Software Tools for Technology Transfer manuscript No.
(will be inserted by the editor)

Synthesizing Switching Logic using Constraint Solving*

Ankur Taly[1], **Sumit Gulwani**[2], **Ashish Tiwari**[3]

[1] Computer Science Dept., Stanford University e-mail: ataly@stanford.edu
[2] Microsoft Research, Redmond, WA 98052, e-mail: sumitg@microsoft.com
[3] SRI International, Menlo Park, CA 94025, e-mail: tiwari@csl.sri.com

Received: date / Revised version: date

Abstract. For a system that can operate in multiple different modes, we define the *switching logic synthesis problem* as follows: given a description of the dynamics in each mode of the system, find the conditions for switching between the modes so that the resulting system satisfies some desired properties. In this paper, we present an approach for solving the switching logic synthesis problem in the case when
(i) the dynamics in each mode of the system are given using differential equations and, hence, the synthesized system is a hybrid system, and
(ii) the desired property is a safety property.

Our approach for solving the switching logic synthesis problem, called the *constraint-based approach*, consists of two steps. In the first *constraint generation* step, the synthesis problem is reduced to satisfiability of a quantified formula over the theory of reals. In the second *constraint solving* step, the quantified formula is solved. This paper focuses on constraint generation.

The constraint generation step is based on the concept of a *controlled inductive invariant*. The search for controlled inductive invariant is cast as a constraint solving problem. The controlled inductive invariant is then used to arrive at the maximally liberal switching logic. We prove that the synthesized switching logic always gives us a well-formed and safe hybrid system.

When the system, the safety property, and the controlled inductive invariant are all expressed only using polynomials, the generated constraint is an $\exists\forall$ formula in the theory of reals, whose satisfiability is decidable.

* Research supported in part by the National Science Foundation under grants CNS-0720721, CSR-EHCS-0834810, CSR-0917398 and CCF-1017483 and by NASA under Grant NNX08AB95A. Work done when the first author was visiting SRI International.

1 Introduction

Formal verification is beginning to play an important role in the process of building reliable and certifiable complex engineered systems. Rather than design and then verify, an alternate approach is to automatically synthesize correct systems. The synthesis approach is attractive since it generates correct systems by design. However, computationally, the synthesis problem appears to be much harder than the verification problem and there are few general approaches for solving it.

Approaches for verification can be broadly classified into two classes based on whether they are based on iterative fixpoint calculations or computing abstractions. Recently, a third approach, called the *constraint-based* approach, for verification has been investigated [9,8]. The constraint-based approach works in two steps. In the first step, called the *constraint generation* step, the verification problem is reduced to satisfiability of an $\exists\forall$ formula over some theory. In the second step, called the *constraint solving* step, the generated constraint is solved using an existing solver or some heuristic wrapper over a symbolic Satisfiability Modulo Theory (SMT) solver or a numeric solver. While both steps are crucial for success of the constraint-based approach, the constraint generation step is particularly important since it decides the size, form and complexity of the constraint. This paper focuses on constraint generation.

The reduction of the verification problem to a constraint solving problem is achieved by noticing that verification of several important properties boils down to finding an appropriate "witness". For example, we can verify safety by searching for a strong enough inductive invariant: a set of states is an inductive invariant if it is a superset of the set of all reachable states (the set is an invariant) and the immediate successor of any (reachable or unreachable) state in the set is also in the set (the set is inductive). Unfortunately, the space of possi-

ble invariants is huge and it is impractical to search over this entire space. Here we take the help of the human user. We assume that the user has an idea of the "form" of the invariant. Using this knowledge of the form, the "unbounded" search for invariants is "bounded" by focusing the search only on invariants that are of that specific form. The form of the invariant is specified using a "template", which is an expression containing some holes. The existence ($\exists$) of an appropriate instance of the template that is also an inductive invariant ($\forall$) naturally maps to an $\exists\forall$ formula. This is the essence of the constraint generation step. If the $\exists\forall$ formula is valid (over the underlying theory), then it means that there exists an inductive invariant (of the given form) that proves safety.

The constraint-based approach for verification naturally generalizes to a constraint-based approach for synthesis. Given an under-specified system, we can choose templates for the unknown parts of the system, as well as for the unknown inductive invariant. We can then simultaneously seach for the unknown parts of the system and the unknown invariant. Specifically, we existentially quantify on the "holes" in the system and the "holes" in the template. The constraint solver then searches for instances of *all* these "holes" so that the resulting system is proved safe by the resulting invariant. In practice, however, this naive approach does not work well for synthesis because the constraint solver often chooses values that result in a degenerate system (such as, a zeno system, or a deadlocked system) where the safety property is vacuously true. Moreover, the above method does not take advantage of the correlations that exist between the various unknowns and uses a separate template for each unknown. Having too many templates contributes to the incompleteness and reduces the effectiveness of the approach.

In this paper, we define a specific instance of the synthesis problem, called the *switching logic synthesis* problem. We present a constraint-based approach, inspired by [9], to solve the switching logic synthesis problem; see Figure 1. The novelty in our approach here is that we do not directly search for values to fill the "holes" in the system; that is, we do not directly search for the switching conditions. Instead we use constraint solving to find an *inductive controlled invariant* set. Hence we only have to choose a single template – for the inductive control invariant – and none for the unknown switching conditions. In a final postprocessing step, we use the generated controlled invariant to synthesize the actual switching logic. This postprocessing step generates the weakest (most general) possible controller from the controlled invariant. Our approach is guaranteed to synthesize a non-blocking hybrid system that is also safe, whenever some such hybrid system exists.

Inductive Controlled Invariant. An invariant for a system is any superset of the set of reachable states of that system. Safety properties can be proved by finding suitable invariants. However, invariance is difficult to check in general. A better alternative is to search for *inductive* invariants. Inductive invariants are attractive because inductiveness is a "local" property – for each state in the inductive set, we only need to check that the *immediate next* states reached from that state (rather than *all* reachable states) are also in the inductive set. Fortunately, the set of reachable states is always inductive and hence, the use of inductive invariants is a sound and complete method for safety verification.

