} b \iff remove(a) \sqsubseteq_{\mathcal{D}_{\mathbb{F}}} remove(b)$. The monotonic concretization $\gamma_{\mathcal{D}_{\mathbb{F}}^{\mathcal{L}}}$ is defined as the composition $\gamma_{\mathcal{D}_{\mathbb{F}}} \circ remove$.

Extrapolation operators store information about the history of the extrapolation process into the counters of their left argument, whereas the other primitives reset these counters: we define the union $a \sqcup_{\mathcal{D}_{\mathbb{F}}^{\mathcal{L}}} b$ by $\mathit{annotate}((\mathit{remove}(a)) \sqcup_{\mathcal{D}_{\mathbb{F}}} (\mathit{remove}(b)))$, the affine combination $\mathit{affine}_{\mathcal{D}_{\mathbb{F}}^{\mathcal{L}}}((\alpha_i, d_i), \beta)$ by the abstract element $\mathit{annotate}(\mathit{affine}_{\mathcal{D}_{\mathbb{F}}}((\alpha_i, \mathit{remove}(d_i)), \beta))$, the abstract clock tick primitive $\mathrm{TICK}_{\mathcal{D}_{\mathbb{F}}^{\mathcal{L}}}$ by the map $\mathit{annotate} \circ \mathrm{TICK}_{\mathcal{D}_{\mathbb{F}}} \circ \mathit{remove}$, and the interval constraints $\mathit{range}_{\mathcal{D}_{\mathbb{F}}^{\mathcal{L}}}(d, I)$ by the interval map $\mathit{range}_{\mathcal{D}_{\mathbb{F}}}(\mathit{remove}(d), I)$.

We define extrapolation operators. Let f_a, f_b, and f_M be extensive functions over the set $\overline{\mathbb{F}}$; let n_a, n_b, n_M, and n be some naturals. The functions f_a, f_b, and f_M and the naturals n_a, n_b, n_M, and n are left as parameters of our extrapolation strategy. The widening $((M_1, a_1, b_1, a'_1, b'_1), n_1) \nabla_{\mathcal{D}_{\mathbb{F}}^{\mathcal{L}}} ((M_2, a_2, b_2, a'_2, b'_2), n_2)$ is defined by $((M_1 \nabla_{f_M}^{n_M} M_2, a_1 \nabla_{f_a}^{n_a} a_2, b_1 \nabla_{f_b}^{n_b} b_2, a'_1 \nabla_{f_a}^{n_a} a'_2, b'_1 \nabla_{f_b}^{n_b} b'_2), 0)$. The narrowing $(t_1, n_1) \Delta_{\mathcal{D}_{\mathbb{F}}^{\mathcal{L}}} (t_2, n_2)$ is then defined by $(t_2, n_1 + 1)$ in the case when $n_1 < n$ and $(t_2, n_2) \sqsubseteq_{\mathcal{D}_{\mathbb{F}}^{\mathcal{L}}} (t_1, n_1)$, and by (t_1, n_1) otherwise.

5 Refining an existing abstraction

We now show how we can extend an existing abstraction defined as in Def. 1 so that it can also deal with arithmetic-geometric constraints.

5.1 Domain extension

Let $(\mathit{Env}_0^\sharp, \gamma_0, \sqcup_0, \mathrm{ASSIGN}_0, \mathrm{GUARD}_0, \mathrm{TICK}_0, \bot_0, \nabla_0, \Delta_0, \mathrm{RANGE}_0, \mathrm{ABS}_0, \mathrm{REDUCE}_0)$ be an abstraction which is called the underlying domain. We build the abstraction $\mathit{Env}^\sharp$ as the Cartesian product $\mathit{Env}_0^\sharp \times (\mathcal{V} \to \mathcal{D}_{\mathbb{F}}^{\mathcal{L}} \cup \{\top\})$. The element $\top \notin \mathcal{D}_{\mathbb{F}}^{\mathcal{L}}$ denotes the absence of constraint. The concretization $\gamma : \mathit{Env}^\sharp \to \wp(\mathit{Env})$ maps each pair (e, f) to the following set of environments:

$$\gamma_0(e) \cap \left\{ \rho \in \mathit{Env} \;\middle|\; \forall X \in \mathcal{V} \text{ such that } f(X) \neq \top,\ \rho(X) \in \gamma_{\mathcal{D}_{\mathbb{F}}^{\mathcal{L}}}^{\rho(\mathtt{clock})}(f(X)) \right\}.$$

Moreover, abstract iterations start with the element $\bot = (\bot_0, [X \mapsto \top])$.

