

Verification of the Safety Properties of Embedded Systems

Maria Kourkouli, George Hassapis,

mkour@auth.gr, chasapis@eng.auth.gr

Dept. of Electrical and Computer Engineering, Aristotle University of Thessaloniki,
54124 Thessaloniki Greece

Abstract. The satisfaction of critical design specifications of embedded systems which consist of continuous time and discrete time components must be verified before their expensive construction and operation is undertaken, for safety and economical reasons. This verification can be done by modelling the system and checking whether the system being in a certain initial state can or cannot reach another state that is considered to be safe or unsafe. Hybrid automata have been used in many cases as one of the possible formalisms for modeling hybrid systems. Its use, however, in modeling large systems is hindered by the fact that the several model-checking algorithms that have been developed for searching the state space of the model are computationally intractable and solve models of certain classes of hybrid automata. These algorithms search for the existence of states in the model which can be associated with the safety specifications. In this work a new algorithm is presented for the verification of hybrid systems, which tries to extend its applicability to a wider number of hybrid automata classes and be computationally more efficient than the model checking algorithms. According to this new algorithm, line geometry is used to compute the distance of a state from another. Each state is modeled as a surface and the verification problem is formulated to the problem of finding the distance between two familiar surfaces. The new algorithm can be applied to more general classes of hybrid systems and verify the safety properties with less computation in polynomial time. Through a case study the applicability of the algorithm is demonstrated. The case study concerns the verification of the safety specifications of a railroad crossing. The system is also modeled and verified using the model checking tool PHAVer. A comparison between the two verification techniques is presented.

Keywords: formal methods, embedded systems, hybrid systems, verification

1 Introduction

1.1 Formalism of Hybrid Automata

Embedded systems comprise one of the most important applications of computer systems and components [1]. These systems are used in automotive and aircraft controllers, cellular phones, network switches, household appliances, medical equipment,

plant control and consumer electronics. The term embedded system is used to denote systems that are designed to interact with a physical environment in real time, through sensors and actuators, in order to fulfill control objectives and design specifications. They are composed of hardware and software components, and involve computations that are subject to physical constraints. These systems are designed to do some specific task rather than be a general computer for multiple tasks. Usually, they are part of a bigger system. Designing systems with such characteristics is a difficult task and involves the realization of the control functions and the communication of the embedded computer with other machines and humans. As these systems consist of continuous time and discrete time components hybrid automata seems to be an appropriate formalism for modeling the dynamic behavior of such systems.

The continuous components are modelled by differential equations and the discrete events are modelled by finite state machines. These models are built by considering that a hybrid automaton evolves over the time as a sequence of steps following a set of abstractions which can be mathematically related between each other. These abstractions are the continuous states and discrete locations of the hybrid automaton and the conditions under which the system either moves from a location to another or stays in a location. In a location the system evolves continuously by changing the values of the continuous time variables according to dynamical laws defined by the differential equations of the continuous components. The set of values that these variables have at certain timing instant constitute the state of the system. These values however may present discontinuities whereas the description of the system dynamical behavior after the discontinuity is described by a different set of differential equations. The discontinuity conditions under which the values are changed abruptly are of two types. Those that are originated by the resets and the jumps of the embedded systems and those that allow the system to be described by the set of equations of another location. The last conditions are expressed in the form of ranges of variable values or relationships of the continuous time variables. When they become true they allow the transition of the system from a state of one location to the state of another location. The first type of conditions is called invariants and the second type guards. This modeling technique comprises the hybrid automaton [2] formalism. Formal methods have a special place in computer – based systems development when correct functioning is critical, as for safety or for security reasons. Over the last ten years the hybrid automaton formalism has been used as a modeling technique of a large number of real time embedded applications for verifying the compliance of the design of the control functions of applications to desired specifications or safety properties before proceeding to the development and the engineering of these systems.

