Towards an Abstraction of the Physical Environment of Embedded Systems

Matthieu Martel

CEA - Recherche Technologique
LIST-DTSI-SOL
CEA F91191 Gif-Sur-Yvette Cedex, France
e-mail : matthieu.martel@cea.fr

1 Introduction

Static analyzers for critical embedded softwares often poorly abstract the physical environment in which, in practice, the embedded systems are run. To take an extreme example (more reasonable examples abound in articles dedicated to hybrid systems [1, 12]), the static analysis of avionic codes should abstract the plane environment, that is, the engines, the wings, and the atmosphere itself. This is because embedded software strongly interact with their environment, picking physical values by means of sensors and modifying them via actuators. In practice, the sensors correspond to volatile variables in C programs and, at analysis time, the user assigns to these variables a range given by the minimal and maximal values the sensor can send. In this case, a static analyzer must assume that the value sent by the sensor may switch from its minimum to its maximum in arbitrary short laps of time, while, in practice, it follows a slow evolution. As a consequence, the results of the static analysis are significantly over-approximated. The abstraction of the physical environment is even more crucial for embedded systems that cannot be physically tested in their real environment, like space crafts whose safety only relies on verification tools.

In this article, we introduce preliminary results concerning the abstract interpretation [3] of hybrid discrete-continuous systems. Related work can be found in [6, 8, 4]. A simple abstraction of sets of continuous functions is presented in Section 2. This domain is used in Section 3 to perform a simple static analysis. Finally, future work directions are discussed in Section 4.

2 Abstraction of Continuous Inputs

Continuous inputs usually are modeled by differential equations. Let R^+ be the set of non-negative real numbers and let $\overline{\mathsf{R}} = \mathsf{R} \cup \{-\infty, +\infty\}$. Let C^1_+ denote the set of continuous piecewise smooth functions $\mathsf{R}^+ \to \overline{\mathsf{R}}$. C^1_+ is partially ordered by the relation $\forall f_1, f_2 \in \mathsf{C}^1_+,\ f_1 \dot{\prec} f_2 \Leftrightarrow \forall t \in \mathsf{R}^+,\ f_1(t) \leq f_2(t)$. Next, $\prec$ is defined by $(f_1, f_2) \prec (g_1, g_2) \Leftrightarrow g_1 \dot{\prec} f_1 \wedge f_2 \dot{\prec} g_2$. There is a first Galois connection:

$$\mathcal{T}_0 = \langle \wp(\mathsf{C}^1_+), \subseteq, \cup, \cap, \top_\subseteq, \bot_\subseteq \rangle \underset{\alpha_0}{\overset{\gamma_0}{\leftrightarrows}} \langle (\mathsf{C}^1_+)^2_\bot, \prec, \vee, \wedge, \top_\prec, \bot_\prec \rangle = \mathcal{T}_1 \qquad (1)$$

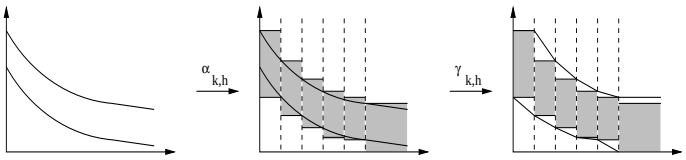

Fig. 1. Example of the transformations performed by $\alpha_{k,h}$ and $\gamma_{k,h}$.

where $\wp(X)$ is the powerset of X, $\top_{\subseteq} = \mathsf{C}^1_+$, $\bot_{\subseteq} = \emptyset$, $(\mathsf{C}^1_+)^2_\bot = (\mathsf{C}^1_+ \times \mathsf{C}^1_+) \cup \{\bot_\prec\}$, $\top_\prec = (f_{\bot_\prec} \ : \ x \mapsto -\infty, f_{\top_\prec} \ : \ x \mapsto +\infty)$, $f_1 \vee f_2 = f$ such that $\forall x \in \mathsf{R}^+,\ f(x) = f_1(x)$ if $f_1(x) \geq f_2(x)$ and $f(x) = f_2(x)$ otherwise. $\wedge$ is defined in the same way. Note that for all $f_1 \in \mathsf{C}^1_+$, $f_2 \in \mathsf{C}^1_+$, $f_1 \vee f_2 \in \mathsf{C}^1_+$ and $f_1 \wedge f_2 \in \mathsf{C}^1_+$.

