A Symbolic Approach for the Safety Verification of Continuous Systems

Mohamed H. Zaki, Sofiène Tahar and Guy Bois §

Dept. of Electrical & Computer Engineering, Concordia University
1455 de Maisonneuve W., Montreal, H3G 1M8, Canada
§Genie Informatique, Ecole Polytechnique de Montreal
C.P. 6079, succ. Centre-Ville, Montreal, H3C 3A7 , Canada
{mzaki, tahar}@ece.concordia.ca, guy.bois@polymtl.ca

Abstract. Embedded systems are computer-based reactive designs that interact directly and dynamically with their environment. When embedded systems are used in safety critical applications, the design process must be able to give safety guarantees to avoid any consequent risks. In this paper, we propose an approach to check safety properties of the dynamical environment of embedded systems using the computer algebra system Maple and the constraints solver RealPaver. The methodology we present is suitable when the dynamics are described by a set of polynomial ordinary differential equations, and the safety conditions in the properties are described as constraints using algebraic inequalities. Such modeling can be useful in representing classes of embedded systems (e.g., optical, mechanical, analog) at functional and behavioral levels.

1 Introduction

Embedded systems are reactive in nature and interact directly and dynamically with their environment. Such interaction is usually achieved through sensors and actuators which are interfaced with the controller CPU through data processing components converting data representations between analog and digital formats. At the specification level, such architecture can be modeled in an abstract way as shown in Figure 1.(a), where the digital controller can be modeled by finite state machines (FSMs), while the dynamical environment can be described using systems of ordinary differential equations (ODEs). The sensor and A/D interface can be modeled as threshold detector and event generator respectively, while the actuator and D/A components can be modeled as switches that choose between different ODEs systems and set the initialization and reset conditions necessary for correct functionality.

When embedded systems are safety critical, the design process must be able to provide safety guarantees to avoid any life threatening risks. An example of a safety requirement is the switching condition of the dynamical parts of an embedded system, e.g., check whether a continuous behavior will trigger a discrete event driving the system into an undesired mode. Simulation is classically used to verify such a property. As exhaustive simulation of all possible scenarios is impossible, and hence, cannot guarantee the correctness of the design. With contrast to simulation, formal verification techniques aim to prove mathematically that a circuit behaves correctly for all possible input signals and initial conditions and that none of them drives the system into an undesired behavior. Therefore such safety properties can be typically verified using formal methods like model checking [1]. However, the effectiveness of model checking is severely constrained by the state space explosion problem and even undecidability limitations when the systems are described by differential equations [5]. In such cases, abstraction techniques are usually required in order to achieve the verification task.

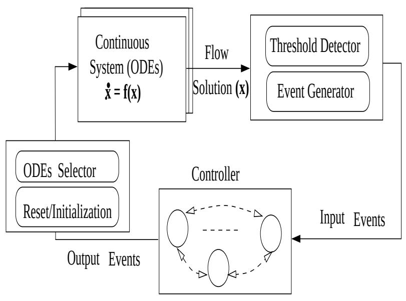
In this paper, we propose a computer algebra approach to check safety properties of the continuous dynamics of generic embedded systems as described in Figure 1.(a). The basic idea is the symbolic extraction of qualitative constraints of the continuous system described using a system of ODEs, which can be automatically checked against a set of specification properties written in a temporal logic [1]. For verification purposes, we combine ideas from invariant generation, linear programming and constraint solving on designs described by non-linear differential equations using the computer algebra system Maple [7] and the constraints solver RealPaver [9].

Related Work. Safety verification has been applied recently to embedded systems using abstraction based verification [5, 11]. In order to enhance the abstraction, the authors in [12, 10] proposed independently to use techniques from algebraic geometry to generate general invariants. In this paper, we are interested in qualitative invariants which provide useful information about the behavior, avoiding the generation of redundant or hard to interpret invariants. In [15], the authors proposed a similar framework using the idea of barrier certificates. Barrier certificates if they exist, are invariants that separate system behavior from a bad state. Such method is complementary to ours as the invariants we use do not require a priori knowledge of initial conditions and in contrast to barrier certificates give knowledge of the whole system behavior rather than specific behaviors. In [16], the authors used invariants with predicate abstraction to build discrete models for analog circuits to verify oscillation behavior using model checking.