In this paper, we consider systems that contain controllable choices, that is, the user/controller can make selections to achieve some safety goal. For such systems, the notion corresponding to invariant sets is called *controlled invariant*. A controlled reach set is the set of reachable states obtained for some choice of the controller. A controlled invariant is a superset of *some* controlled reach set. As before, the computationally interesting notion is that of an *inductive* controlled invariant. We can, therefore, synthesize safe controllers by generating the correct inductive controlled invariant. In this paper, we pursue this idea in the context of hybrid systems, though the idea of inductive controlled invariant is applicable more generally.

Hybrid Systems. There are two classic formalisms for modeling dynamical systems, namely continuous dynamical systems and discrete state transition systems. Continuous dynamical systems are used to model dynamics of physical entities, such as temperature, pressure, speed and angular velocity. In a continuous dynamical system (CDS), the state variables take values in the reals, $\mathbb{R}$, and hence the state space is usually the n-dimensional real space, $\mathbb{R}^n$. The dynamics of a CDS are given using a system of ordinary differential equations – one for each of the n state variables. On the other hand, discrete state transition systems are used to model the dynamics of discrete states, such as states of a program or protocol. The state variables take values in a finite or countable set and the dynamics are given using rules that specify how and when the state variables are updated.

Hybrid systems are obtained by combining the formalism of continuous dynamical systems and discrete state transition systems. In this paper, we will focus on switched systems: a system with finitely many modes, where each mode is equivalent to a continuous dynamical system, and a set of rules, so-called "switching logic", for switching between the different modes of the system. Intuitively, at any given time, such a system is in a particular mode and evolving according to that CDS, until some rule is enabled. When a rule is enabled, the system "switches" to a different mode and evolves according to the new CDS.

Often the switchings are controllable, and the goal of switching logic synthesis is to find a switching logic that will guarantee some desired property of the result-

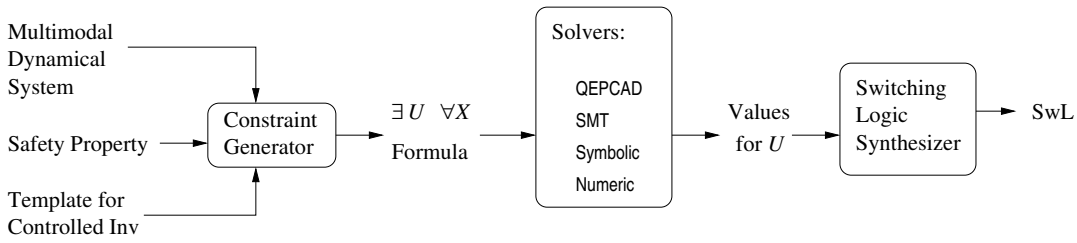

Fig. 1. Overview of the constraint-based approach for switching logic synthesis. Boxes denote software tools and arrows denote inputs and outputs of the tools.

ing system. In this paper, we are interested in the class of safety properties.

Contribution and Outline of the Paper. The outline of the paper is as follows:

- We formally define the switching logic synthesis problem (Section 2).
- We formalize the notion of inductive controlled invariants for multi-modal systems and describe a sound and complete approach for synthesizing switching logic from an inductive controlled invariant (Section 3). Our synthesis technique follows the constraint-based approach and does not use the usual game theoretic approach for controller synthesis, or the controlled reachability approach (also see Section 9).
- Whereas inductiveness is easily definable for discrete transitions, the concept of inductiveness for continuous dynamical systems is not so easy to formulate in a way that makes it checkable. We describe several sufficient checks that guarantee that a set is inductive with respect to some given continuous dynamics (Section 4). These sufficient tests provide foundational rules for verification of inductiveness in continuous systems.
- Using the sufficient checks for inductiveness for continuous systems, we arrive at a practical implementation of the constraint-based synthesis approach (Section 5).
- We also describe some heuristics to generate large controlled invariant sets, that lead to synthesis of liberal controllers (Section 6).

The key contributions of the paper include sufficient tests for verifying inductiveness in continuous systems and a constraint-based approach for switching logic synthesis. The focus of this paper is on presenting the problem definition and the theoretical results underlying the correctness of the synthesis approach. We provide several examples to illustrate the main definitions and the overall approach throughout the paper. Preliminary versions of the results presented in this paper can be found in [24, 25, 9].

2 The Switching Logic Synthesis Problem

In this section, we describe the synthesis problem considered in this paper. We motivate our formal definitions with informal descriptions of the problem.

We are interested in controlling multi-modal continuous dynamical systems. A dynamical system is defined by its state space, which is the set of all possible configurations or states of the system, and its dynamics, which defines how the system configuration changes (with time). Formally, a *continuous dynamical system* (CDS) is a tuple $\langle X, f \rangle$ where X is a set of n real-valued variables that define the state space $\mathbb{R}^n$ and $f : \mathbb{R}^n \mapsto \mathbb{R}^n$ is a vector field that specifies the continuous dynamics. The dynamic behavior of the CDS is given by a trajectory, which is a mapping $\mathbf{x}$ from time $[0, \infty)$ to the state space $\mathbb{R}^n$, that satisfies the system of differential equations $\frac{d\mathbf{x}}{dt} = f(\mathbf{x})$.

We assume that f is locally Lipschitz at all points, which guarantees the existence and uniqueness of solutions of the above system of ordinary differential equations.

