Available online at www.sciencedirect.com
SCIENCE DIRECT
ELSEVIER
Control Engineering Practice 14 (2006) 1259–1267
CONTROL ENGINEERING PRACTICE
www.elsevier.com/locate/conengprac

ACTL strong negation and its application to hybrid systems verification ☆

Zhi Han[a,*], Alongkrit Chutinan[b], Bruce H. Krogh[a]

[a]*Department of Electrical and Computer Engineering, Carnegie Mellon University, Pittsburgh, PA, USA*
[b]*The MathWorks, Inc., 3 Apple Hill Dr, Natick, MA 01760-2098, USA*

Available online 20 March 2006

Abstract

Model checking procedures for verifying properties of hybrid dynamic systems are based on the construction of finite-state abstractions. If the property is not satisfied by the abstraction, the verification is inconclusive and the abstraction needs to be refined so that a less conservative model can be checked. If the hybrid system does not satisfy the property, this verify–refine procedure usually will not terminate. This paper introduces the concept of strong negation for ACTL formulas as an auxiliary condition that can be verified to obtain a conclusive negative verification result from a finite-state abstraction in certain cases. The concepts are illustrated with an example from automotive powertrain control.

Keywords: Verification; Hybrid systems; Model checking; Abstraction

1. Introduction

Recently, there has been a considerable interest in extending methods for formal verification of discrete systems, such as digital circuit designs, to control systems with hybrid dynamics. The goal of formal verification is to demonstrate that a specification for the system is satisfied for all possible system behaviors, starting from a given set of initial conditions and possibly for ranges of system parameters. Because the sets of reachable states for continuous dynamic systems can only be approximated in most cases, current methods for hybrid system verification will usually not terminate when a specification is not satisfied. Thus, current methods can provide positive verification results, but will not provide negative results. This paper presents a method for discovering when a specification will fail using existing tools by augmenting the given specification with an additional term called the *strong negation* of the specification.

The standard approach to hybrid system verification is to construct a finite-state abstraction so that methods for verifying properties of discrete state systems can be applied to hybrid dynamic systems (Alur, Henzinger, Lafferriere, & Pappas, 2000; Chutinan & Krogh, 2001). The main restriction in this approach is that only *universal* properties can be verified. A universal property is a property that is asserted over all possible behaviors of the system. For example, the statement that "the speed of the car remains between the lower and upper speed limits for all possible trajectories" is a universal property. Various topics devoted to universal properties can be found in the formal verification literature. In particular, this paper focuses on the universal properties specified by a class of *computation tree logic* (CTL) called ACTL (Clarke, Grumberg, & Peled, 1999). For a universal property, if the specification is true for the abstraction, it is true for the hybrid system, since all possible hybrid trajectories are accounted for in the conservative abstraction. If the model checker returns a negative result, one is assured the specification is false for the hybrid system only if the abstraction is an absolute equivalent representation, called a *bisimulation*, of the

☆A preliminary version of this paper appeared in the Preprints of the IFAC-Conference Workshop on Discrete Event Systems, Reims, France, September 2004.

*Corresponding author.
E-mail addresses: zhih@ece.cmu.edu (Z. Han), Alongkrit.Chutinan@mathworks.com (A. Chutinan), krogh@ece.cmu.edu (B.H. Krogh).

0967-0661/$ - see front matter © 2006 Elsevier Ltd. All rights reserved.
doi:10.1016/j.conengprac.2006.02.009

original hybrid system (Alur et al., 2000). As this is rarely the case, the only recourse is to refine the abstraction; that is, to make the abstraction less conservative, and then try again with the hope that eventually an abstraction will be constructed for which the specification is true. In certain cases, checking the strong negation condition introduced in this paper will allow an abstraction-based verification tool to terminate with a conclusive negative result, even if the abstraction is not a bisimulation of the hybrid system.