$\alpha_0(X) = (f^-, f^+)$ such that $\forall x \in \mathsf{R}^+,\ f^-(x) = \inf\{f(x) \ : \ f \in X\}$ and $f^+(x) = \sup\{f(x) \ : \ f \in X\}$. For all $X \subseteq \mathsf{C}^1_+$, α_0 computes functions belonging to C^1_+, at least if, in $\mathcal{T}_0$, we restrict ourselves to families of functions F for which $\{x \ : \ \exists f \in F,$ the derivative of f is discontinuous in $x\}$ is finite (we believe that this assumption can be relaxed). $\gamma_0(f^-, f^+) = \{f \in \mathsf{C}^1_+ \ : \ f^- \dot{\prec} f \dot{\prec} f^+\}$.

In a static analyzer, the user ought to specify the continuous inputs as values of $\mathcal{T}_1$, i.e. as pairs of functions that the tool will numerically approximate. The second abstraction, introduced now, defines how safe approximations of sets of functions should be computed by the analyzer. Basically f^- and f^+ are conservatively approximated by step functions, (see Figure 1). Let F be a set of floating-point numbers and let E^+ denote the set of step functions $e \ : \ \mathsf{R}^+ \to \overline{\mathsf{F}}$.

$$\mathcal{T}_1 = \langle (\mathsf{C}^1_+)^2_\bot, \prec, \vee, \wedge, \top_\prec, \bot_\prec \rangle \underset{\alpha_{k,h}}{\overset{\gamma_{k,h}}{\leftrightarrows}} \langle \mathsf{E}_+ \times \mathsf{E}_+, \prec, \vee, \wedge, \top_\prec, \bot_\prec \rangle = \mathcal{T}_2 \qquad (2)$$

k is a positive integer and h is a positive floating-point number. $\alpha_{k,h}(f^-, f^+) = (\alpha^-_{k,h}(f^-), \alpha^+_{k,h}(f^+))$ such that:

$$\alpha^-_{k,h}(f^-) = e^- \ : \ \forall i \ : \ 0 \leq i < k,\ \forall t \ : \ ih \leq t < (i+1)h,\ e^-(t) \leq f^-(t) \qquad (3)$$

$$\alpha^+_{k,h}(f^+) = e^+ \ : \ \forall i \ : \ 0 \leq i < k,\ \forall t \ : \ ih \leq t < (i+1)h,\ e^+(t) \geq f^+(t) \qquad (4)$$

$$\forall t \geq kh,\ e^-(t) \leq f^-(t) \text{ and } e^+(t) \geq f^+(t) \qquad (5)$$

Equations (3-5) do not indicate how to compute $\alpha^-_{k,h}$ and $\alpha^+_{k,h}$, but any implementation that conforms to these conditions is a correct abstraction. We believe that standard numerical algorithms can be adapted to compute $\alpha^-_{k,h}$ and $\alpha^+_{k,h}$. For instance, for functions of $\mathcal{T}_1$ defined by differential equations, the first k steps of e^- and e^+ could be computed using a Runge-Kutta method, and using the IEEE Standard 754 rounding modes towards $-\infty$ for e^- and towards $+\infty$ for e^+ [2]. For the last step, corresponding to time $t \geq kh$, safe approximations can also be computed. For example, if f^- and f^+ belong to C^2_+ (their first derivative is piecewise smooth) the sign of their second derivative may help to find a fine approximation. Interesting abstractions could also be designed for periodic functions (e.g. in Fourier's space), like the trigonometric functions, which

often correspond to realistic physical inputs of digital signal processing algorithms used in embedded software [10]. In addition, for $k' > k$, $\alpha_{k',h}$ is a finer abstraction than $\alpha_{k,h}$. A similar property holds for h.