Proposition 1 (Theorem 2.3.1, p80 [4]). *Consider a Lipschitz vector field f and the system of differential equations $\frac{d\mathbf{x}(t)}{dt} = f(\mathbf{x}(t))$, $\mathbf{x}(0) = \mathbf{x_0}$. The solution of this system of differential equations, denoted by $F(\mathbf{x_0}, t)$, always exists and is unique. Moreover, $F(\mathbf{x_0}, t)$ depends continuously on the initial state $\mathbf{x_0}$.*

Often a single continuous dynamical system is insufficient to describe all possible behaviors of a system. Many systems have multiple modes and they have different dynamics in each mode. This happens, for example, when we introduce actuators inside physical devices that change the device's dynamics. In such cases, the dynamics of a system is described by a *collection* of CDSs. We call such a system a multi-modal dynamical system. A multi-modal system has a finite number of different modes and in each mode, it behaves like a different continuous dynamical system. For instance, consider the water level in a tank with an inflow valve. Such a system has two dynamics – one when the valve is closed and one when it is open. Formally, we define a multi-modal dynamical system (`MDS`) and its semantics as follows.

Definition 1 (Multi-modal Continuous Dynamical System). A *multi-modal continuous dynamical system*, MDS, is a tuple $\langle X, f_1, f_2, \ldots, f_k, \texttt{Init}\rangle$, where $\langle X, f_i\rangle$ is a continuous dynamical system (representing the i-th mode) and $\texttt{Init} \subseteq \mathbb{R}^X$ is the set of initial states. Given an initial state $\mathbf{x_0} \in \texttt{Init}$, we say that a function $\mathbf{x}(t) : [0, \infty) \to R^X$ is a *trajectory* for MDS, if there is an increasing sequence $0 \leq t_1 < t_2 < \cdots$ (either finite or diverging to ∞) such that

- $\mathbf{x}(0) = \mathbf{x_0}$ and $\mathbf{x}(t)$ is continuous over $t \geq 0$, and
- for each interval (t_i, t_{i+1}), there is a mode $j \in I$ such that $\mathbf{x}(t)$ is smooth and $\frac{d\mathbf{x}(t)}{dt}(t') = f_j(\mathbf{x}(t'))$ for all t' in the range $t_i < t' < t_{i+1}$. When $i = 0$, then we require $j = 1$; that is, mode 1 is the initial mode.

Note that we are fixing the domain of a trajectory to always be the set $[0, \infty)$. Following Definition 1, a multi-modal system can nondeterministically switch between its modes. However, switching between the different modes in a multi-modal dynamical system is often controllable. The goal of controlling a system is to achieve safe operation with some desired performance. For instance, in the water tank example, the transition between the two modes can be controlled by opening and closing the valve. The controller may be required to guarantee that the water level in the tank remains between two thresholds. There are several controllers that can achieve this property. A controller that opens the valve just when the water level reaches the lower threshold and closes it soon thereafter, will keep the level closer to the lower threshold, but it is very restrictive as it prevents the system from reaching several possible safe states. We are interested in designing controllers that guarantee safety, but that also do not unnecessarily restrict the system from reaching safe states.

A controller for a multi-modal system is specified as a *switching logic*.

Definition 2 (Switching Logic). A *switching logic*, SwL, for a multi-modal dynamical system, $\texttt{MDS} := \langle X, (f_i)_{i\in I}, \texttt{Init}\rangle$, is a tuple $\langle (g_{ij})_{i\neq j; i,j\in I}, (\texttt{Inv}_i)_{i\in I}\rangle$, containing guards $g_{ij} \subseteq \mathbb{R}^X$ and state (location) invariants $\texttt{Inv}_i \subseteq \mathbb{R}^X$.

Informally, the guard g_{ij} specifies the condition under which the system *could* switch from mode i to mode j and the state invariant $\texttt{Inv}_i$ specifies the condition which *must* be respected while in mode i.

A multi-modal system MDS can be combined with a switching logic SwL to create a hybrid system $\texttt{HS} := \texttt{HS}(\texttt{MDS}, \texttt{SwL})$ in the following natural way:
(a) the hybrid system HS has $|I|$ modes with dynamics given by $\frac{d\mathbf{x}(t)}{dt} = f_i(\mathbf{x}(t))$ in mode i,
(b) g_{ij} is the guard on the discrete transition from mode i to mode j, and
(c) $\texttt{Inv}_i$ is the state invariant in mode i.
The state space of the hybrid system is $I \times \mathbb{R}^n$. The initial states are $\{1\} \times \texttt{Init}$. The discrete transitions in HS have identity reset maps, that is, the continuous variables do not change values during discrete jumps. The semantics of hybrid systems that define the set of reachable states of hybrid systems are standard [1]. The set of trajectories of the hybrid system $\texttt{HS}(\texttt{MDS}, \texttt{SwL})$ is a subset of the set of trajectories of the underlying MDS containing only those trajectories whose mode switchings are consistent with the switching logic SwL: specifically, a mode switch from mode i to mode j at state $\mathbf{x}(t)$ is consistent with SwL if
(i) $\mathbf{x}(t) \in g_{ij}$ and
(ii) $\mathbf{x}(t) \in \texttt{Inv}_i$ and $\mathbf{x}(t) \in \texttt{Inv}_j$.

Technically, the reachable states of a hybrid system are tuples $(i, \mathbf{x})$, where $i \in I$ is the mode and $\mathbf{x} \in \mathbb{R}^n$ is the continuous state, but we often project the set of reachable states on its $\mathbf{x}$ component (and ignore the i component) and consider this projection on $\mathbb{R}^n$ as the set of reachable states of the hybrid system.

Though semantically well-defined, some hybrid systems have undesirable behaviors. For example, it can happen that a hybrid system, in mode i, reaches a point $\mathbf{x}$ on the boundary of $\texttt{Inv}_i$, but there is no valid trajectory from $\mathbf{x}$; that is, there is no discrete transition enabled at $\mathbf{x}$, and following mode i dynamics takes the system out of $\texttt{Inv}_i$. The non-blocking requirement disallows such cases. We are interested in synthesizing non-blocking hybrid systems.

Definition 3. A hybrid system HS is said to be *non-blocking* if for every mode i, and for every point $\mathbf{x}$ on the boundary of the state invariant for mode i, there exists a mode j (may be same as i) and $\epsilon > 0$ such that
(i) $i = j$ or $\mathbf{x} \in g_{ij}$, and
(ii) the dynamics of mode j keeps the system within the state invariant of mode j for at least ϵ time.