1.2 Verification problem

The problem that has received much recent research attention has been the verification of the safety properties of embedded systems. Verification of safety properties of such systems is a critical problem because of the interaction between the discrete and continuous dynamics. Safety is expressed as the property that a system has of not entering in an unsafe state and in the specifications is usually expressed as a formula in appropriate logics. Given a specification formula encoding of a safety property, the

task is to determine whether a desired or undesired state of a system model, related with this safety requirement, can be reached from an initial state. In other words, given a set of possible initial conditions and inputs and a finite time horizon, one wishes to identify those initial conditions and inputs that can guarantee safe behaviour at any time. While the correct and safe operation is critical in several ways, complexity grows enormously. For these reasons determining whether a system satisfies a given property of interest, becomes an absolute necessity. In a formal way, the problem of verifying the satisfaction of security and safety specifications of a hybrid system can be expressed by a symbolic model checking approach as follows. Given a class of hybrid system automata models H and a class of specification properties P , the aim is to find a computational procedure that can decide if there is a state in the state space of the model that satisfies the specification and whether this state can be reached from an initial state in a finite number of computational steps. If this procedure expressed in the form of an algorithm exists then the verification problem is called decidable. In the last years, a variety of verification techniques have been developed [3]. The difficulties that are presented in the verification of embedded systems are due to the uncountable number of distinct states in continuous state space. In order to design and implement a methodology for the safety verification it is necessary to be able to represent reachable sets of continuous systems and evolve them according to the system's dynamics.

Automated formal methods for verification that include deductive approaches and model checking are some of the developed techniques. The developed tools are classified according to how the sets of states are represented and the assumptions on the dynamics under which states are progressing. Model checking algorithms require the computation of the set of reachable states of a model. A few software tools have been built to support the model checking approach for solving the state reachability problem, such as Kronos [4], Hytech [5], Uppaal [6], d/dt [7] and HyVisual [8], Phaver [9]. The tools are developed in order to simulate, design and verify hybrid and embedded systems. They limit the continuous dynamics to simple abstractions such as rectangular inclusions and then compute the set of reachable states exactly and effectively by symbolic manipulation of linear inequalities. As the systems that are controlled get more complex, abstraction technique is a necessity so that the available computation tools will be able to cope with the increasing complexity. One abstraction theory is to relax the requirement that the abstraction is equivalent to the original system, and replace it with an abstraction that is only approximately equal to the original system. All these theories consider a metric that can quantify the difference between the system and its abstraction, that is the quality of the abstraction [10]. First order or higher order logics have been used for specification and verification. Such logics are very expressive for modeling complex embedded behavior but it is difficult to design algorithms to check whether a decision problem is decidable and computable. On the other hand, we can approximate the set of reachable states by polyhedral or ellipsoids using optimization techniques. There are two strategies for approximation, under approximation and over approximation. We approximate a reachable state space with a representation for its subset/superset in order to reduce the manipulation complexity and the representation. Symbolic model checking techniques can be used to compute automatically the reachable state space of a hybrid system. They are based on verification algorithms that perform reachability analysis to check whether trajec-

tories of the hybrid system can reach certain desirable/undesirable regions of state space. These algorithms have the drawback that they are in danger of never terminating if they are applied to systems with infinite state space. To circumvent this problem an algorithm is presented in this work, which is based on line geometry and avoids the process of searching the state space. The distance between two points is calculated in order to approximate the next state. The algorithm is presented in detail in next section.

The next section provides some background information for the hybrid automata formalism and for the distance computation problem, useful to understand the new algorithm. In section 3 the proposed algorithm is presented. In section 4 the application of the proposed method to the verification of the safe operation of an embedded system that is designed to control a railroad crossing is demonstrated. The system is also modeled and verified using the model checking tool PHAVer and a comparison between the two verification techniques is made. The conclusions that are drawn from this work are given in section 5.

2 Hybrid systems and Distance Computation

Figure 1 illustrates the modeling of the behavior of a system by the hybrid automaton formalism. Hybrid automata are finite automata enriched with a finite set of real – valued variables. The components of a hybrid automaton are the set of locations, which includes the initial state, the variables, the continuous state space, the set of events and the set of transitions. As it is illustrated in figure 1 the initial state is S1. The continuous system progresses according to the differential equations shown in Figure 1. When the guard1 is true the transition from S1 to S2 is enabled. The system enters in location S2 and is evolving according to the new equations f_{21} , f_{22} and f_{23} until the guard2 becomes true and the transition from S2 to S1 is enabled. Invariant or reset conditions can be associated with the locations. The state of a hybrid automaton is given by the couple $(S, f(x,y,z))$ which associates a location with the value of the continuous state vector. During the evolution of the system we want to check if the system arrives in a desirable/undesirable state.