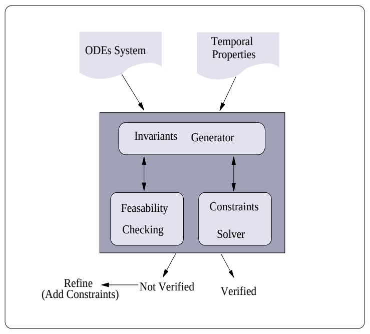
The rest of the paper is organized as follows: We start with an overview of the verification methodology in Section 2. We proceed with describing the system invariants and their generation in Section 3. Finally, we describe safety verification of continuous systems with illustrative examples in Section 4, before concluding with a discussion in Section 5.

2 Verification Approach

The basic idea of the proposed verification is to qualitatively divide the continuous state space of the dynamics of the embedded system into distinct invariant regions on which satisfaction of safety properties can be verified. This qualitative analysis is based on the Darboux theory of integrability [6]. The approach we propose is illustrated in Figure 1.(b). Starting with a system of ODEs, along with specification properties written in computational temporal logic ($\forall$CTL) [1], we construct a system of constraints using properties constraints along with the qualitative invariants symbolically extracted from the ODEs system.



(a) Embedded System Model



(b) Verification Methodology

Fig. 1. Verification for Continuous Systems

For a *safety property* specified in temporal logic formula, e.g., $\forall\Box\neg p$, (i.e., always, for all possible execution, the constraint p will be satisfied), we get the dual property $\exists\Diamond\neg p$ (which means that there is an execution falsifying the constraint p) and apply feasibility checking on it within the invariant regions of interest. If the constraints system is satisfiable, we conclude that the property is violated (cannot be verified) otherwise, the property is verified.

Based on the nature of the system of constraints, we apply satisfiability decision procedures; feasibility checking for linear constraints based systems and constraints solving for non-linear constraints based systems. For linear constraints, *Fourier-Motzkin Elimination* method [2] is used to check for feasibility. In non-linear constraints, a constraint solving approach based on interval arithmetics [9] is used to test the unsatisfiability of the constraints. The construction of the constraints is incremental in the sense that more precision can be achieved by adding more information to the original construction of the system. When the property is marked violated, one possible reason is because of the false negative problem due to the over approximation of the abstraction. In this case refinement techniques may be introduced.

3 State Space Invariants

System Description: We consider the dynamics of the embedded system model in Figure 1.(a), which can be described by non-linear polynomial ODEs of the form:

$$\dot{x_k} = \frac{dx_k}{dt} = \mathcal{P}_k(x_1, \dots, x_d) = a_0 + \sum_{l=1}^{m} \mathcal{P}_{l,k}(x_1, \dots, x_d)$$

where t is the independent real time, $k = 1, \dots, d$ and d is the system order. $\mathcal{P}_k$ is a polynomial of degree m, a_0 is a constant and $\mathcal{P}_{l,k}$ is a polynomial of degree l,

$$\mathcal{P}_{l,k} = \sum_{i_1+\ldots+i_d=l} a_{i_1,\ldots,i_d} x_1^{i_1} \ldots x_d^{i_d}$$

where $a_{i_1,\ldots,i_d}$ is a constant. We assume that the differential equation has a unique solution for each initial value.

Usually, a continuous system has a behavior which varies in different regions of state space. The regions' boundaries are usually defined by special solutions of the system known in the literature as *separatrices* which partition the concrete state space into a set of qualitative distinctive bounded regions.

Definition 1. *Given the system of ODEs* $\frac{dx_k}{dt} = \mathcal{P}_k(x_1(t), \dots, x_d(t))$*, with* $k = 1, \dots d$ *(*$\frac{d\mathbf{x}}{dt} = \mathbf{P}(\mathbf{x})$*,* $\mathbf{x} \in \mathbb{R}^d$ *and* $\mathbf{P} = (\mathcal{P}_1, \dots, \mathcal{P}_d)$*), we can define a corresponding vector field as*

$$\mathcal{D}_{\mathbf{P}} = \mathbf{P}.\partial_{\mathbf{x}} = \sum_{k=1}^{d} \mathcal{P}_k \frac{\partial}{\partial x_k}$$