$\gamma_{k,h}$ has to compute a pair of functions of C^1_+ but, e.g., $\vee\{f \in \mathsf{C}^1_+ \ : \ f \prec e^+\} \notin \mathsf{C}^1_+$. We have to perform an additional approximation and to chose, for example, to let $\gamma^+_{k,h}(e^+)$ be a linear interpolation function linking the points $(ih, e^+(ih))$ to $((i+1)h, e^+((i+1)h))$, for $0 \leq i < k$, as shown in Figure 1.

3 A Simple Analysis

Computer calculations are carried out using floating-point numbers that introduce roundoff errors. For critical embedded systems, a major safety property should consist of proving that programs take the same decisions than ideal systems working with real numbers. Let p be a program whose inputs are given by sensors corresponding to volatile variables $x_1, \ldots, x_n$. For $1 \leq i \leq n$, $F_i \subseteq \wp(\mathsf{C}^1_+)$ is a set of possible concrete functions modeling the sensor related to x_i. $[\![p]\!]_{\mathsf{R}}$ and $[\![p]\!]_{\mathsf{F}}$ denote the collecting semantics of p in which calculations are carried out with real number and floating-point numbers, respectively. We wish to compare the traces of $[\![p]\!]_{\mathsf{R}}(F_1, \ldots, F_n)$ and $[\![p]\!]_{\mathsf{F}}(F^\sharp_1, \ldots, F^\sharp_n)$, where $F^\sharp_i = \alpha_0 \circ \alpha_{k,h}(F_i)$, $1 \leq i \leq n$. We assume that loops are precisely cadenced, that is that we know how long lasts an iteration. This is usually known since embedded software are real-time programs for which the designer knows, e.g. that some loop runs at x Hz. So we have at least a precise interval of time to date each point of an execution trace. WCET tools also finely compute this information [5].

To perform our analysis, we define a simple concrete semantics $[\![.]\!]$ in which a value is a pair of $\mathsf{F} \times \mathsf{R}$ written $v = f\boldsymbol{\varepsilon}_f + e\boldsymbol{\varepsilon}_e$. $\boldsymbol{\varepsilon}_f$ and $\boldsymbol{\varepsilon}_e$ are formal variables. This semantics, detailed in [11], is a special case of a family of more informative semantics [9]. For example, for $v_1 = f_1\boldsymbol{\varepsilon}_f + e_1\boldsymbol{\varepsilon}_e$ and $v_2 = f_2\boldsymbol{\varepsilon}_f + e_2\boldsymbol{\varepsilon}_e$, the sum is defined by $[\![v_1 + v_2]\!] = \uparrow_\circ (f_1 + f_2)\boldsymbol{\varepsilon}_f + [e_1 + e_2 + \downarrow_\circ (f_1 + f_2)]$. For $x \in \mathsf{R}$, $\uparrow_\circ (x) \in \mathsf{F}$ is the roundoff of x and $\downarrow_\circ (x) = x - \uparrow_\circ (x) \in \mathsf{R}$. In the abstract semantics $[\![.]\!]^\sharp$ related to $[\![.]\!]$, the values are made of pairs of $\mathsf{I}_{\mathsf{F}} \times \mathsf{I}_{\mathsf{MP}}$, i.e. an interval of floating-point numbers and an interval of multiple precision floating-point numbers closely approximating a set of real numbers. We can now define the abstract semantics of a volatile variable x_i in the time interval $[t, t']$:

$$[\![x_i]\!]^\sharp = [a, b]\boldsymbol{\varepsilon}_f + [-u, u]\boldsymbol{\varepsilon}_e \tag{6}$$

with $F^\sharp_i = (F^{-\sharp}_i, F^{+\sharp}_i)$, $a = \min(F^{-\sharp}_i([t, t']))$, $b = \max(F^{+\sharp}_i([t, t']))$, and $u = \max(ulp(|a|), ulp(|b|))$. $ulp(x)$ is the unit in the last place of x [7].