5.2 Refinement operators

The underlying domain and the arithmetic-geometric domain refine each other when the computation of an abstract primitive requires it. We introduce here some operators that describe these refinement steps.

The operator $r^{\leftarrow}$ uses the arithmetic-geometric constraints to refine the underlying domain. Given an abstract element $(e, f) \in \mathit{Env}^\sharp$ and a subset $V \subseteq \mathcal{V}$ of variables, we define $r^{\leftarrow}((e, f), V)$ by $(\mathrm{REDUCE}_0(g, e), f)$ where $g(X)$ is given by:

$$\begin{cases} \mathit{range}_{\mathcal{D}_{\mathbb{F}}^{\mathcal{L}}}(f(X), \mathrm{RANGE}_0(e)(\mathtt{clock})) & \text{if } X \in V \text{ and } f(X) \neq \top, \\ \mathbb{R} & \text{otherwise.} \end{cases}$$

Conversely, we use the underlying domain to build new arithmetic-geometric constraints or to refine existing arithmetic-geometric constraints. Let $X \in \mathcal{V}$

be a variable, let $a,\ b \in \mathbb{F}^+$ be two non negative real parameters, and let $e \in Env_0^\sharp$ be an abstract element of the underlying domain. The variable X can soundly be associated with the arithmetic-geometric constraint $g_e(X,(a,b))$, where $g_e(X,(a,b))$ is given by:

$$\begin{cases} annotate\left(\left(\left\lceil \frac{max(0,\lceil l-b \rceil)}{a} \right\rceil, a, b, 1, 0\right)\right) & \text{if } a \neq 0, \\ annotate\left((a, b, 1, 0, l)\right) & \text{otherwise,} \end{cases}$$

where $l \in \overline{\mathbb{F}}$ is the least upper bound (in $\overline{\mathbb{F}}$) of the set $\{|x| \mid x \in \text{RANGE}_0(e)(X)\}$.

We now define the operator $r^{\rightarrow}$ which refines arithmetic-geometric constraints over a set of variables by weakening the range constraints that can be extracted from the underlying domain. Given an abstract element $(e,f) \in Env^\sharp$, a subset $A \subseteq \mathcal{V}$, and a map $\Gamma \in A \rightarrow \mathbb{F}^+ \times \mathbb{F}^+$, we define $r^{\rightarrow}((e,f),A,\Gamma)$ by (e,f') where $f'(X)$ is defined by:

$$\begin{cases} g_e(X,\Gamma(X)) & \text{if } X \in A \text{ and either } f(X) = \top, \text{ or } g_e(X,\Gamma(X)) \sqsubseteq_{\mathcal{D}_{\mathbb{F}}^{\mathcal{L}}} f(X), \\ f(X) & \text{otherwise.} \end{cases}$$

