Towards the Safety Verification of Real-Time Systems with the Coq Proof Assistant

Olga Tveretina

Abstract:

Hybrid systems involve the interaction of discrete and continuous dynamics. Hybrid systems have been used as a mathematical model for many safety critical applications. One of the most important analysis problems of hybrid systems is the reachability problem. In this paper we argue that the proof assistant Coq can be used for the hybrid systems verification. An example of a train crossing control is provided.

Keywords: *Formal methods, real-time, theorem proving.*

1. Introduction

Real-time embedded systems have become very important in our everyday life. Programs such as device drivers and embedded controllers must run on real-time constraints. Demand placed on the embedded systems functionality, complexity and critical nature are increasing.

Hybrid systems are systems where is a significant interaction between the continuous and discrete parts and high performance specifications are to be met by the system. For example, hybrid systems arise from the interaction of discrete planning algorithms and continuous processes.

Many of the hybrid systems applications are critical safe and require the guarantee of safety operation. The problem of safety verification seeks for an answer to the question: is there a potentially unsafe state reachable from an initial state? Therefore, formal verifying safety properties of a hybrid system consist of building a set of reachable states and checking whether this set intersects with a set of unsafe sets. Therefore, one of the most central problems in the analysis of hybrid automata is the problem. It checks whether all trajectories of a given hybrid system meet a given safety requirement. For systems with continuous dynamics it is very difficult to compute the set of all states reachable from an initial set.

Abstraction is one of the complexity reduction techniques. It reduces the state space of a system by mapping it to an abstract set of states that preserve the actual behaviour of the system. A predicate abstraction approach for the verification of hybrid systems is represented in [2]. However, the computational cost of predicate abstraction can be too high, and for large systems practically infeasible. That is why we start from a method that decomposes the state space of a system according a rectangular grid [6]. We show applicability of our method with the verification example that we have formalized within the proof assistant. It models the gate controller of a railroad crossing.

2. Problem description

We use a hybrid automaton as a mathematical formula for describing hybrid systems. We formally define the notion of hybrid automaton. The notion of a hybrid automaton was introduced in order to extend verification methods towards the systems with continuous and discrete dynamics [1]. For simplicity, we have assumed that the number of discrete locations is finite and the vector field is Lipschitz continuous. It guarantees that the solutions of the differential equations are well defined.

Definition. A Hybrid automaton is a tuple $HS = (DS, n, S_0, Inv, F, Guard,$ and $Reset)$ with the following components:

- DS is a finite set of discrete locations; $n>0$ is the dimension HS. The state space of HS is $S = DS \times R^n$. Each state has thus the form (l,x), where $l \in DS$ and $x \in R^n$. $S_0 \in S$ is a set of initial states.
- $Inv: DS \rightarrow P(R^n)$ assigns to each discrete location l an invariant set, which constrains the value of the continuous part while the discrete part is l, i.e. continuous evaluation can go on as long as x remains in $Inv(l)$.
- $Guard: DS \times DS \rightarrow R^n$ describes a guard condition, i.e. if a system remains in a discrete location l_1 and a continuous state x reaches the guard $Guard(l_1,l_2)$ then the discrete state may change its value to l_2.
- $F: DS \times R^n \rightarrow R^n$ is a vector field.
- $Reset: DS \times DS \times R^n \rightarrow R^n$ is a reset function.

Definition. (Admissible function) We say that a function $f: R^n \rightarrow R^n$ is admissible if there are m_1 and m_2 such that for all $x_1, x_2, x_3 \in R^n$ such that $x_1 = m_1 x_2 + (1\text{-}m_1) x_3$ for some $0 < m_1 < 1$ there is $0 \leq m_2 \leq 1$ such that $f(x_1) = m_2 f(x_2) + (1\text{-}m_2) f(x_3)$.

In the following we assume that functions defining continuous behaviour satisfy this property. It will be used for the proving of the correctness of the approach.

3. Transition system

A system state can change in two ways: either continuously by time evaluation or by discrete transitions. Hence, there are two kinds of transitions. First one is a continuous transition, describing the evolution of a system in a given location. Second one is a discrete transition, describing movement from one location to another location and, possibly, changing the continuous variables according to the function Reset. Based on this, the semantics of a hybrid automaton is given by the following transition system [2].