The analysis defined by $[\![.]\!]^\sharp$ allows one to detect divergences of control flow between $[\![.]\!]_{\mathsf{R}}$ and $[\![.]\!]_{\mathsf{F}}$, that is to detect different decisions taken by an ideal program and its implementation. For a value $v = f\boldsymbol{\varepsilon}_f + e\boldsymbol{\varepsilon}_e$ of $[\![.]\!]^\sharp$, they correspond to conditions $cond(v)$ for which $[\![cond(v)]\!]_{\mathsf{F}} = cond(f) \neq cond(f + e) = [\![cond(v)]\!]_{\mathsf{R}}$.

4 Future Work

First of all, the analysis of Section 3 is too simple to be useful in practice. Indeed, "small" divergences of control flow between $[\![.]\!]_{\mathsf{R}}$ and $[\![.]\!]_{\mathsf{F}}$ are common

and admitted. However, $[\![.]\!]^\sharp$ shows how we technically plan to address more interesting problems. To allow "small" divergences of control flow, the safety property expressed in Section 3 should be reformulated as "given a program which, in $[\![.]\!]_{\mathsf{R}}$, reaches some control point between the instants t and t', when does the same program, in $[\![.]\!]_{\mathsf{F}}$, reach the same control point?". For example, if an alarm must be raised when the value of a sensor reaches a certain threshold at time t, we wish the implementation raise the same alarm in an acceptable delay around t and we want to bound this delay.

Secondly, as discussed in Section 2, conservative algorithms computing safe approximations $\alpha_{k,h}$ satisfying the correctness criteria of equations (3-5) have to be designed. Next, we want to add the interactions from programs to their environment (actuators). This makes the continuous functions related to the volatile variables change dynamically, significantly complicating the analysis.

Finally, software for embedded systems often are concurrent programs interacting together by classical communications between computer programs but also via their actions on the shared physical environment. We also plan to extend our work to the case of concurrent systems.

References

1. R. Alur, C. Courcoubetis, N. Halbwachs, T. A. Henzinger, P.-H. Ho, X. Nicollin, A. Olivero, J. Sifakis, and S. Yovine. The algorithmic analysis of hybrid systems. *Theoretical Computer Science*, 138, 1995.
2. ANSI/IEEE. *IEEE Standard for Binary Floating-point Arithmetic*, 1985.
3. P. Cousot and R. Cousot. Abstract interpretation: a unified lattice model for static anal ysis of programs by construction or approximation of fixpoints. In *POPL'77*, pages 238–252. ACM Press, 1977.
4. Blondel V. et al. Deciding stability and mortality of piecewise affine dynamical systems. *Theoretical Computer Science*, 255, 2001.
5. C. Ferdinand, R. Heckmann, M. Langenbach, F. Martin, M. Schmidt, H. Theiling, S. Thesing, and R. Wilhelm. Reliable and precise wcet determination for a real-life processor. In *EMSOFT'01*, number 2211 in LNCS. Springer Verlag, 2001.
6. J. Feret. Static analysis of digital filters. In *ESOP'04*, number 2986 in LNCS, pages 33–48. Springer-Verlag, 2004.
7. D. Goldberg. What every computer scientist should know about floating-point arithmetic. *ACM Computing Surveys*, 23(1), 1991.
8. N. Halbwachs, P Raymond, and Y.-E. Proy. Verification of linear hybrid systemss by means of convex approximations. In *SAS'94*, LNCS. Springer Verlag, 1994.
9. M. Martel. Propagation of roundoff errors in finite precision computations: a semantics approach. In *ESOP'02*, number 2305 in LNCS. Springer-Verlag, 2002.
10. M. Martel. Validation of assembler programs for dsps: A static analyzer. In *PASTE'04*. ACM Press, 2004.
11. M. Martel. An overview of semantics for the validation of numerical programs. In *VMCAI'05*, number 3385 in LNCS. Springer-Verlag, 2005.
12. P. J. Mosterman. An overview of hybrid simulation phenomena and their support by simulation packages. In *Hybrid Systems: Computation and Control*, LNCS, pages 165–177. Springer Verlag, 1999.