5.3 Primitives

Binary operators are all defined in the same way. Let $\circledast$ be an operator in $\{\sqcup, \vartriangle, \triangledown\}$ and let $(e_1,f_1),(e_2,f_2)$ be two abstract elements of $Env^\sharp$. Before applying a binary operator, we refine arithmetic-geometric constraints so that both arguments constrain the same set of variables: we set for any $i \in \{1,2\}$, $(e_i', f_i') = (r^{\rightarrow}((e_i,f_i), V_{3-i} \backslash V_i, \Gamma_{3-i}))$, where for any $i \in \{1;2\}$, $V_i = \{X \mid f_i(X) \neq \top\}$ and $\Gamma_i(X) = (a,b)$ when $f_i(X)$ matches (M,a,b,a',b'). We then apply $\circledast$ component-wise: we set $e'' = e_1' \circledast_0 e_2'$; we set $f''(X) = f_1'(X) \circledast_{\mathcal{D}_{\mathbb{F}}^{\mathcal{L}}} f_2'(X)$ for any $X \in V_1 \cup V_2$; we set $f''(X) = \top$ for any $X \in \mathcal{V} \setminus (V_1 \cup V_2)$. After applying a binary operator, we use the arithmetic-geometric constraints to refine the underlying domain: we define $(e_1,f_1) \circledast (e_2,f_2)$ by $r^{\leftarrow}((e'',f''),\mathcal{V})$.

We use a heuristics to drive the abstraction of assignments. This heuristics weakens the precondition to simplify the assigned expression: it takes some variable ranges, some arithmetic-geometric constraints and an expression; it replaces in the expression some variables with their range. The choice of the variables that are replaced is left as a parameter of the abstraction. Thus, the heuristics $heu \in ((\mathcal{V} \cup \{\mathtt{clock}\} \rightarrow \mathcal{I}) \times (\mathcal{V} \rightarrow \mathcal{D}_{\mathbb{F}}^{\mathcal{L}} \cup \{\top\})) \times \mathcal{E} \rightarrow ((\mathbb{R} \setminus \{0\} \times \mathcal{V})^* \times \mathbb{R})$ maps each pair $((\rho^\sharp, f), E)$ to a pair $((\alpha_i, V_i)_{1 \leq i \leq n}, \beta)$ that satisfies:

$$\gamma((\!|E|\!)) \subseteq \left\{ \beta + \sum_{1 \leq i \leq n} \alpha_i \times \rho(V_i) \;\middle|\; \begin{array}{l} \forall \rho \in Env \text{ such that:} \\ \forall X \in \mathcal{V}, \rho(X) \in \rho^\sharp(X) \cap \gamma_{\mathcal{D}_{\mathbb{F}}^{\mathcal{L}}}^{\rho(\mathtt{clock})}(f(X)) \\ \text{and } \rho(\mathtt{clock}) \in \rho^\sharp(\mathtt{clock}) \end{array} \right\}.$$

Let $(e,f) \in Env^\sharp$ be an abstract element. We consider several cases when abstracting an assignment $X = E$:

1. When computing an assignment $X = Y$ with $Y \in \mathcal{V}$, we associate the variable X with any constraint about Y. Thus we set:

$$\textsc{assign}(X = Y, (e, f)) = (\textsc{assign}_0(X = Y, e), f[X \mapsto f(Y)]);$$

2. When computing an assignment $X = E$ where $E \notin \mathcal{V}$ such that the pair $heu((\textsc{range}_0(e), f), E)$ matches $((), \beta)$, we remove any arithmetic-geometric constraints about X. Thus we set:

$$\textsc{assign}(X = E, (e, f)) = (\textsc{assign}_0(X = E, e), f[X \mapsto \top]);$$

3. Otherwise, we denote by $((\alpha_i, V_i)_{1 \leq i \leq n}, \beta) = heu((\textsc{range}_0(e), f), E)$ the approximation of E by the heuristics. Before the assignment, we use the underlying domain to refine information about the variables (V_i) that occur in the simplified expression. When such a variable is tied with no arithmetic-geometric constraint, we build one with arbitrary coefficients for the affine transformations. We also refine existing constraints without modifying the coefficients of the affine transformations. Thus we define the element (e', f') by $r^{\rightarrow}((e, f), \{V_i \mid 1 \leq i \leq n\}, \Gamma)$ where for any $i \in \mathbb{N}$ such that $1 \leq i \leq n$, we have $\Gamma(V_i) = (1, 0)$ if $f(V_i) = \top$ (*missing constraints*) and $\Gamma(V_i) = (a, b)$ if $f(V_i) = (M, a, b, a', b')$ (*existing constraints*). Then we apply the assignment component-wise: we define (e'', f'') by $(\textsc{assign}_0(X = E, e'), f[X \mapsto affine_{\mathcal{D}_{\mathbb{F}}^{\mathcal{L}}}((\alpha_i, f'(V_i)), \beta)])$. At last, we refine the underlying domain by the new computed constraint: we set $\textsc{assign}(X = E, (e, f)) = r^{\leftarrow}((e'', f''), \{X\})$.