A hybrid system HS is safe with respect to a safety property $\texttt{Safe} \subseteq \mathbb{R}^n$ if the set of its reachable states is contained in Safe. Formally, we define the logic synthesis problem as follows:

Definition 4 (Switching Logic Synthesis Problem). Given a multi-modal dynamical system, $\texttt{MDS} := \langle X, f_1, f_2, \ldots, f_k, \texttt{Init}\rangle$ and a safety property $\texttt{Safe} \subseteq \mathbb{R}^n$, the *switching logic synthesis problem* seeks to synthesize a switching logic SwL such that the hybrid system $\texttt{HS}(\texttt{MDS}, \texttt{SwL})$ is non-blocking and safe with respect to Safe.

3 The Synthesis Procedure

In this section we present a high-level procedure for solving the switching logic synthesis problem described in Definition 4. We fix our notation and denote the given multi-modal dynamical system by MDS, its initial set of states by Init and the given safety property by Safe; see also Table 1.

We first define the notion of a controlled invariant.

X	:	set or vector of n variables
f	:	vector field, $\mathbb{R}^n \mapsto \mathbb{R}^n$
Inv, CInv	:	subset of $\mathbb{R}^n$
Init, Safe	:	subset of $\mathbb{R}^n$
p	:	polynomial over X; $\mathbb{R}^n \mapsto \mathbb{R}$
$\mathbf{x}, \mathbf{y}$	:	point in the set $\mathbb{R}^n$

Table 1. Summarizing the notation used in the paper. Note that we do not distinguish between $\mathbb{R}^X$ and $\mathbb{R}^n$.

Definition 5 (Controlled Invariant). A set CInv is said to be a *controlled invariant* for a MDS $:= \langle X, (f_i)_{i\in I}, \texttt{Init}\rangle$ if for all $\mathbf{x}_0 \in$ Init, there exists a trajectory (Definition 1) $\mathbf{x}(t)$ such that $\mathbf{x}(0) = \mathbf{x}_0$ and for all $t \geq 0$, $\mathbf{x}(t) \in$ CInv.

Note that an invariant requires that, for each initial state, *every* trajectory (starting from that initial state) remains inside the invariant. In contrast, a controlled invariant only requires that, for each initial state, there is *some* trajectory (starting from that initial state) that remains inside the controlled invariant.

Example 1. Let $\dot{x}$ denote $\frac{dx}{dt}$. Consider a multi-modal system with two modes and over two variables x and y.

$$\text{Mode 1}: \ \dot{x} = 1, \ \dot{y} = 0$$
$$\text{Mode 2}: \ \dot{x} = 0, \ \dot{y} = 1$$

Suppose $x = 0, y = 0$ is the only initial state. Irrespective of how the system switches modes, the value of x and y will be non-negative. Hence, $x \geq 0 \wedge y \geq 0$ is an invariant of this multi-modal system. However, $x \geq 0 \wedge y = 0$ is not an invariant since mode 2 can cause y to become positive. But, there is a switching strategy – never switch to mode 2 and always remain in mode 1 – that guarantees that y remains zero. Hence, $x \geq 0 \wedge y = 0$ is a controlled invariant. Finally, note that the set $x + y \leq 0$ is neither an invariant nor a controlled invariant. □

Definition 5 does not suggest any easy way to compute nontrivial controlled invariants. Hence, we define the notion of *inductive* controlled invariants. Since the dynamics are continuous here, we first need to define a few notions. Recall that the vector fields f_i's are Lipschitz and hence, by Proposition 1, we have a unique trajectory $F_i(\mathbf{x_0}, t)$ in mode i that starts from state $\mathbf{x_0}$. By $F_i(\mathbf{x_0}, (0, \epsilon))$ we denote the set of all points reached in the time interval $(0, \epsilon)$; that is,

$$F_i(\mathbf{x_0}, (0, \epsilon)) := \{\mathbf{x} \mid \mathbf{x} = F_i(\mathbf{x_0}, t), 0 < t < \epsilon\}.$$

For a set $S \subseteq \mathbb{R}^n$, let ∂S denote the boundary of S in the topological sense. We are now ready to define inductive controlled invariants.

Definition 6 (Inductive Controlled Invariant). A closed set CInv is an *inductive controlled invariant* for MDS $:= \langle X, (f_i)_{i\in I}, \texttt{Init}\rangle$ if

(A1) Init $\subseteq$ CInv and

(A2) $\forall \mathbf{x} \in \partial$CInv $: \exists i \in I : \exists \epsilon > 0 : F_i(\mathbf{x}, (0, \epsilon)) \subseteq$ CInv

```
SynthSwitchLogic(MDS, Safe) :
1. Find a closed set CInv that satisfies
   Conditions (A1) and (A2) from Definition 6 and
   Condition (A3):   CInv ⊆ Safe
   If no such set is found, return failure
2. Let bdry_i := {x ∈ ∂CInv | ∃ε > 0 : F_i(x, (0, ε)) ⊆ CInv}
     for all i ∈ I
3. Let Inv_i := CInv for all i ∈ I
4. Let g_ij := bdry_j ∪ Interior(CInv) for all i ≠ j; i, j ∈ I,
Return SwL := ⟨(g_ij)_{i≠j;i,j∈I}, (Inv_i)_{i∈I}⟩
```

Fig. 2. Procedure for synthesizing switching logic presented at a semantic level

Intuitively, Condition (A2) in Definition 6 says that for every point on the boundary of CInv, there is a vector field f_i that points inwards and brings the system (instantaneously) inside the set CInv, see also [3]. Just as inductive invariants are also invariants, inductive controlled invariants are also controlled invariants.

Proposition 2. *If a closed set CInv is an inductive controlled invariant for MDS, then it is also a controlled invariant for MDS.*

Proposition 2 is a direct consequence of the definitions of controlled invariants and inductive controlled invariants.

