Theoretical Computer Science 642 (2016) 1–23

Contents lists available at ScienceDirect

Theoretical Computer Science

www.elsevier.com/locate/tcs

Deadness and how to disprove liveness in hybrid dynamical systems

Eva M. Navarro-López *, Rebekah Carter

School of Computer Science, The University of Manchester, Oxford Road, Kilburn Building, Manchester, M13 9PL, UK

ARTICLE INFO

Article history:
Received 13 January 2015
Received in revised form 15 March 2016
Accepted 10 June 2016
Available online 16 June 2016
Communicated by P. Aziz Abdulla

Keywords:
Hybrid systems
Liveness
Stability analysis
Discontinuous systems
Hybrid automata

ABSTRACT

What if we designed a tool to automatically prove the dynamical properties of systems for which analytic proof is difficult or impossible to obtain? Such a tool would represent a significant advance in the understanding of complex dynamical systems with nonlinearities. This is precisely what this paper offers: a solution to the problem of automatically proving some dynamic stability properties of complex systems with multiple discontinuities and modes of operation modelled as hybrid dynamical systems. For this purpose, we propose a reinterpretation of some stability properties from a computational viewpoint, chiefly by using the computer science concepts of safety and liveness. However, these concepts need to be redefined within the framework of hybrid dynamical systems. In computer science terms, here, we consider the problem of automatically disproving the liveness properties of nonlinear hybrid dynamical systems. For this purpose, we define a new property, which we call *deadness*. This is a dynamically-aware property of a hybrid system which, if true, disproves the liveness property *by means of a finite execution*. We formally define this property, and give an algorithm which can derive deadness properties automatically for a type of liveness property called inevitability. We show how this algorithm works for three different examples that represent three classes of hybrid systems with complex behaviours.

1. Introduction

A hybrid dynamical system is a mathematical model for a part of the real world where discrete and continuous parts interact with each other. Such systems can model all kinds of situations, from biological systems [21,31] to a controller interacting with its environment [67,70], from electronic circuits [38,53] to mechanical systems [51].

Suppose we have a hybrid system, which is modelled in a particular framework. We might try analysing this model mathematically to work out what would happen in the system, using Lyapunov stability theory [15,23,33,48,54]. However, it is typically unclear what the behaviour patterns of a general hybrid system are, due to the interactions of the continuous dynamics and the discrete transitions. So although we can define mathematically what we would like to be true, it is not necessarily possible to decide whether this is actually true.

As a second choice, maybe we consider simulation as a method for discovering what the behaviour patterns of the system are. With this method we can get a very clear idea of what might happen for particular start states, but it is hard to know that we have covered all possible behaviour patterns of a system, no matter how many simulations we run.

* Corresponding author. Tel.: +44 161 27 56209; fax: +44 161 27 56204.
E-mail addresses: eva.navarro@manchester.ac.uk (E.M. Navarro-López), racarter174@gmail.com (R. Carter).

http://dx.doi.org/10.1016/j.tcs.2016.06.009

This leads to a third option of analysing the system: formal verification. This involves analysing the hybrid system using automatic methods to see if it conforms to some desired pattern. The verification process involves specifying the model of the system, specifying the property we wish it to conform to, and then checking if the model satisfies the desired property. This checking phase can be achieved with either logical deduction (deductive verification or theorem proving) [2,58], or an exhaustive search of the states in the system [19,63].

Whichever method is used to find the relationship between the system and the property, we must first have a specification of the system itself, which can be framed in many terms; we use the *hybrid automaton* framework [4,42,51]. With this background model, we then specify the patterns we wish the system to obey, which are usually framed as *logical properties* in some defined logic, although other approaches have been proposed [17]. Some possible logics include linear temporal logic (LTL) [59], computation tree logic (CTL) [27], and the class of various metric temporal logics [5,44].