Definition. (Transition System) Given a hybrid automaton $HS = (DS, n, S_0, Inv, F, Guard, Reset)$, the transition system $Tr = \{S, \rightarrow_c, \rightarrow_d, S_0\}$ of HS consists of

- the set of states S;
- the set of initial states S_0;
- two relations $\rightarrow_c$ and $\rightarrow_d$ satisfying the following

$$(l,x) \rightarrow_c (l,y) \Leftrightarrow \exists t \geq 0, \forall i, F_i(l,x,t) = y_i \wedge y \in Inv(l),$$

$$where\, x = (x_1, \ldots, x_n), y = (y_1, \ldots, y_n)$$

$$(l,x) \rightarrow_d (l', y) \Leftrightarrow (l,x) \in Guard(l, l') \wedge y = Reset(l,l',x)$$

Definition. (Trajectory) Given a hybrid automaton HS, a trajectory of HS is a sequence $s_1, s_2, \ldots, s_m$, such that for all i and j such that $i>j$, $s_j \rightarrow_d s_j$ or $s_j \rightarrow_c s_j$.

Given a hybrid automaton HS and a set of unsafe states U, the safety verification problem concerns with proving that all trajectories of HS cannot enter U.

4. Reachability analysis

Our method is based on the approach that decomposes the continuous state space according to a n-dimensional rectangular grid. Such abstractions are mostly performed in a manual manner. In general, a state space can be represented by polyhedra [2]. This is a more flexible approach but it requires an algorithm for dealing with these polyhedra. A rectangular grid is less flexible but it is simpler to implement the corresponding operations.

We assume that we have an algorithm that can produce a decomposition of a state space. We denote by S^a a set of all abstract states. In order to define a discrete abstraction of a given system, we have to describe the transitions between the abstract states, the set of initial abstract states S^a_0, and the set of abstract unsafe states U^a. Given a hybrid automaton HS and a set of abstract states, the set of initial abstract states can be computed. An artificial transitivity can create many false counter examples, i.e. abstract behaviours that do not correspond to any concrete ones. That is our incentive to optimise an abstract transition system from [2] by reformulating the relation defining an abstract continuous step.

Definition. (Abstract Transition System) Given a hybrid automaton $HS = (DS, n, S_0, Inv, F, Guard, Reset)$, the abstract transition system $Tr^a = \{S^a, \rightarrow^a_c, \rightarrow^a_d, S^a_0\}$ of HS consists of

- the set of abstract states S^a;
- the set of initial abstract states S^a_0;
- two relations $\rightarrow^a_c$ and $\rightarrow^a_d$ satisfying the following

$$(l,x^a) \rightarrow^a_c (l,y^a) \Leftrightarrow x \in x^a, y \in y^a : (l,x) \rightarrow_c (l,y)$$

$$(l,x^a) \rightarrow^a_d (l', y^a) \Leftrightarrow x \in x^a, y \in y^a : (l,x) \rightarrow_d (l', y)$$

Definition. (Abstract Trajectory) Given a hybrid automaton HS and a set of abstract states S^a, an abstract trajectory of HS is a sequence $s^a_1, s^a_2, \ldots, s^a_m$, such that for all i and j such that $i>j$, $s^a_j \rightarrow_d s^a_j$ or $s^a_j \rightarrow_c s^a_j$.

Note, that in general, an abstract trajectory starting from some initial abstract state is not unique.

5. Modelling of hybrid systems with the proof assistant Coq

Coq is an interactive proof assistant, which allows formal defining mathematical objects and helps the user with proving the properties of these objects. Basically, the use of Coq follows three steps: a) define the objects and axioms used, b) state a theorem, c) provide proof steps until the proof is completed. For more details we refer to [3].

We do not have enough space to present all formalizations in Coq that is why we present only a small part of it.