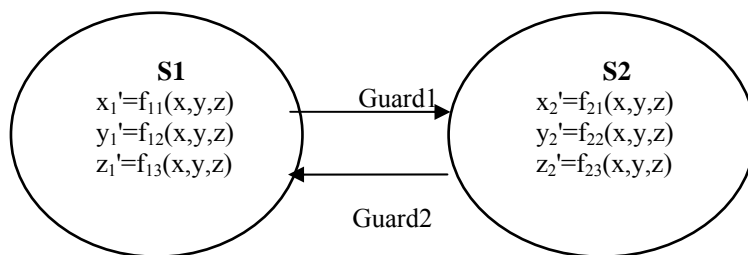


Fig. 1. Hybrid System modeled by hybrid automaton

It is assumed that the solution of the differential equations in location S1 corresponds to a familiar surface such as an ellipsoid. The first location can be characterized by the function f_1 and the second location is characterized by the function f_2 , where f_1 and f_2 are the respective functions originating from the solution of x_1', y_1', z_1' and x_2', y_2', z_2' . The functions can have the following form:

$$f_1 = \left(\frac{x}{A_1}\right)^2 + \left(\frac{y}{B_1}\right)^2 + \left(\frac{z}{C_1}\right)^2 - w_1 \quad (1)$$

$$f_2 = \left(\frac{x}{A_2}\right)^2 + \left(\frac{y}{B_2}\right)^2 + \left(\frac{z}{C_2}\right)^2 - w_2 \quad (2)$$

where w_1 and w_2 are constants and $A_i, B_i, C_i, i = \{1,2\}$ are derived from the differential equations.

As an example consider that $f_{11}=2x/4, f_{12}=2y/9, f_{13}=2z/25, f_{21}=2x, f_{22}=2y/4$ and $f_{23}=2z/6$, then the f_1 and f_2 are respectively:

$$f_1 = \left(\frac{x}{2}\right)^2 + \left(\frac{y}{3}\right)^2 + \left(\frac{z}{5}\right)^2 \quad (3)$$

and

$$f_2 = \left(\frac{x}{1}\right)^2 + \left(\frac{y}{4}\right)^2 + \left(\frac{z}{6}\right)^2 \quad (4)$$

Given two surfaces f_1 and f_2 the distance computation problem between them is defined as the distance between two points, that is one point that belongs in surface f_1 and the other that belongs in surface f_2 :

$$d = \{ |p-q|, p \in f_1 \text{ and } q \in f_2 \} \quad (5)$$

For solving the problem of the distance computation between two familiar surfaces there are a lot of efficient ways [11], [12]. Computing the distance between two known surfaces requires the manipulation of nonlinear equations. For solving a system of non linear equation there are many techniques such as the conjugate direction method and the Newton–Raphson method. Some common techniques used to calculate the distance between two surfaces are the Lagrange technique [11], the Voronoi regions [15], the Minkowski distance [16] and the use of line geometry [13]. Computing the distance between two surfaces based on line geometry requires the use of line coordinates [13]. Each point on a surface is associated with the line passing through the point and spanned by the normal of the surface at that point. The normal congruence of a surface is defined as a set of lines associated with the points on the surface. By using line geometry, we are able to transform the distance computation to a system of two equations in two variables. Starting from an initial point $p = (p_1, p_2, p_3)$ (this point is identified with the initial state of the system model) and considering the following ellipsoid in a general form

$$f = \left(\frac{x_1}{A}\right)^2 + \left(\frac{y_1}{B}\right)^2 + \left(\frac{z_1}{C}\right)^2 - 1 = 0 \quad (6)$$

there will be a $q = (q_1, q_2, q_3)$, in the same surface or in another surface, that satisfies the condition for d , as it is expressed in equation 5. There exists also a value of t in order to satisfy the equation:

$$(p_1, p_2, p_3) - (q_1, q_2, q_3) = t \nabla f(q_1, q_2, q_3) = t \left(\frac{2q_1}{A^2}, \frac{2q_2}{B^2}, \frac{2q_3}{C^2} \right) \quad (7)$$

Solving equation (4) for q we get

$$q_1 = \frac{A^2 * p_1}{A^2 + 2 * t} \quad (8)$$

$$q_2 = \frac{B^2 * p_2}{B^2 + 2 * t} \quad (9)$$

$$q_3 = \frac{C^2 * p_3}{C^2 + 2 * t} \quad (10)$$

and replacing it on equation (6) we obtain a new polynomial equation in t of degree 6.

$$\left(\frac{A^2 * p_1}{A^2 + 2 * t}\right)^2 + \left(\frac{B^2 * p_2}{B^2 + 2 * t}\right)^2 + \left(\frac{C^2 * p_3}{C^2 + 2 * t}\right)^2 - 1 = 0 \quad (11)$$

Having found that there are real roots of this equation is a proof of the existence of a distance from the point p to point q .