The correspondence between the system of ODEs and the vector field $\mathcal{D}_{\mathbf{P}}$ is obtained by defining the time derivative of functions of $\mathbf{x}$ as follows. Let $\mathcal{G}$ be a function of $\mathbf{x}$: $\mathcal{G} : \mathbb{R}^k \to \mathbb{R}$, then $\frac{d\mathcal{G}}{dt} := \dot{\mathcal{G}} = \mathcal{D}_{\mathbf{P}}(\mathcal{G}) = \mathbf{P}.\partial_{\mathbf{x}}\mathcal{G}$. The time derivative is called the derivative along the flow since it describes the variation of function $\mathcal{G}$ of $\mathbf{x}$ with respect to t as $\mathbf{x}$ evolves according to the differential system. We look at functions which are constant on their zero level set. Darboux polynomials $\mathcal{J}_i$ were first studied by Darboux in the last century, since then have been investigated in the qualitative and algebraic analysis of continuous systems. Darboux polynomials split the phase portrait into regions where the behavior is qualitatively different (bound or unbound). These functions $\mathcal{J}_i$ provide the essential skeleton for the state space from which all other behaviors can be qualitatively determined. A Darboux polynomial is of the form

$$\mathcal{J}(\mathbf{x}) = 0 \textit{ when } D\mathcal{J} = \mathcal{K}\mathcal{J}$$

with $\mathcal{J} \in \mathbb{R}[\mathbf{x}]$ and $\mathcal{K} = \mathcal{K}(\mathbf{x})$ is a polynomial called the cofactor of $\mathcal{J} = 0$, with a degree of at most $\mathcal{M} - 1$.

Property 1 *Given a system of ODEs and a vector field* $\mathcal{D}_{\mathbf{f}}$*,* $\mathcal{J}$ *is an invariant of the the system if* $\mathcal{J}$ *divides* $\mathcal{D}_{\mathbf{f}}$*, more formally, if there exists* $\mathcal{K} \in \mathbb{R}[\mathbf{x}]$ *such that* $\mathcal{D}_{\mathbf{f}}(\mathcal{J}) = \mathcal{K}\mathcal{J}$*. The solution set of the system vanishes on the curve of* $\mathcal{J}$*.*

For example, a straight line $ux + vy + w = 0$ satisfies $u\frac{dx}{dt} + v\frac{dy}{dt} = u\mathcal{P}_1(x, y) + v\mathcal{P}_2(x, y) = (ux + vy + w = 0)\mathcal{K}(x, y)$ if it is invariant under the flow of the system. In case we are only concerned with linear Darboux invariants of the form $\psi(\mathcal{J}x) := a_0 + \sum_{k=1}^{d} a_k x_k$.

Example 1. For the system described by the following ODEs: $\dot{x} = x(a_1x + b_1y + c_1)$ and $\dot{y} = y(a_2x+b_2y+c_2)$ with $a_2 = b_1 = 0$, $c_1 = c_2$ and $a_1b_2 \neq 0$. The number of linear invariants is equal to $3m - 1 = 5$, namely $f_1 = y + \frac{c_1}{b_2}$, $f_2 = x + \frac{c_1}{a_1}$, $f_3 = x - \frac{b_2}{a_1}y$, $f_4 = x$, $f_5 = y$.

Generating Second Integrals: The problem of finding the invariant is based on the evaluation of the coefficients of the predefined forms of Darboux polynomials $\mathcal{J}$ and their cofactors $\mathcal{K}$. This can be considered as the second step of the Prelle Singer algorithm [8] used to build first integrals from Darboux polynomials and was implemented in a Maple package called PSsolver [3].

Example 2. Using PSsolver package, for the system $\dot{x} = x(x^2 + 2y^2 - 1)$ and $\dot{y} = y(x^2 + 2y^2 - 3y + 1)$, we find following Darboux invariant polynomials $j_1 = x - y + 1$, $j_2 = -x - y + 1$, $j_3 = 0.5x - y + 0.5$, $j_4 = -0.5x - y + 0.5$, $j_5 = x$, $j_6 = y$.

4 Safety Verification

We define the invariant regions as a conjunction of Darboux invariant predicates. The invariant regions can be considered as abstraction of the state space such that being inside an invariant region means that the system dynamics will always stay in this region. Moreover, we consider the fact that a safety property $\forall\Box\mathcal{I}$ is always satisfied in a region, if its dual property $\exists\Diamond\neg\mathcal{I}$ is never satisfied in that region.

In the remaining of this section, we show how to derive the verification for two possible cases; namely the linear case, where we use feasibility checking and the non-linear case, where we use constraint solving.

