The Concept of Deadlock and Livelock in Hybrid Control Systems

Alessandro Abate[1], Alessandro D'Innocenzo[2], Giordano Pola[2,3], Maria Domenica Di Benedetto[2], and Shankar Sastry[1]

[1] Department of Electrical Engineering and Computer Sciences, University of California, at Berkeley - Berkeley, USA
{aabate,sastry}@eecs.berkeley.edu
[2] Department of Electrical Engineering and Computer Science, Center of Excellence DEWS, University of L'Aquila - L'Aquila, Italy
{adinnoce,pola,dibenede}@ing.univaq.it
[3] Department of Electrical Engineering, University of California, at Los Angeles - Los Angeles, USA
pola@ee.ucla.edu

Abstract. This short paper qualitatively introduces the definition of the concepts of Deadlock and Livelock for a general class of Hybrid Control Systems (HCS). Such a characterization hinges on three important aspects: firstly, the concept of composition of HCS; secondly, the general concept of specifications and their composition for HCS; finally, the dynamical structure and behaviors of HCS. The first aspect is introduced in a novel manner, including ideas from the literature of discrete transition systems and accounting for concepts such as that of dynamical feedback interconnection. The second point includes general properties that are of interest from a systems and control theory perspective. The third part categorizes the diverse and possibly pathological behaviors that are distinctive of HCS. A first look at the problem of Deadlock and Livelock Verification concludes the manuscript.

1 Introduction

The concept of deadlock and its close relative, that of livelock, have been widely investigated in the literature of various branches of computer science. Deadlock, in particular, has often been regarded as a pathology and associated with the deficiency of a liveness specification, that of forward progress [6]. Much interesting work has been focused on verifying the presence of deadlock situations in algorithms or programs, or on ensuring its absence upon their composition [3][4].

Hybrid Systems are rather general mathematical models that connect between discrete, logical, synchronous systems and continuous, real-time, asynchronous ones [2]. It has often been observed that they present behaviors or are endowed with properties that are "at the limit" between classical transition systems and dynamical models [2].

Motivated by a number of case studies, this work aims at "exporting" the notions of deadlock and livelock to the Hybrid Control Systems (HCS) case. More precisely, the objective has been that of first introducing a mathematically rigorous definition of the phenomena and providing a clear characterization of them. We stress that the introduced concepts naturally tailor back to the corresponding ones in the literature of, respectively, discrete and continuous systems.

2 Deterministic Hybrid Control Systems

The model for HCS is a melange between the classic hybrid automaton [2] and the HIOA [4]. In particular, it adheres to a *denotational* definition at the internal, state-space level, while it is inspired by an *operational* characterization at the external, input/output level. More precisely, an HCS is characterized by a finite collection of modes, each of which is associated with a domain and a control-dependent vector field. The set of transition relations is composed of a collection of edges (ordered pairs of modes), guards (subsets of the domains, possibly control-dependent), and deterministic reset functions. The control space, which contains real time-dependent control functions, is assumed to be bounded. Finally, the HCS is endowed with an observation space: the output functions will be obtained from the hybrid executions via a static output map. The set of initial conditions is a subset of the hybrid state space. To introduce the concept of executions of the HCS, it is first necessary to define the *hybrid time set*, a rather classical notion in the literature, as an ordered sequence of time intervals that represent the "dwelling times" of the continuous evolution within a mode. The *hybrid execution* is a hybrid trajectory (a pair of discrete and continuous evolutions of the flow) which is defined on the hybrid time set and abides by the flowing and switching within a HCS and is thus characteristic of its internal structure. It is possible to raise some rather general assumptions to enforce the determinism of the model.

The output of the hybrid system is, for each execution, a function from the hybrid time set to the output space. Since our purpose is to set up a notion of input-output interconnection, in the spirit of [4], we suppose that the *interconnectible* output of hybrid systems considered is instead a set of physical signals, function of the real time, obtained by a simple operation on the output of the HCS. This assumption is motivated by the need to give an asynchronous notion of interconnection.

3 Hybrid Systems Composition

Abstractly, the concept of systems *composition* may be introduced in many ways, depending on the characteristics and properties of the systems that are considered, the structure of the operation, and the particular properties that we may want to check for. In this work we consider an operation that may be interpreted as a form of *parallel composition.* Unlike previous work though, which simply performed parallel compositions as crude variables "sharing", inspired here by a more control theoretical perspective we allow the connections between inputs and outputs of the systems to depend on general functions endowed with some properties. Doing so, we naturally introduce an *output feedback* framework. Notice that the introduction of a model structure with internal and external components, similar to that in [4], allows to conceive the system at the level of its hidden/internal variables (the hybrid state space with its vector fields and transition relations) as a black box and only focus on the external components when performing the interconnection.

Proper "compatibility" conditions on two general HCS need to be raised before composing them. The actual HCS, result of the composition, is defined as follows: the "internal" structure of the composed system is basically the cartesian product of the two original hybrid automata. Two interconnecting static maps turn a transformation of the original output space of one of the two systems into part of the original input space of the other system, and vice versa. The new output space is simply the cartesian product of the original two, while the input space of the composition is, intuitively, the set of "unused inputs" of the composition. In the extreme case, the composition may be purely dynamical.