The complete procedure, at a semantic level, for solving the switching logic synthesis problem is presented in Figure 2. The key idea behind the synthesis procedure is to find an inductive controlled invariant set CInv and then design the switching logic so that the resulting hybrid system always remains in CInv. Conditions (A1) and (A2) imply that CInv is an inductive controlled invariant and Condition (A3):

(A3) CInv $\subseteq$ Safe

implies that all states in the set CInv are safe. It follows from Condition (A2) that the boundary ∂CInv of the set CInv can be written as a union

$$\partial\texttt{CInv} = \bigcup_{i\in I} \texttt{bdry}_i \tag{1}$$

where $\texttt{bdry}_i$ contains all points $\mathbf{x}$ on the boundary ∂CInv such that $\exists \epsilon > 0 : F_i(\mathbf{x}, (0, \epsilon)) \subseteq$ CInv. This fact is used to define the sets $\texttt{bdry}_i$ in Line 2. In Line 4, we use the sets $\texttt{bdry}_i$ and CInv to define the guards for the discrete transitions from mode i to mode j.

We next state and prove some properties of the procedure SynthSwitchLogic in Figure 2. We show that the synthesized hybrid system is always non-blocking and safe (soundness). Furthermore, if there is a safe hybrid system, then under some fairly general conditions, the procedure SynthSwitchLogic will return a switching logic SwL and synthesize a safe system HS(MDS, SwL) (completeness).

Theorem 1 (Soundness). *For every switching logic SwL returned by procedure SynthSwitchLogic(MDS,Safe),*

the hybrid system $\mathtt{HS}(\mathtt{MDS}, \mathtt{SwL})$ *is non-blocking and safe with respect to* $\mathtt{Safe}$.

Proof. We first prove that the hybrid system $\mathtt{HS}(\mathtt{MDS}, \mathtt{SwL})$ is non-blocking. Let $\mathbf{x}$ be any point on the boundary $\partial\mathtt{CInv}$ of the set $\mathtt{CInv}$. From Equation (1), we know that there is a set $\mathtt{bdry}_j$ such that $\mathbf{x} \in \mathtt{bdry}_j$. Also by definition, for all $i \neq j$, $g_{ij} = \mathtt{bdry}_j$ and hence, $\mathbf{x} \in g_{ij}$. Furthermore, we also know that $\forall \mathbf{x} \in \mathtt{bdry}_j : \exists \epsilon : F_j(\mathbf{x}, [0, \epsilon]) \subseteq \mathtt{CInv}$. This implies that there is an $\epsilon > 0$ such that the system can stay in mode j for ϵ time. This proves that the hybrid system is non-blocking.

Given a set $\mathtt{CInv}$ satisfying Condition (A1), Condition (A2) and Condition (A3), we have proved that the switching logic $\mathtt{SwL}$ generated from $\mathtt{CInv}$ results in a non-blocking hybrid system $\mathtt{HS}(\mathtt{MDS}, \mathtt{SwL})$. Since the state invariant $\mathtt{CInv}$ satisfies $\mathtt{CInv} \subseteq \mathtt{Safe}$ (Condition (A2)), safety follows trivially for the non-blocking hybrid system $\mathtt{HS}(\mathtt{MDS}, \mathtt{SwL})$. □

We prove completeness under a technical assumption. We say a hybrid system $\mathtt{HS}$ has the *min-dwell-time* property if there exists a fixed time duration t_a such that for all reachable states $\mathbf{x}$, if the hybrid system permits a mode switch from i to j at $\mathbf{x}$, then there must exist a mode k such that the hybrid system permits a mode switch from i to k at $\mathbf{x}$ and the system can stay in mode k for at least t_a units of time starting at $\mathbf{x}$. The min-dwell-time property implies that successive mode switchings can be forced to be t_a units apart.

Example 2 (Illustrating min-dwell-time property). The following hybrid system does not have the min-dwell-time property:

$$\begin{array}{rll} \text{Mode } 1: & \dot{x} = -1,\ \dot{y} = 3, & \mathtt{Inv}_1 := (x - y \geq 0) \\ \text{Mode } 2: & \dot{x} = -1,\ \dot{y} = -3, & \mathtt{Inv}_2 := (x + y \geq 0) \\ \text{Guard } g_{12}: & x - y = 0 & \\ \text{Guard } g_{21}: & x + y = 0 & \end{array}$$

where $\mathtt{Inv}_1$ and $\mathtt{Inv}_2$ are the mode invariants. It is easy to see that, as the system state approaches $(0, 0)$, the time between successive switchings monotonically decreases, reaching 0 in the limit. Hence, this hybrid system does not have the min-dwell-time property. □

Theorem 2 (Completeness). *If there exists a switching logic* $\mathtt{SwL}$ *such that the hybrid system* $\mathtt{HS}(\mathtt{MDS}, \mathtt{SwL})$ *is non-blocking, it is safe with respect to the closed set* $\mathtt{Safe}$*, and it satisfies the min-dwell-time property, then the procedure* $\mathtt{SynthSwitchLogic(MDS,Safe)}$ *will return a switching logic.*

Proof. Consider the hybrid system $\mathtt{HS} = \mathtt{HS}(\mathtt{MDS}, \mathtt{SwL})$ that is assumed to exist. Let I denote the modes of $\mathtt{HS}$ and let X be the continuous variables of $\mathtt{HS}$. Let $R_{\mathtt{HS}}$ be the set of reachable states of $\mathtt{HS}$. Let R be the projection of $R_{\mathtt{HS}}$ on the continuous variables; that is,

$$R := \{\mathbf{x} \in \mathbb{R}^X \mid \exists i \in I : (i, \mathbf{x}) \in R_{\mathtt{HS}}\}$$

Consider the closure $\mathtt{Cl}(R) = R \cup \partial R$ of R. We will prove the theorem by showing that the set $\mathtt{Cl}(R)$ satisfies the three conditions (A1), (A2) and (A3).