4.1 Feasibility Checking

The problem of determining whether a linear predicate cross an invariant region formed by a conjunction of linear constraints can be formulated as a feasibility problem by solving linear inequalities. The feasibility problem is thus to determine whether the linear constraints are consistent, and if so, find a point that satisfies them; $x \in \mathcal{X}$ is feasible if it satisfies the constraints. The problem can be formulated as solving a system of linear inequalities as follows. Given a matrix $\mathbf{A} = [a_{ij}]$ in $\mathbb{R}^{m\times n}$ and a column vector $\mathbf{b} \in \mathbb{R}^m$, find $\mathbf{x} = (x_1, \ldots, x_n)^T$ such that $\exists\mathbf{x} : \mathbf{A}\mathbf{x} \leq \mathbf{b}$, i.e.,

$$\exists\mathbf{x} : \bigwedge_{1\leq i\leq m} \sum_{1\leq j\leq n} a_{ij}x_j \leq b_i$$

Several methods have been developed to deal with this class of problem. For instance, a standard method is the *Fourier-Motzkin Elimination* [2]. Intuitively, the Fourier-Motzkin elimination procedure iteratively projects one variable x by rewriting the system of inequalities into a new system without x, which has a solution if and only if the original system has a solution. The basic idea of variable elimination is to pick one variable and eliminate it, then continue until all variables are eliminated [2].

Example 3. Suppose that the dynamical environment of the system in Figure 1.(a) is described by $\dot{x} = x^2 + 2xy + 3y^2$ and $\dot{y} = 4xy + 2y^2$, with $x_0 \in]0, 0.5]$ and $y \in]0, 1]$. Suppose the property to check is $\forall \Box x > -1$, meaning that the state variable x valuation will not fall below -1. Based on the methodology in Figure 1.(b) and using the Maple PSsolver package, we found that the system has three invariant lines; $j_1 = y, j_2 = x + y, j_3 = x - y$ (See Figure 2.(a)). We can divide the state space into regions formed by the conjunction of different invariant predicates. Using feasibility checking (with Maple), we start by identifying the regions to which the initial conditions belong. The following regions satisfy the initial conditions $\Theta_1 = (y > 0, x + y > 0, x - y < 0)$, $\Theta_2 = (y > 0, x + y > 0, x - y > 0)$. We check whether $\exists \Diamond x \leq -1$ is satisfiable in the invariant regions Θ_1 and Θ_2 using feasibility checking. We find that for region Θ_1, the constraints system is feasible (original property cannot be verified), while for the region Θ_2, the constraints system is infeasible (property is satisfied).

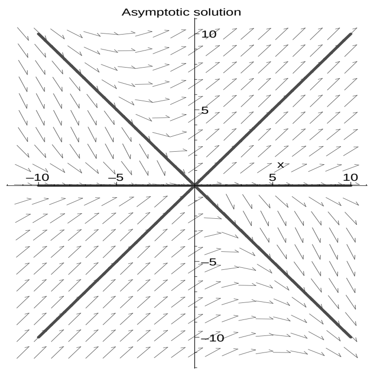
4.2 Verification using Constraints Solving

To verify properties on regions described by a conjunction of non-linear constraints, we use an interval based constraint solving which can decide unsatisfiability for the system of non-linear constraints. The feasibility problem is thus to determine whether the non-linear constraints are consistent. In unsatisfiability constraint solving, if a solver pronounces the infeasibility of the input constraints, then this result is sound. Realpaver [9] is an example tool that can solve constraints of this category. RealPaver is able to solve non-linear equations or inequality constraints over the real numbers where each domain is represented by a closed interval. Given a system of constraints, RealPaver computes a union of boxes that contains all solutions satisfying these constraints. If no box is computed by RealPaver, then this system is guaranteed to have no solution. Interval computation guarantees the solution to be reliable; the real solutions are enclosed by the computed intervals [9].