The abstraction of a clock tick $\textsc{tick}(e, f)$ is defined component-wise by $(\textsc{tick}_0(e), f')$ where $f'(X) = \textsc{tick}_{\mathcal{D}_{\mathbb{F}}^{\mathcal{L}}}(f(X))$ if $f(X) \neq \top$ and $f'(X) = \top$ otherwise. We do not deal directly with guards in the arithmetic-geometric domain. Nevertheless, if after applying a guard, we can prove in the underlying domain that the absolute value that is associated with a variable is less than the absolute value that is associated with another variable, we use this information to refine arithmetic-geometric constraints. So we define the primitive $r^{\textsc{abs}}$ that refines arithmetic-geometric constraints, according to absolute value constraints. Given a relation $\mathcal{R} \subseteq \mathcal{V}^2$ and a map $f \in \mathcal{V} \rightarrow \mathcal{D}_{\mathbb{F}}^{\mathcal{L}} \cup \{\top\}$, the map $r^{\textsc{abs}}(\mathcal{R}, f) \in \mathcal{V} \rightarrow \mathcal{D}_{\mathbb{F}}^{\mathcal{L}} \cup \{\top\}$ associates any variable X with a minimal element (for $\sqsubseteq_{\mathcal{D}_{\mathbb{F}}^{\mathcal{L}}}$) of the set $(\{f(X)\} \cup \{f(Y) \mid (X, Y) \in \mathcal{R}\}) \setminus \{\top\}$ if this set is not empty, or with the element $\top$ otherwise. Then the abstract element $\textsc{guard}(X, I, (a, f))$ is defined by $(\textsc{guard}_0(X, I, a), r^{\textsc{abs}}(\textsc{abs}_0(\textsc{guard}_0(X, I, a)), f))$.

In order not to break the extrapolation process, we never refine arithmetic-geometric constraints after applying an extrapolation operator. Thus we define $\textsc{reduce}(\rho^{\sharp}, (e, f))$ by $(\textsc{reduce}_0(\rho^{\sharp}, e), f)$. Moreover, the domain $\mathcal{D}_{\mathbb{F}}^{\mathcal{L}}$ cannot help in comparing the absolute value of variables, so we set $\textsc{abs}(e, f) = \textsc{abs}_0(e)$. Nevertheless, the domain $\mathcal{D}_{\mathbb{F}}^{\mathcal{L}}$ can refine variable range: we set $\textsc{range}(e, f)(X)$ by $\textsc{range}_0(e)(X) \cap range_{\mathcal{D}_{\mathbb{F}}^{\mathcal{L}}}(f(X), \textsc{range}_0(e)(\texttt{clock}))$.

Theorem 9. $(Env^{\sharp}, \gamma, \sqcup, \textsc{assign}, \textsc{guard}, \textsc{tick}, \bot, \nabla, \triangle, \textsc{range}, \textsc{abs}, \textsc{reduce})$ *is an abstraction.*

Proof. We sketch the proof of Th. 9: all soundness requirements come from the soundness of both the underlying domain and the arithmetic-geometric predomain. During an extrapolation iteration (ascending or descending), the set of arithmetic-geometric constraints (i.e., the set of the variables that are not mapped into $\top$) is ultimately stationary (since the number of constraints is increasing, whereas $\mathcal{V}$ is finite); then each arithmetic-geometric constraint sequence is ultimately stationary; once the information that refines the underlying domain is fixed, the underlying domain termination criteria in Def. 1.(8-9) apply. $\square$

6 Dealing with buffers

The definition of the primitive $\textsc{tick}_{\mathcal{D}_{\mathbb{F}}}$ in Sect. 4.4 implicitly supposes that the affine transformations that must be captured are fully computed between two clock ticks. For instance, Ex. 2 can be analyzed accurately because the multiplication and the division are computed in the same loop iteration. We first slightly modify Ex. 2 so that this atomicity assumption is not satisfied. Then we refine the primitive $\textsc{tick}_{\mathcal{D}_{\mathbb{F}}}$ to handle more complex cases precisely.