3 Verification Algorithm

The presented modeling formalism in association with the theory of the distance computation leads to an algorithm for the solution of the verification problem. The state space for the first location in figure 1 is described by the surface $f_1=0$ and the state space for the second location is described by the surface $f_2=0$. If the guard is true the transition is enabled and we compute the distance from the point on surface f_1 to another point on surface f_2 . That is, we compute the distance between a point P and an ellipsoid. If the transition is not enabled, which means that the guards are not satisfied, we compute the distance from the point on surface f_1 to another point on surface f_1 , when the invariants are satisfied. That is, we compute the distance between a point P and the ellipsoid in this location where this point belongs. The problem that is posed is the distance computation between the two surfaces using one from the developed

methods. Each point on a surface is associated with another. We approach the desirable state by iteratively finding a sequence of point ellipsoid distances, each of which is computed by solving the polynomial equation (11) of degree six in one variable. The algorithm is terminated in the following cases: (a) the desirable state is reached, (b) the state lies in a forbidden region, (c) the calculation leads to a repeated calculation and the system arrives in deadlock, (d) the calculated point does not encode a feasible run according to the definition of hybrid automaton, (e) the equation (11) has not real roots. The above algorithm is illustrated below in the figure 2. In this algorithm P is the initial state, P' and Q' are the new generated points that satisfy the conditions for the equation 7, "Guards" is the set of guards that has to be satisfied for enabling the transition, "Desirable" is the set of desirables states.

```

P(p1,p2,p3) ∈ fl      % initial state

if P satisfies Guards      (i)
  then Go to the next location in a point Q(q1,q2,q3) |{(P-Q)=t∇f(Q)}
    if Q ∈ Desirable      (ii)
      then END
    else Go to another point Q' into the same location |{(P-Q')=t∇f(Q')}
      P:=Q'
      Q:= Q'
      Go to (ii)
    end
  else Go to another point P' into the initial location |{(P-P')=t∇f(P')}
    P:=P'
    Go to (i)
  end

```

Fig. 2. Presented Algorithm

A similar approach can be used in computing the distance between a point and every other surface, such as a cylinder, cone or torus. In the case of a cylinder the used equations have the form:

$$f = x_1^2 + x_2^2 - R^2 = 0 \quad . \quad (12)$$

$$\nabla f = (2x_1, 2x_2, 0) \quad . \quad (13)$$

We get

$$(p_1, p_2, p_3) - (q_1, q_2, q_3) = t^*f(q_1, q_2, q_3) = t^*(2x_1, 2x_2, 0) . \quad (14)$$

The model checking algorithms are faced with two potential types of explosion, time and space. In order to avoid state space explosion they use symbolic representation of the state spaces using Binary Decision Diagrams [17] or by application of abstractions and symmetries [18]. The above method should be more efficient than known methods because it does not exhibit state explosion. Standard automatic verification algorithms were observed to run in time and in space which increased polynomially in the size of the system while the number of reachable states increase exponentially. The presented algorithm is applicable because the distance computation problem can be solved in polynomial time by using low degree algebraic equations. A characteristic feature of the algorithm is the fact that it can be extended easily to the general case of a hybrid automaton. The drawback is the difficulty of finding the respective surface for every state of our physical system.

4 Application example

Hybrid automata represent the basic formalism for modeling and verifying the physical embedded systems. A hybrid automaton [19] is defined as $H = (Q, X, \Sigma, A, \text{Inv}, F)$ where

- Q is the set of locations. It includes also the initial location q_0 ,
- X is the continuous state space including the initial continuous state x_0 ,
- Σ is the set of events,
- A is the set of transitions which occur respectively the guards and the jump conditions
- Inv is an invariant that assigns a domain of the continuous state space to each location and
- F is the flow field which is assigned to each location.