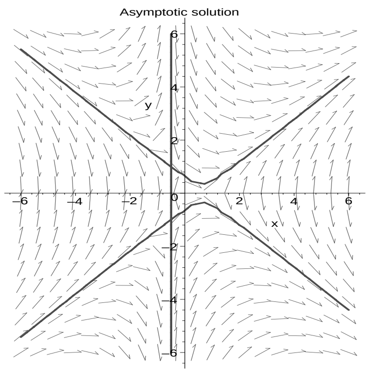
Example 4. Suppose that the dynamical environment of the system in Figure 1.(a) is described by $\dot{x} = y + 2xy$ and $\dot{y} = -x + 2x^2 - y^2$, with the following property $\forall \Box y^2 + x > 0$. The extracted invariants are $\{j_1 = 2x + 1, j_2 = 1 - 2x + \frac{8}{5}x^2 - \frac{12}{5}y^2\}$ (see Figure 2.(b)). Using RealPaver, we find out that the property is only satisfied in regions $\Theta_3 = (j_1 < 0, j_2 < 0)$ and $\Theta_4 = (j_1 > 0, j_2 < 0)$. This means that a flow in another region than regions Θ_3 and Θ_4 can generate a discrete event, which could trigger the control part into an undesired state.

5 Conclusion

The lack of methods for safety verification of continuous dynamics has been the main obstacle towards algorithmic verification methodology for embedded sys-



(a) Linear Invariants



(b) Non-linear Invariants

Fig. 2. Phase Portrait and Invariants of Examples 3 and 4

tems models. In this paper, we present a qualitative based analysis that can be suitable to tackle such problems and can be incorporated naturally with formal verification techniques. The method can be also applied alongside traditional methods like simulation based techniques to raise confidence in the produced system. To apply our approach we have used the computer algebra tool Maple in addition to the RealPaver constraint solver which is suitable for unsatisfiability checking. Future work includes extending the approach to handle discrete components and apply the verification on practical case studies.

References

1. E. Clarke, O. Grumberg , D.A. Peled. Model Checking. MIT Press: Cambridge, MA, 1999.
2. V. Chvatal, Linear Programming book, W.H. Freeman, 1983.
3. L.G. Duarte, S.E. Duarte, L.A. da Mota, J.E. Skea: An Extension of the Prelle-Singer Method and a Maple Implementation. Computer Physics Communications, 144:46-62, Elsevier, 2002.
4. W. Hartong, R. Klausen, L. Hedrich: Formal Verification for Nonlinear Analog Systems: Approaches to Model and Equivalence Checking, Advanced Formal Verification, Kluwer, pp. 205-245, 2004.
5. T.A. Henzinger, P.-H. Ho, and H. Wong-Toi. HyTech: A Model Checker for Hybrid Systems. Software Tools for Technology Transfer, 1:110-122, Kluver, 1997.
6. A. Goriely. Integrability and Nonintegrability of Ordinary Differential Equations, Advanced Series on Nonlinear Dynamics, Vol 19, World Scientific, 2001.
7. MapleSoft inc., www.maplesoft.com/
8. M Prelle and M Singer: Elementary First Integral of Differential Equations. Transactions of the American Mathematical Society, Vol. 279(1), pp. 215-229, 1983.
9. L. Granvilliers. On the Combination of Interval Constraint Solvers. Reliable Computing, 7(6):467-483, 2001

10. S. Sankaranarayanan, H. Sipma, Z. Manna. Constructing Invariants for Hybrid Systems. In Hybrid Systems: Computation and Control, LNCS 2993, pp 539-554, Springer, 2004.
11. A. Tiwari and G. Khanna. Series of abstractions for hybrid automata. In Hybrid Systems: Computation and Control, LNCS 2289, pp. 465-478, Springer, 2002.
12. A. Tiwari and G. Khanna. Nonlinear systems: Approximating reach sets. In Hybrid Systems: Computation and Control, LNCS 2993, pp. 600-614. Springer, 2004.
13. R. E. Moore. Methods and Applications of Interval Analysis, Society for Industrial Applied Mathematics, Philadelphia, 1979.
14. X Zhang. Invariant Hyperplanes and Darboux Integrability of Polynomial Vector Fields, Journal of Physics A: Mathematical and General, Volume 35, Number 46, 124:9931-9941, Institute of Physics Publishing, 2004.
15. S. Prajna, A. Jadbabaie. Safety Verification of Hybrid Systems Using Barrier Certificates. In Hybrid Systems: Computation and Control, Springer, pp. 477-492. 2004.
16. M. Zaki, S. Tahar, and G. Bois: Abstraction Based Verification of Analog Circuits Using Computer Algebra and Constraint Solving; Proc. International Workshop on Symbolic Methods and Applications to Circuit Design, Florence, Italy, 2006.