Methods based on abstractions which "strongly preserve" certain classes of properties have been proposed for various verification frameworks (Dams, Gerth, & Grumberg, 1997, 1993; Godefroid, Huth, & Jagadeesan, 2001; Godefroid & Jagadeesan, 2002). These frameworks have two types of transitions in the abstraction: `must`-transitions and `may`-transitions. A transition in the abstraction is a `must`-transition if there exists at least one corresponding *actual* transition in the underlying system, otherwise it is labeled as a `may`-transition, where the corresponding transition may or may not exist in the actual system. By distinguishing the two types of transitions in the algorithm, conclusive negative results can be obtained for the system. This technique is not applicable for verification of hybrid systems, however, since the finite-state abstraction of a hybrid system is based on reachability analysis for the continuous dynamics (Alur et al., 2000; Chutinan & Krogh, 2001), where all transitions are `may`-transitions. To construct the `must`-transitions for a hybrid dynamic system, the set of states reachable from each abstract state has to be computed, which is the most expensive computation in the verification procedure because it involves the computation of sets of reachable states for continuous dynamic systems.

For background, the following section describes CheckMate, a MATLAB-based tool for hybrid system verification. Section 3 introduces the basic definitions used in formal verification. Section 4 presents strong negation for ACTL formulas and shows how it can be used to demonstrate how an ACTL expression is not satisfied based on verification of an abstraction of a hybrid system. Section 5 illustrates the application of strong negation to provide conclusive negative results. The concluding section summarizes the contributions of this paper and discusses directions for further work.

2. Hybrid system verification using CheckMate

This section introduces CheckMate, a tool developed at Carnegie Mellon University to perform simulation and formal verification of hybrid dynamical systems (Chutinan & Krogh, 2003; Silva, Richeson, Krogh, & Chutinan, 2000). CheckMate is developed in the MATLAB-SIMULINK environment. In contrast to tools like HYTECH (Alur, Henzinger, & Ho, 1996), where only linear hybrid automata are considered, CheckMate verifies hybrid system models with linear or nonlinear continuous dynamics and polyhedral guard conditions. CheckMate models assume *urgent* semantics: if one of the polyhedral thresholds is enabled, the system makes a switch immediately. Due to the complexity in polyhedra computation, CheckMate has been limited to hybrid dynamic systems with order less than 6. The CheckMate software and its document are available at http://www.ece.cmu.edu/~webk/checkmate.

Hybrid systems are modeled in CheckMate using SIMULINK block diagrams, and formal specifications of desired properties are expressed as ACTL formulas (defined in Section 3). The basic structure of a CheckMate model is a closed loop consisting of a switched continuous system block (SCSB), a set of polyhedral threshold blocks (PTHBs) and a finite state machine block (FSMB). The following paragraphs describe these blocks.

An SCSB models the continuous dynamics of the hybrid system. The continuous dynamics in CheckMate are modeled with state equations of the form $\dot{x} = f_d(x)$, where d is the discrete mode of the system. To model the switching behavior of the hybrid system, the SCSB block has a input port d which is the discrete state of the hybrid model. For different values of the discrete states, the SCSB selects different sets of ODEs. The discrete state input is generated from the FSMB (described below).

PTHBs represent convex polyhedral sets described by $\{x|Cx \leqslant d\}$. CheckMate models hybrid systems that switch when the state reaches boundaries of polyhedral sets in the state space.

An FSMB models the switching scheme of the hybrid systems using a STATEFLOW chart. The inputs to the chart are the events generated by PTHBs. The outputs of the chart are the discrete states, which govern the switching of the SCSB.

The verification procedure of CheckMate is based on approximating the infinite-state transition system using the abstraction framework called the approximate *quotient transition system* (QTS) (Chutinan & Krogh, 2001). Flowpipes are the sets of states reachable along trajectories starting from a given set of initial states. The boundaries of the polyhedral thresholds are partitioned into polyhedral regions that represent the set of states reached by the flowpipe. The computations of reachable states (flowpipes) continues until all reachable boundaries have been partitioned. This creates the initial QTS. The QTS is discussed in Section 3.

Fig. 1a shows an overview of the CheckMate verification procedure. Given an ACTL formula φ and a CheckMate model, CheckMate first converts the model into an equivalent hybrid automaton model (Henzinger, 1996). Then an initial partition of the hybrid automaton is computed using *flow-pipe* computations (Chutinan & Krogh, 2003). The QTS is then verified against the given ACTL specification φ using standard model checking techniques for finite transition systems (Clarke et al., 1999). If the verification fails due to the coarseness of the partition for the QTS, the partition is refined to get a tighter approximation. The process can be repeated until the QTS