First, since initial states are reachable by definition, $\mathtt{Init} \subseteq \mathtt{Cl}(R)$. Hence, (A1) holds. Next we show that $\mathtt{Cl}(R) \subseteq \mathtt{Safe}$. Since $\mathtt{HS}$ is safe, we know that $R \subseteq \mathtt{Safe}$. Taking closure of both sides, we have $\mathtt{Cl}(R) \subseteq \mathtt{Cl}(\mathtt{Safe})$. Since $\mathtt{Safe}$ is a closed set, $\mathtt{Cl}(\mathtt{Safe}) = \mathtt{Safe}$. Therefore $\mathtt{Cl}(R) \subseteq \mathtt{Safe}$ and hence (A3) holds.

Finally, we show that $\mathtt{Cl}(R)$ satisfies Condition (A2). Let $\mathbf{x} \in \partial\mathtt{Cl}(R)$. We need to show that there is a mode i s.t. for some $\epsilon > 0$, we have $F_i(\mathbf{x}, [0, \epsilon]) \subseteq \mathtt{Cl}(R)$.
Case $1 : \mathbf{x} \in R$. Let i be the mode s.t. $(i, \mathbf{x}) \in R_{\mathtt{HS}}$. If $\mathbf{x}$ is in the interior of the state invariant of mode i, then, since we assume vector fields to be locally Lipschitz, there is a small enough $\epsilon > 0$ s.t. $F_i(\mathbf{x}, [0, \epsilon]) \subseteq R$. If $\mathbf{x}$ is on the boundary of the state invariant of mode i, then, since $\mathtt{HS}$ is assumed to be non-blocking, we know there is some $\epsilon > 0$ and some mode j s.t. $F_j(\mathbf{x}, [0, \epsilon]) \subseteq R$. Thus, in both cases, we have established Condition (A2).
Case $2 : \mathbf{x} \in \mathtt{Cl}(R) - R$. For any $\epsilon > 0$, let B_ϵ be an ϵ-ball around $\mathbf{x}$. Let $B_{\epsilon,i}$ be points in this ball that are reachable with mode i. Formally,

$$B_\epsilon := \{\mathbf{y} \in \mathbb{R}^X \mid ||\mathbf{y} - \mathbf{x}|| < \epsilon\}$$
$$B_{\epsilon,i} := \{\mathbf{y} \in B_\epsilon \mid (i, \mathbf{y}) \in R_{\mathtt{HS}}\}$$

Since $\mathbf{x} \in \mathtt{Cl}(R)$, there will be reachable points close to $\mathbf{x}$. Hence, for some i, $B_{\epsilon,i}$ will be nonempty. Since there are only finitely many modes in I, there will be some mode, say i^*, s.t. for all $\epsilon > 0$, B_{ϵ,i^*} is nonempty. For a point $\mathbf{y} \in B_{\epsilon,i^*}$, let $t_e(\mathbf{y})$ denote the least time at which a switch to a different mode is enabled in any trajectory of $\mathtt{HS}$. Formally,

$$t_e(\mathbf{y}) := \inf\{t \mid \text{at state } F_{i^*}(\mathbf{y}, t),\ \mathtt{HS} \text{ permits mode switch from mode } i^* \text{ to some mode } j\}$$

If $\mathtt{HS}$ does not permit a mode switch at state $F_{i^*}(\mathbf{y}, t)$ for any $t \geq 0$, then let $t_e(\mathbf{y}) = \infty$. Now, pick any infinite sequence $\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, \ldots$ of points such that $\mathbf{x}_n \in B_{\epsilon/2^n, i^*}$. By construction, this sequence of reachable points converges to $\mathbf{x}$. Consider now the sequence

$$t_e(\mathbf{x}_1), t_e(\mathbf{x}_2), \ldots \tag{2}$$

of time durations after which a mode switch is enabled.

There are two subcases. Either Sequence (2) converges to 0 or it does not. First consider the case when it does not. Then there exists some $t^* > 0$ s.t. infinitely many elements in Sequence (2) are greater than t^*. This means that there are *reachable* points arbitrarily close to $\mathbf{x}$ s.t. $\mathtt{HS}$ trajectories starting from those points remain in mode i^* and do not switch modes for at least t^* time units. Consider the following sequence of points in R,

$$F_{i^*}(\mathbf{x}_1, t^*),\ F_{i^*}(\mathbf{x}_2, t^*),\ \ldots \tag{3}$$

Proposition 1 says that $F_{i^*}(\mathbf{x}, t^*)$ is continuous in $\mathbf{x}$, and since the sequence $\mathbf{x_1}, \mathbf{x_2}, \ldots$ converges to $\mathbf{x}$, it follows

that Sequence (3) converges to $F_{i^*}(\mathbf{x}, t^*)$. Hence, we can conclude that $F_{i^*}(\mathbf{x}, t^*) \in \texttt{Cl}(R)$. The same reasoning holds for any t between 0 and t^* and hence $F_{i^*}(\mathbf{x}, [0, t^*]) \subseteq \texttt{Cl}(R)$. This establishes (A2).

Now we consider the case when Sequence (2) converges to 0. In this case, consider the new sequence of points

$$F_{i^*}(\mathbf{x}_1, t_e(\mathbf{x}_1)), F_{i^*}(\mathbf{x}_2, t_e(\mathbf{x}_2)), F_{i^*}(\mathbf{x}_3, t_e(\mathbf{x}_3)), \ldots \quad (4)$$

The first argument converges to $\mathbf{x}$ and the second argument converges to 0, hence this sequence converges to $F_{i^*}(\mathbf{x}, 0)$, which is $\mathbf{x}$. At each point in the sequence, by definition, switching is enabled. Using the min-dwell-time property, there is a sequence of destination modes $j_1, j_2, \ldots$ s.t. HS can spend t_a time in mode j_n starting from the state $F_{i^*}(\mathbf{x}_n, t_e(\mathbf{x}_n))$. Since there are only finitely many modes, there is at least one mode, say j^*, s.t. HS can spend t_a time in mode j^* starting from infinitely many points in Sequence (4). Hence, this means that there are *reachable* points arbitrarily close to $\mathbf{x}$ s.t. HS trajectories starting from those points remain in mode j^* (and do not switch modes) for at least t_a time units. As in the previous case, we immediately conclude that this implies $F_{j^*}(\mathbf{x}, [0, t_a]) \subseteq \texttt{Cl}(R)$. This establishes (A2).