6.1 Motivating example

Example 3. This example iterates a loop where a floating point is first divided by a coefficient $\alpha > 2$ and then multiplied by the coefficient α. Unlike Ex. 2, the division and the multiplication are not computed in the same iteration of the loop. At each iteration, the current value of X is multiplied by α and the result is stored in a buffer (denoted by the variable $\textsc{buffer}$). The next value for X is obtained by dividing the value that was in the buffer at the beginning of the iteration (while the current value of X is stored in a temporary variable $\textsc{tmp}$). For the sake of simplicity, we have removed reinitialization branches.

$V = \mathbb{R}$; $X = I$; $\textsc{tmp} = 0$; $\textsc{buffer} = I$;
while $(V \geq 0)$ {
  $V = \mathbb{R}$;
  $\textsc{tmp} = X$;
  $X = [\frac{1}{\alpha} - \varepsilon_1; \frac{1}{\alpha} + \varepsilon_1] \times \textsc{buffer} + [-\varepsilon_2; \varepsilon_2]$;
  $\textsc{buffer} = [\alpha - \varepsilon_3; \alpha + \varepsilon_3] \times \textsc{tmp} + [-\varepsilon_4; \varepsilon_4]$;
  tick}

where $0 < \varepsilon_i < 1$, for any $i \in \{1; 2; 3; 4\}$, $\alpha > 2$, and $I \in \mathcal{I}$.

Moreover, initialization values range in the intervals I. The parameter α is a coefficient in the example. The parameters ε_1 and ε_3 encode relative rounding errors, and the parameters ε_2 and ε_4 encode absolute rounding errors. The variable V allows stopping the loop iteration.

At the first abstract iteration, before the first clock tick, the variable $\textsc{buffer}$ is associated with the 5-tuple $(M_1, a_1, b_1, 1, 0)$ where M_I is the least upper bound of the set $\{|x| \mid x \in I\}$, $a_1 = \lceil \alpha + \varepsilon_3 \rceil$ and $b_1 = \lceil \varepsilon_4 \rceil$. After the first clock tick, it is associated with $(M_1, 1, 0, a_1, b_1)$. At the second abstract iteration, before the clock tick, the variable X is associated with the 5-tuple $(M_1, a_2, b_2, a_1, b_1)$ where $a_2 = \lceil \frac{1}{\alpha} + \varepsilon_1 \rceil$ and $b_2 = \lceil \varepsilon_2 \rceil$. After the second clock tick, the variable X is associated with the 5-tuple $(M_1, 1, 0, a_1, max(b_1, b_2))$. We notice that the arithmetic-geometric domain cannot help in bounding the range of the variable X because of the computation of the exponential (since we have $a_1 > 2$). All information has been lost when computing the first clock tick in the abstract.

6.2 Refining the domain

To refine the domain, we have to decide at each clock tick which affine computations are finished. For that purpose, we introduce two parameters $\beta_m, \beta_M \in \mathbb{F}^+$ very close to 1 and such that $\beta_m < 1 < \beta_M$. We then consider that an affine transformation $[X \mapsto a \times X + b]$ denotes a finished computation if and only if $\beta_m < a < \beta_M$. In fact, in the case when $a > \beta_M$ the arithmetic-geometric progression domain will provide useless range and in the case when $a < \beta_m$ the interval domain can provide accurate range by using widening and narrowing. Thus, we redefine the element $\text{TICK}_{\mathcal{D}_\mathbb{F}}(M, a, b, a', b')$ by $(M, a, 0, a', max\{b, b'\})$ in the case when both $a \notin [\beta_m; \beta_M]$ and $a' \geq 1$, and by $(M, 1, 0, max\{a, a'\}, max\{b, b'\})$ otherwise. This definition still satisfies Thm. 5.