In hybrid dynamical systems, the vast majority of logical properties studied have been *invariance properties*, which say that a system will always satisfy some logical expression for all time [20,36,63]. These invariance properties are a subset of the more general class of *safety* properties, which have an informal definition which says that a 'bad thing' never happens, and is the type of properties that have attracted more attention in the field of hybrid systems, together with model checking techniques [65]. It is interesting to highlight the application of model checking techniques to automatically generate switching controllers for linear hybrid automata [12,13]. The approach proposed in our paper has not been designed for controller synthesis. However, it could be used to evaluate the performance of closed-loop systems by automatically checking the satisfaction of some dynamical properties after a controller has been applied to the system. We stress the fact that we deal with nonlinear hybrid automata. This entails more difficulties than for linear hybrid automata for the automatic generation of controllers. Verification techniques for nonlinear systems are not as common as for linear systems. A recent tool for the verification of properties (model validation and parameter synthesis) for hybrid systems with nonlinear dynamics – in particular, piecewise nonlinear systems – is the Breach tool [25]. We also highlight the verification for nonlinear hybrid automata using barrier certificates [62].

The complementary property to safety is *liveness*, which has been barely touched as a formal verification problem for continuous-time systems. Some kind of liveness is implicit in various dynamical properties such as strictly decreasing Lyapunov functions, periodic orbits, and proving that solutions of differential equations exist, however explicit statement and computational proof of liveness properties has only happened in certain classes of hybrid systems. These classes include linear hybrid automata [6], where the dynamics are always solvable, and also systems for which piecewise constant bounds on the derivatives can be given [39], where approximations of the reachable space are easy to find. There is also recent work on proving sub-classes of liveness in continuous and hybrid systems: inevitability properties [18,26,66] and region stability [49,60,61]. Inevitability says that 'eventually a certain region of the state-space is reached', and is the simplest of the liveness properties both for specification and proof. Liveness has also been considered in examples, for instance [43] looks at safety and liveness for a controller for an automatic intersection, and [28] proves liveness properties showing that robots move in a certain way.

There are also academics who consider time-bounded liveness properties instead of liveness properties on infinite time, and these include [11,32]. Such bounded liveness properties are effectively safety properties in the way they are proved, so are generally considered less complex. However, there are some essential properties of systems which cannot be transformed into a time-bounded form, like infinite recurrence of states for instance. We should also distinguish liveness from the similarly-named concept of livelock which comes from the theory of parallel processes. Livelock and deadlock are both notions of two (or more) processes interfering with each other to stop useful motion occurring. Deadlock is where the two systems block each other from any further motion, whereas livelock is when the states of the systems keep changing, but the desired thing never happens. In this sense, both deadlock and livelock are like counter-examples to liveness, as they prove that the desired thing does not happen. In the world of parallel programs, tight definitions of deadlock and livelock are given by [69], and in hybrid control systems definitions are given by [1].

In this paper, we consider (unbounded) safety and liveness properties that can be written in a logic such as linear temporal logic (LTL), and will give a formal definition for these properties on hybrid automata, using the ideas of [3] who defined what a safety or liveness property is in concurrent programs. In dynamical systems, liveness properties can characterise many useful properties, as liveness says that something eventually becomes true. For instance, liveness can characterise the idea of convergence to some desired state, a key part of Lyapunov stability theory. The property of always returning to some desired state can also be characterised by a liveness property. In this paper we relate liveness properties to some stability-related properties of hybrid dynamical systems, mainly, global attractivity. Attractivity encompasses the idea that a system trajectory will keep getting closer and closer to a desired point in space. Stability and attractivity capture the intuition that we have about good behaviour of a dynamical system, but are not easy to prove automatically. We highlight that the stability-like properties covered by our approach are conceptually different than practical stability. Practical stability was originally proposed to deal with practical control problems, mainly, the study of finite time control and stability [34, 73,74]. It is related to convergence to behaviours with pre-specified bounds during a fixed time interval. It is similar to uniform boundedness, but in contrast to uniform boundedness, in practical stability the bounds in the state space and time are pre-defined [45].

Liveness properties are characterised by the idea that, at any finite point in an execution, they could always be satisfied at some point in the future, or alternatively that the only type of execution which can disprove such a property is one of infinite length. In continuous space–time it can be very difficult to be able to find infinite length paths. If we could gain