Thus, we conclude that $\texttt{Cl}(R)$ satisfies Conditions (A1), (A2) and (A3) and hence procedure SynthSwitchLogic will return a switching logic. □

Although the above procedure is sound and complete, it is not computationally feasible as there is no easy way to check for Condition (A2). In the next section we will present effectively checkable sufficient conditions for guaranteeing Condition (A2). This causes loss of completeness, but it preserves soundness.

4 Checking Inductiveness for Continuous Dynamics

The procedure for solving the switching logic synthesis problem was described at a semantic level in the previous section. Before that procedure can be concretely implemented, we need to find ways to decide Condition (A2). Condition (A2) is not easy to check because F_i's are solutions of differential equations and we usually do not have access to them. In this section, we present methods to check Condition (A2) without using the solution F_i of the differential equations.

Ignoring the multiple modes for now, in order to check Condition (A2), we need the ability to perform the following critical test: for a given set IndInv, determine if all trajectories starting from a (boundary) point of IndInv remain inside the region IndInv for some sufficiently small ϵ time. This is really the "inductiveness" condition for continuous dynamical systems. Formally, we can write it as

$$\forall \mathbf{x} \in \partial\texttt{IndInv} : \exists \epsilon > 0 : F(\mathbf{x}, (0, \epsilon)) \subseteq \texttt{IndInv} \quad (5)$$

where F is the flow defined by some given vector field f.

Let us fix a point $\mathbf{x}$ on the boundary of IndInv. For this point $\mathbf{x}$, we need to determine if

$$F(\mathbf{x}, (0, \epsilon)) \subseteq \texttt{IndInv} \quad (6)$$

where $dX/dt = f(X)$ is some given continuous dynamics whose (unknown) trajectories are described by F, and IndInv is a given target set.

Whereas it is evident from Condition (A2) why deciding Condition (5) is important for switching logic *synthesis*, the reader will also note that deciding Condition (5) is also of fundamental importance for *verification* of continuous dynamical systems.

We will present some effective tests for checking Condition (6) in this section. In Section 4.1, we will present tests that can be peformed locally on the point $\mathbf{x}$. In Section 4.2, we will present a test that needs to be peformed not only at $\mathbf{x}$, but also at all points in an "open region" around $\mathbf{x}$. In Section 5, we will show how we can effectively decide these tests to get an implementation of our synthesis approach.

We first need to fix a representation for IndInv. We use semi-algebraic sets as candidates for $\texttt{IndInv} \subseteq \mathbb{R}^X$. Semi-algebraic sets can be symbolically represented using Boolean combinations of inequalities over polynomial expressions. For simplicity, let us just focus on a single polynomial inequality, namely, $p(X) \geq 0$, where $p(X)$ is a polynomial over the variables X.

The key idea behind checking Condition (5) is to use *Lie* derivatives. Intuitively, we can check that trajectories do not leave $p \geq 0$ by checking that $\frac{dp}{dt}$ is greater-than zero whenever $p = 0$. Technically, the derivative of p with respect to time, $\frac{dp}{dt}$, is called the *Lie derivative*, $L_f(p)$, of p with respect to the vector field f. It can be computed using the chain rule, as shown below. Let X be a vector of n variables, and let $dX/dt = f(X)$ be a system of n differential equations; say, $\frac{dx_1}{dt} = f_1(X), \ldots, \frac{dx_n}{dt} = f_n(X)$. We define the n-th derivative of p with respect to time, $L_f^{(n)}(p)$, as follows:

$$\begin{aligned} \boldsymbol{\nabla} p &:= \left(\frac{\partial p}{\partial x_1}, \frac{\partial p}{\partial x_2}, \ldots, \frac{\partial p}{\partial x_n}\right) \\ L_f(p) &:= \boldsymbol{\nabla} p \cdot f \\ L_f^{(n)}(p) &:= \begin{cases} L_f(p) & \text{if } n = 1 \\ \frac{dL_f^{(n-1)}(p)}{dt} & \text{otherwise} \end{cases} \end{aligned} \quad (7)$$

where $\boldsymbol{\nabla} p \cdot f$ is the dot-product of the two vectors, i.e.,

$$\boldsymbol{\nabla} p \cdot f = \left(\frac{\partial p}{\partial x_1}, \ldots, \frac{\partial p}{\partial x_n}\right) \cdot (f_1, \ldots, f_n) = \sum_{i=1}^{n} \frac{\partial p}{\partial x_i} f_i$$

If each f_i is a polynomial (over the n variables X) and if p is also a polynomial in $\mathbb{Q}[X]$, then Equation (7) shows that $L_f^{(n)}(p)$ is a polynomial in $\mathbb{Q}[X]$ and it can be symbolically computed.

Illustration of Lie derivative computation can be found in Example 3 and Example 4.

4.1 Sufficient Tests for Inductiveness

Assume that `IndInv` is the set (of all points $\mathbf{y}$ where) $p(\mathbf{y}) \geq 0$. We have the following infinite family of candidate tests, parameterized by k, for checking the inclusion in Formula (6)

$$\left(\bigwedge_{i=1}^{k-1} L_f^{(i)}(p)(\mathbf{x}) = 0 \right) \wedge L_f^{(k)}(p)(\mathbf{x}) > 0 \tag{8}$$

In the special case when $k = 1$, we get the following simple test for checking the inclusion in Formula (6)

$$L_f(p)(\mathbf{x}) > 0 \tag{9}$$

For every k, Test (8) is sufficient for proving Condition (6).