We implement our algorithm in the physical embedded system that describes the control of a railroad crossing. The system consists of one process that combines the movement of the train and the controller of the crossing gate. We wish the algorithm to verify that two trains are not allowed to reach the gate simultaneously. It is supposed that each approaching train has a speed that varies between 48 and 52m/sec. When it approaches the gate it slows down to a speed between 40 and 52 m/sec. After crossing the gate and when it is 100 meters far away from the gate the second train can cross the gate. A safe distance between two trains is at least 1500 meters. A light is situated before the gate and it is green or red allowing the train to continue or not. The light is red or green for a known time period. In figure 3 a schematic diagram of the railroad crossing is given.

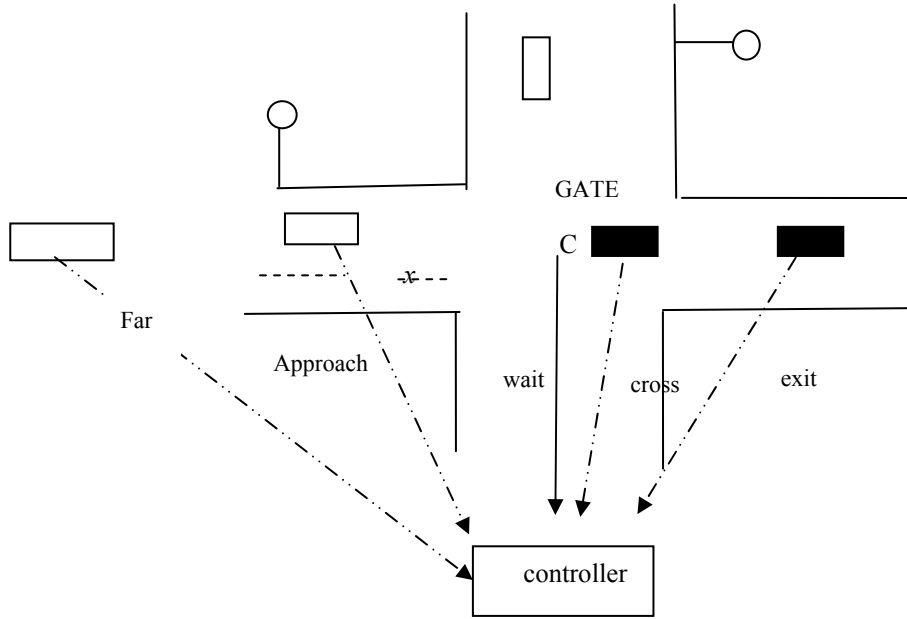


Fig. 3. Railroad crossing

The system is described by a hybrid automaton with six states. The states become from the combination of the different places that the train is located, far from the gate (F), approach (A), cross (C) and the state of the light, red (R) or green (G). Then the six states are labeled respectively FR, AR, CR, FG, AG, and CG. The automaton can never get in the state CR as the train cannot cross the gate while the light is red. Some of the transition between the states satisfies the following guards. The automaton that corresponds to the above system description is depicted in Figure 4, where x represents the distance from the gate, x' corresponds to the speed of the train, y is a clock for indicating the time that the light is green or red and G_1, G_2, G_{11}, G_{22} are the guards that have to be satisfied for enabling the transition and are described by the logic equations.

$$G_1 = (y > dr \ \& \ 0 \leq x \leq 1000 \ \& \ -52 \leq x' \leq -40)$$

$$G_2 = (y = dr \ \& \ x > 1000)$$

$$G_{11} = (y = dr \ \& \ x \leq 1000) \ \text{or} \ (y \leq dr \ \& \ 0 \leq x \leq 1000 \ \& \ 52 \leq x' \leq -40)$$

$$G_{22} = (y = dg \ \& \ 40 \leq x' \ \& \ x = 100) \ \text{or} \ (dg \leq y \ \& \ 40 \leq x' \leq 52 \ \& \ x \leq 100)$$