In Ex. 3, after the first clock tick and provided that $a_1 < \beta_m$, the variable BUFFER is now associated with $(M_1, a_1, 0, 1, b_1)$. At the second abstract iteration, before the clock tick the variable X is associated with the 5-tuple $(M_1, a_3, b_3, 1, b_1)$ where $a_3 = \lceil \lceil \alpha + \varepsilon_3 \rceil \times a_1 \rceil$ and $b_3 = \varepsilon_3$. Then after the second clock tick and provided that $a_3 < \beta_M$, the variable X is associated with $(M_1, 1, 0, a_3, max(b_1, b_3))$. This constraint is stable and allows the computation of an accurate range for the variable X.

7 Benchmarks

We tested our framework with three programs of a same family of critical embedded software written in C. For each program we tested the ASTRÉE [2] analyzer with the classical domains (intervals [4], octagons [8], decision trees, and expanded digital filter domains [6]) and without/with the new arithmetic-geometric domain. For each of these analyses, we report in Fig. 4 the analyzed program size, the number of global variables, the number of arithmetic-geometric constraints that are captured by the analysis, 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 AMD Opteron 248, with 8 Gb of RAM. In two of the three programs, the arithmetic-geometric progression domain solve all the remaining false alarms which gives a proof of absence of run-time errors.

8 Conclusion

We have proposed a new numerical domain which relates the value of each variable to a clock counter. It approximates each value by an expression of the form

lines of C	**70,000**		**216,000**		**379,000**	
global variables	13,400		7,500		9,000	
ari-geo progressions	disabled	enabled	disabled	enabled	disabled	enabled
ari-geo constraints		257		458		634
iterations	53	47	228	64	238	67
average time per iteration	1mn30s	1mn47s	5mn40s	6mn07s	10mn17s	11mn35s
analysis time	1h20mn	1h24mn	21h32mn	6h33mn	40h58mn	12h55mn
warnings	24	**0**	80	1	189	**0**

Figure 4. Some statistics.

$[X \mapsto \alpha \times X + \beta]^{(n)}(M)$, where (M, α, β) are discovered automatically and n is the maximum value of the clock counter. This approximation is proved correct and allows us to bound the value of some floating-point variables by using the program execution time. These bounds cannot be discovered either by ignoring the clock counter or by just bounding the difference between variable values and the clock value (c.f. [1]). Our framework allows the full certification of huge critical embedded softwares. The accuracy gain significantly reduces the exploration space which leads to an analysis speed-up in some of our examples.

Acknowledgments. We deeply thank the anonymous referees. We also thank Francesco Logozzo, Enea Zaffanella, 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. In *The Essence of Computation: Complexity, Analysis, Transformation.*, 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. *Méthodes itératives de construction et d'approximation de points fixes d'opérateurs monotones sur un treillis, analyse sémantique des programmes.* PhD thesis, Université Scientifique et Médicale de Grenoble, 1978.
4. 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.
5. P. Cousot and R. Cousot. Abstract interpretation frameworks. *Journal of logic and computation*, 2(4), August 1992.
6. J. Feret. Static analysis of digital filters. In *European Symposium on Programming (ESOP'04)*, number 2986 in LNCS. Springer-Verlag, 2004.
7. X. Leroy, D. Doligez, J. Garrigue, D. Rémy, and J. Vouillon. The Objective Caml system, documentation and user's manual. Technical report, INRIA, 2002.
8. A. Miné. The octagon abstract domain. In *Proc. WCRE'01(AST'01)*, IEEE, 2001.
9. A. Miné. Relational abstract domains for the detection of floating-point run-time errors. In *Proc. ESOP'04*, LNCS. Springer, 2004.