Definition DiscStates:=Set.
An abstract state is defined as a list of natural numbers.
Definition AbsState := list nat.
A partition of an abstract state is defined as a list of lists of naturals.
Definition Partition:= list (list nat).
Some other ingredients are defined as follows.
Parameter Invariant: DiscStates → AbsStates.
Parameter Guard: DiscStates → DiscStates → AbsStates.
Parameter Flow: DiscStates → AbsStates → AbsStates.
Parameter Reset: DiscStates → DiscStates → AbsStates → AbsStates.

As an example we consider, the gate controller of a railroad crossing from [4]. The model consists of two subsystems: a train and the gate controller. The train is required to send a signal *app* at least two minutes before it enters the crossing. The train sends a signal *out* when it leaves the crossing and it must happen no afterwards than 5 minutes after the *app* signal (this is expressed by the guard $2< x <5$). The gate must be closed in at least 1 minute after the *app* signal is received and not later than in 2 minutes (the guard is $1< y <5$). The gate responds by opening within 1 minute after receiving the *out* signal. We have considered a product of timed automata that combines the train process and the controller process, and the resulting system is depicted in Figure 2. It has the following discrete locations: Loc1: a train is far from the gate and the gate is open; Loc2: the train is approaching the crossing and the gate is open; Loc3: the train is approaching the crossing and the gate is closed; Loc4: the train has left the crossing and the gate is closed. We want to verify that the gate is never closed longer than 5 minutes. There are two continuous variables. Therefore, the dimension of the continuous state space is 2.

Definition Dim := 2.
There are four discrete states. That is modelled in Coq as following.
Inductive DS: Set := | D1: DS | D2: DS | D3: DS | D4: DS.
The partition of the continuous state space is:
Definition P := (0::1::2::3::4::5::6)::(0::1::2::3::4::5::6).

We were able to 'model check' that unsafe states are not reachable. The full formalization of the verification approach includes also a proof of the correctness of the procedure in Coq that will use a definition of an admissible function. Due to the property described in the definition, a segment of a straight line is mapped to a segment

of a straight line. This means that for each continuous state that is in some abstract state there is a corresponding abstract transition. This proof is left for future work.

6. Conclusions and future research

We have made a first step towards the fully formal verification of hybrid systems in the proof assistant Coq. The presented framework allows combination of discrete abstraction with other approaches, as "barrier certificates" for example. The train-crossing example was "model checked" in Coq. As for future work, we intend to investigate the structure of special classes, for example linear hybrid systems, for which our method is efficient.

ACKNOWLEDGMENTS

I am grateful for the support by Milad Niqui in the field of Coq, and for the support by Herman Geuvers and Dan Synek for the examples of hybrid systems formalizations in Coq. The research is supported by the BRICKS/FOCUS project 642.000.501.

AUTHOR

Olga Tveretina - Institute for Computing and Information Sciences, Radboud University Nijmegen, The Netherlands. E-mail: o.tveretina@cs.ru.nl.

References

[1] R. Alur, C. Courcoubetis, N. Halbwachs, T. A. Henzinger, Pei-Hsin Ho, Xavier Nicollin, A. Olivero, J. Sifakis, S. Yovine, “The algorithmic analysis of hybrid sys-tems”, *Theoretical Computer Science*, vol. 138, 1995, pp. 3-34.

[2] R. Alur, Th.Dang, F.Ivancic, “Reachability Analysis of Hybrid Systems via Predicate Abstraction”, *ACM transactions on embedded computing systems (TECS)*, 2004.

[3] Y. Bertot, P. Casteran, “Interactive Theorem proving and Program Development”, Springer, 1998.

[4] T.A. Henzinger, X. Nicollin, J. Sifakis, S.Yovine, “Symbolic Model Checking for Real-time Systems", 7th *Symposium of Logics in Computer Science*, 1992.

[5] A. Henzinger, P.W. Kopke, A. Puri, P. Varaiya, “What's Decidable About Hybrid Automata?”, *Journal of Computer and System Sciences*, vol. 57, no. 1, 1998, pp. 94-124.

[6] S. Ratschan, Z. She, “Safety verification of hybrid systems by constraint propagation-based abstraction refinement”, *ACM Transactions on Embedded Computing Systems (TECS)*, vol. 6(1), 2007.