For the application of our algorithm we use the following functions that correspond to an ellipse and where x_1, x_2, x_3 corresponds to the variables of time and distance from the gate

$$f_1 = \frac{x_1^2}{48} + \frac{x_2^2}{52} + \frac{x_3^2}{1} - k_1 \quad (15)$$

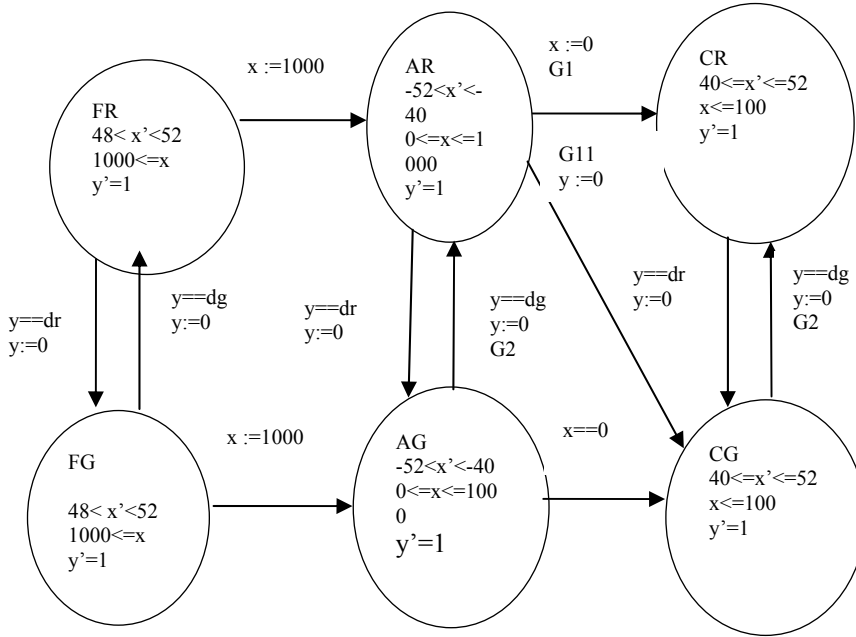


Fig. 4. Train automaton

$$f_2 = \frac{x_1^2}{52} + \frac{x_2^2}{40} + \frac{x_3^2}{1} - k_2 \quad (16)$$

The k_1 and k_2 are random constants. The time space that the light is in state red is chosen 3 sec and respectively the light is green for 2 sec. The described algorithm was implemented in the Matlab [20] environment and the condition that two trains cannot reach the gate simultaneously was verified. The initial state defines that the train is at a distance >1100 m from the gate and the light is red. The algorithm is initialized with the location that corresponds to the initial position of the train and the status of the light. The program consists of a main function and subroutines which are used in order to approximate the new point on states. The subroutine finds the new reached point, which is either in the same ellipse with the initial point or in another ellipse that corresponds to another state. In our implementation the initial state is assumed to be the state FR. Using the equations (15) and (16) the new reached point is in location AG and the $p_new = [900, 48, 0]$. The p_new is controlled by the main function if it satisfies the associated constraints, such as guards, invariants and then it continues to the next state or stay in the same. The next new point is situated in the same location AG. When the algorithm is terminated the elapsed time is about 29 sec.

The above system was also modeled and verified using the Polyhedral Hybrid Automaton Verifier, PHAVer, model-checking tool [9]. This is a tool for the safety verification of hybrid systems with piecewise-constant bounds on the derivatives. Safety verification reduces to the reachability problem, which is decidable only for a subclass of hybrid systems. PHAVer uses an over-approximating algorithm to ap-

proximate piecewise affine dynamics with linear hybrid automata. For the simulation with PHAVer the hybrid automaton is reformulated so that it can be analyzed using polyhedra [14]. In this tool the system is modeled by hybrid I/O automata with affine dynamics [14]. The over-approximating algorithm is based on the following principle. Consider a location S with invariant $\text{Inv}(S)$ and activity specified by the conjunction of linear expressions:

$$\bigwedge_{i=1}^m a_i^T \dot{x} + b_i^T x \propto_i c_i, \quad \propto_i \in (<, \leq) . \quad (17)$$

Each linear expression can be approximated by the following rule:

$$\forall i = 1, \dots, n \quad a_i^T \dot{x} \propto_i c_i - d_i, \quad d_i = \inf_{x \in \text{Inv}(S)} b_i^T x . \quad (18)$$