Asynchronism is preserved in the composition. The semantics of the composed model allow to not care about the presence of "cyclic constraints". The composition does not exclude the presence of pathological events (Zeno or blocking, for instance), which arises at an internal level. A rather slack condition on the continuity of the interconnecting maps allows to preserve determinism in the composition. Furthermore, the commutativity and associativity properties hold.

4 Composing Hybrid Systems Specifications

In this section we consider rather general specifications defined on hybrid trajectories in the observation space. They may be defined, for instance, via temporal logic formulae for real-time systems. Furthermore, we shall also introduce an explicit dependence on the control signals: this would allow to express specifications that are general enough to cover the most important problems in control theory. Instances of such specifications are that of *reachability*, *invariance*, *viability*, *attractivity.* Safety, liveness and forward progress can be reinterpreted through the above properties, as well as verification and control synthesis tasks.

We look for the set of trajectories, that is the behaviors, that verify a particular specification. Because of the deterministic hypothesis for the model, it is possible to associate this set of trajectories to a certain collection of initial conditions.

Given two HCS, two corresponding specifications and a composition procedure, the composed specification is defined as the *conjunction* of the two original specifications, modulo proper *variables substitutions* according to the interconnection maps associated with the composition procedure.

Consider the cartesian product of the sets of initial conditions of the single systems associated with trajectories that verify the corresponding property. Within this set, it is particularly interesting to look at the set of initial conditions in the composed system, that originates trajectories that do not verify the composed specification. These initial conditions are associated to "pathological" executions. It is indeed among the trajectories in this set that we shall categorize those associated with deadlock and livelock.

5 Definition of Deadlock and Livelock for Hybrid Control Systems

From a dynamical standpoint, the concepts of deadlock and livelock are intrinsically related to the idea of a trajectory being "constrained" or "stalled" somewhere in the state space. This locking condition is then further specified with regards to the presence or absence of indefinite motion within the region.

The fundamental concepts of this paper are then qualitatively introduced as follows. The "pathological" trajectories singled out above can be of two kinds: those that end up in a *hybrid invariant set*, and those that do not. The executions that do enter in an invariant set are either *deadlock* or *livelock*: the first are characterized by the *absence of motion* in finite time ("stalling" situations). The second are instead characterized by *endless motion*, either in their continuous or discrete component.

Notice that the definition above hinges on a purely *dynamical* level. This represents the last point, after that of *composition* and that of *specification*, which is regarded as necessary to introduce the notions of deadlock and livelock in the framework of HCS. Special instances of the above behaviors that are "notorious" for HCS are, in the case of deadlock situations, blocking conditions, stable equilibria in finite time, chattering and genuine Zeno. For the case of livelock, examples are represented by stable equilibria in infinite time and limit cycles.

6 Conclusions and Future Work

This extended abstract only qualitatively introduces the concept of deadlock and livelock for HCS. A number of fundamental details have been skipped for the sake of space. Also, interesting interpretations of the above concepts in a number of application instances have not been reported in this work. An extended and detailed manuscript can be found in the form of a technical report [1].

From the above discussions, it comes at no surprise that the next obligatory step after the definition and characterization of the notion of deadlock and livelock for HCS is that of looking at ways to *detect* it. Deadlock and Livelock *prevention* and *resolution* are other topics that do not find space in the present paper. The authors are also working on other extensions of the presented results. The concept of composition is prone to be generalized, and the issue of "deep composition", i.e. of a composition procedure preserving certain properties through its structure, clearly connects with the above ideas when the absence of deadlock or livelock is the specification to be exported.

References

1. Alessandro Abate, Alessandro D'Innocenzo, Giordano Pola, Maria Domenica Di Benedetto, Shankar Sastry : *The Concept of Deadlock and Livelock in Hybrid Control Systems.* Technical Report, UCB/EECS-2006-181, `http://www.eecs.berkeley.edu/Pubs`, Dec 2006.
2. John Lygeros, Karl Henrik Johansson, Slobodan N. Simic, Jun Zhang, Shankar Sastry : *Dynamical Properties of Hybrid Automata.* IEEE Transactions on Automatic Control, vol. 48, no. 1, Jan 2003.
3. Rajeev Alur, Thomas Henzinger: *Modularity for Timed and Hybrid Systems.* Proceedings of the 8th International Conference on Concurrency Theory (CONCUR 97), LNCS 1243, pp. 74-88, 1997.
4. Nancy Lynch, Roberto Segala, Frits Vaandrager: *Hybrid I/O Automata.* Information and Computation, 185(1):105-157, 2003.
5. Martin Abadi, Leslie Lamport: *Composing Specifications.* REX Workshop on Stepwise Refinement of Distributed Systems, Mook, NL, May 1989.
6. Bowen Alpern and Fred Schneider: *Defining Liveness.* Information Processing Letters, vol. 21, pp. 181-185, 1985.