If the approximation is too coarse, the location is split in order to improve accuracy. Hence, the number of states grows rapidly. In figure 5 a part of the structure of Phaver code of the hybrid automaton train is depicted.

```

// Hybrid system traindr :=2;
automaton train
state_var: y, x;
//input_var:
synclabs: a0, a1, a2, a11, a22;
loc FR: while 1000 <= x wait {48 <= x' & x' <= 52
& y' == 1};
when y==dr sync all do{y'==0} goto FG;
when True sync a0 do{x'==1000} goto AR;
loc FG: while 1000 <= x wait {48 <= x' & x' <= 52
& y' == 1};
when y == 3 sync a22 do {y'==0} goto FR;
when True sync a0 do {x'==1000} goto AG;
loc AR: while 0<=x & x<=1000 wait {-52<=x' &
x'<=-40 & y'==1};
when y > dr & 0<=x & x<=1000 sync a1 do{x'==0}
goto CR;
when y == dr & x > 1000 sync all do {y'==0} goto
AG;
when y == dr & x <= 1000 sync a1 do {y'==0} goto
CG;
when y <= dr & 0 <= x & x <= 1000 sync a1 do
{y'==0} goto CG;
wait {-52<=x & x<=-40}

```

Fig. 5. PHAVER code

In contrast to the PHAVer approach, the algorithm presented above does not require a specific formalism, and by using line geometry is able to verify embedded systems without searching in an exponentially grown number of discrete states. In the

described algorithm simple equations are solved analytically in order to verify that a desired state is reachable. The reachable set that is computed by PHAVer is illustrated in figure 6. It shows the distance from the gate and the time that the light is red or green and information is not given for the location. The distance and the time correspond to the reachable states. The elapsed time in PHAVer is estimated more than our algorithm. The order of variables is the same as provided by the automaton output. The reachable is in the region of the 200 meters. It is not possible to extract the sequence of the states that the hybrid model passes until the desired state is reachable. Instead part of the sequence of the states that the model passes through using the presented algorithm is traced down: $FR \rightarrow AG \rightarrow AG \rightarrow CG \rightarrow FG$.

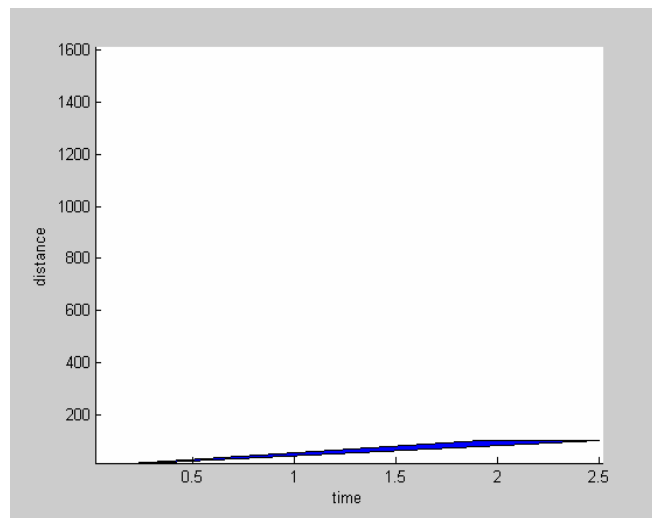


Fig. 6. Reachable set

5 Conclusion

All developed techniques for checking the satisfaction of safety requirements in embedded systems by using hybrid automata models rely on the efficiency of the algorithm used to compute reachable sets of the hybrid automaton from an initial state. They differ mainly in the assumptions made about the way we formally describe the sets, and the way the continuous state dynamics evolve. A new method to approach the verification of safety properties of embedded systems was presented in this article. In the context of this algorithm, a number of algebraic calculations are realized in order to determine whether a state of the hybrid automaton model, which can be associated with the specifications of the safety operation of the system, exists in the reachability region of the automaton. Using the distance computation with line geometry it is possible to formulate the verification problem as the trace of two simple problems.

One is the existence of the solution of a polynomial and the second is the control of the solution if it belongs to a prescribed set. The technique presented in this paper can be applied to any surface that describes well the location of the hybrid system. Accordingly the surface that we use, the degree of the used equations is reduced or is augmented. The developed model checking algorithms, as PHAVer, need to perform expensive polyhedral computation, and their algorithm complexity is exponential in number of variables in the automata. The existing techniques cannot check all the paths for reachability analysis on account of problem size. From the case study of the control of a railroad crossing, it is demonstrated that our algorithm does not affront the problem of state explosion. In contrast we obtain a result in small time. Our algorithm appears to be implemented in complicated case where we manipulate multidimensional surface. In this case the equation (11) it is more complex and it is not six degree. The system can consist of a lot of variables that are changing during the time. The computational efficiency of this algorithm is due to the substitution of the transition to location by the distance from this location. The algorithm can be written in every programming language. Thanks to the rich command language, one can split the locations by defining splitting surfaces. This is important in order to accomplish the verification process and reach a useful answer in a short time. In this work it was demonstrated that using the formal method of hybrid automata and the distance computation between two given surfaces it is possible to verify the control of a railroad crossing model.

References

1. Computer Science and Telecommunications Board. Embedded, Everywhere: A Research Agenda for Networked Systems of Embedded Computers, (2001)
2. Alur, R., Courcoubetis, C., Henzinger, T., P-H.Ho.: Hybrid automata: an algorithmic approach to the specification and verification of hybrid systems. In Hybrid Systems I, pp 209--229, Lecture Notes in Computer Science 736, Springer-Verlag, (1993)
3. Tomlin, C., Mitchell, I., Bayen, M., Oishi M., A.: Computational Techniques for the Verification of Hybrid Systems, In: Proceedings of the IEEE, Vol 91, No 7, (July 2003)
4. Daws, C., Olivero, A., Tripakis, S., Yovine, S.: "The tool KRONOS," in Hybrid Systems III. New York: Springer-Verlag, vol. 1066, Lecture Notes in Computer Science, pp. 208--219, (1996)
5. Henzinger, T.A., Ho, P.H., Wong-Toi, H.: Hytech: A model checker for hybrid systems, International Journal on Software Tools for Technology, Transfer 1, pp.110 --122, (1997)
6. Kim, G., Larsen, Pettersson, P., Wang Yi.: Model - Checking for Real-Time Systems, (1993)
7. Asarin, E., Dang, T., Maler, O.: The d/dt Tool for Verification of Hybrid Systems, In Proceedings of Computer Aided Verification: 14th International Conference, CAV 2002, LNCS, vol 2404, pp.7460, Springer Heidelberg, (2002)
8. Lee, E.A.: Hyvisual: A hybrid system Modeling Framework based on Ptolemy II, Elsevier Science, (2006)
9. Frehse, G.: Phaver. Software package, <http://www.andrew.cmu.edu/~gfhrehse>. (2005)
10. Girard, A., Pappas, G.J: Approximation metrics for discrete and continuous systems, MS-CIS-05-10, Dept. Computer and Information Science, University of Pennsylvania, Philadelphia, USA, (May 2005)
11. Lennerz, C., Schomer E.: Efficient distance computation for quadratic curves and surfaces. In: Proceeding of geometric modeling and processing, pp. 60--90, (2002)

12. Xiao-Diao Chena, Yong, J.H., Guo-Qin Zheng, Jean-Claude Paul, Jia-Guang Sun.: Computing minimum distance between two implicit algebraic surfaces, *Computer Aided Design* vol.38, pp.1053--1061, (2006)
13. Sohn, K-A., Juttler, B., Kim, M-S., Wang, W.: Computing distances between surfaces using line geometry In: *Pacific conference on computer graphics and applications*, pp. 236--245, (2002)
14. Frehse, G.: Phaver: Algorithmic verification of hybrid systems past Hytech. In: *HSCC*, LNCS 3414. Springer, pp. 258--273, (2005)
15. Kawachi, K., Suzuki, H.: Distance computation between non-convex polyhedra at short range based on discrete Voronoi regions. *Proc. of Geometric Modeling and Processing*, pp. 123--128, Hong Kong, April 10-12, (2000)
16. Seong, J.K., Kim, M.S., Sugihara, K.: The Minkowski sum of two simple surfaces generated by slope-monotone closed curves. *Proc. of Geometric Modeling and Processing*, pp. 33-42, Saitama, Japan, July 10-12, (2002)
17. Burch, J. R., Clarke, E. M., McMillan, K. L., Dill, D. L., Hwang, L. J.: *Symbolic Model Checking: 10²⁰ states and beyond*, *Logic in Computer Science*, (1990)
18. Emerson, E. A., Jutla, C. S.: *Symmetry and Model Checking*, *Lecture Notes in Computer Science*, 697. In *Proceedings of CAV'93*, (1993)
19. Alur, R., Courcoubetis, C., Halbwachs, N., Henzinger, T. A., Ho, P. H., Nicollin, X., Olivero, A., Sifakis, J., Yovine, S.: *The algorithmic analysis of hybrid systems*. vol.138, pp. 3-34, *Theoretical Computer Science*, (1995)
20. Matlab The Language of Technical Computing, <http://www.mathworks.com>