Lemma 1. *Let F be a solution of the differential equation $dX/dt = f(X)$. Let $\mathbf{x}$ be any point such that $p(\mathbf{x}) \geq 0$. For any integer $k > 0$, if $L_f^{(i)}(p) = 0$ at $\mathbf{x}$ for $i = 1, \ldots, k-1$ and $L_f^{(k)}(p) > 0$ at $\mathbf{x}$, then there is an $\epsilon > 0$ such that $F(\mathbf{x}, (0, \epsilon)) \subseteq \{\mathbf{y} \mid p(\mathbf{y}) \geq 0\}$. Formally, for each $k > 0$,*

$$\begin{aligned} \forall \mathbf{x} : \; & (\bigwedge_{i=1}^{k-1} L_f^{(i)}(p)(\mathbf{x}) = 0 \wedge L_f^{(k)}(p)(\mathbf{x}) > 0) \\ & \Rightarrow \exists \epsilon > 0 : F(\mathbf{x}, (0, \epsilon)) \subseteq \{\mathbf{y} \mid p(\mathbf{y}) \geq 0\} \end{aligned}$$

In the special case when $k = 1$, we get, for all $\mathbf{x}$,

$$L_f(p)(\mathbf{x}) > 0 \Rightarrow \exists \epsilon > 0 : F(\mathbf{x}, (0, \epsilon)) \subseteq \{\mathbf{y} \mid p(\mathbf{y}) \geq 0\}$$

Proof. Let $\mathbf{x}$ be any point such that $p(\mathbf{x}) \geq 0$. Consider the value of p at the point $F(\mathbf{x}, \epsilon)$, where ϵ is a small positive constant. Let us denote by $p(t)$ the value $p(F(\mathbf{x}, t))$. For t close to 0, by Taylor expansion, we have (note that all derivatives of p with respect to t are defined)

$$\begin{aligned} p(t) &= p(0) + \cdots + \frac{d^i p(0)}{dt^i} \frac{t^i}{i!} + \cdots \\ &= p(0) + \cdots + L_f^{(i)}(p)(\mathbf{x}) \frac{t^i}{i!} + \cdots \end{aligned}$$

For sufficiently small t, the sign of $p(t)$ will be determined by lower-order terms and we can ignore the higher-order terms. If $p(0)$, which is the value of p at $\mathbf{x}$, is strictly positive, then, for sufficiently small t, $p(t)$ remains positive and hence the claim follows. The remaining case is when $p(0) = 0$, which happens when $\mathbf{x}$ is on the surface $p = 0$ of $p \geq 0$. By assumption, the first k terms in the above summation are zero, and the $(k+1)$-st term is positive. Hence, for sufficiently small t, $p(t)$ will be positive and hence the claim follows. □

For every k, we get a sufficient condition based on using Lie derivatives that allows us to verify Condition `(A2)` without requiring the explicit computation of F_i. Each of these tests is sufficient, but none is necessary for establishing that dynamics remain inside the given set for some ϵ time, as shown in the following example.

Example 3 (Not Necessary). Consider the system `CDS` := $(\{x\}, \{x = 0\}, f)$ where $f(x) = 0$ and the set of states $x \geq 0$. Clearly, starting from $x = 0$, the dynamics remains inside the set $x \geq 0$. But, $L_f(x)$ is not strictly greater-than zero at $x = 0$.

$$L_f(x)|_{x=0} = (\partial x/\partial x * dx/dt)|_{x=0} = 1 * f(x)|_{x=0} = 1 * 0 = 0$$

It can be seen that $L_f^{(n)}(x)|_{x=0} = 0$ for all $n \geq 1$ and hence Test (8) fails for all k. □

Since, for each k, Test (8) is a sufficient condition, any disjunction of these conditions is also sufficient; for example,

$$L_f(p)(\mathbf{x}) > 0 \; \vee \; (L_f(p)(\mathbf{x}) = 0 \wedge L_f^{(2)}(p)(\mathbf{x}) > 0) \tag{10}$$

It is tempting to think that replacing $>$ by $\geq$ in one such condition will make the test both necessary and sufficient, but this is not true. The following example shows why, for example, the test $L_f(p) \geq 0$, which is a necessary condition, is *not* sufficient; see also Platzer [18].

Example 4. Consider the system `CDS` := $(\{x\}, \{x = 0\}, f)$ where $f(x) = 1$ and the set of states $-x^2 \geq 0$. Note that

$$L_f(-x^2) = \partial(-x^2)/\partial x * dx/dt = -2x * f(x) = -2x.$$

At $x = 0$, $L_f(-x^2) = 0$, and hence $L_f(-x^2) \geq 0$. But, it is easy to note that starting from $x = 0$ with dynamics $dx/dt = 1$, we do not stay inside the set $-x^2 \geq 0$ for any duration $(0, \epsilon)$. □

Despite being a strong sufficient condition, Test (9) has been used by Prajna, Jadbabaie and Pappas [20] and Taly, Gulwani and Tiwari [24]. The variant of Test (9) where $>$ is replaced by $\geq$, despite being not sufficient in general, is a sufficient check in many special cases, and it has been used in the work by Gulwani and Tiwari [9] and Prajna and Jadbabaie [19] for performing verification of continuous and hybrid systems.

4.2 Another Sufficient Test for Inductiveness

In the previous section, we presented tests that are sufficient for checking if trajectories starting from a point enter a given region (Condition (6)). All these tests were based on computations at the given point. However, the tests were too strong: for example, if the trajectory starting from the point stays on the surface, but does not "enter" the interior of the region, then the tests fail. We now present a different test that also works when the trajectory remains on the surface.

The new test is again based on Lie derivatives. Intuitively, we again check that (whenever the point $\mathbf{x}$ is on the boundary of the region $p \geq 0$) the rate of change of p be non-negative (so that it does not exit the region $p \geq 0$). Formally, we replace $>$ by $\geq$ in Test (9). However, as we saw in Example 4, this modified test does not

guarantee that the trajectory from $\mathbf{x}$ will remain inside $p \geq 0$. The problem was that, intuitively, one expects that the $L_f(p)(\mathbf{x}) = 0$ should hold at point $\mathbf{x}$ only when the vector field at $\mathbf{x}$ is "tangential" to the set $p \geq 0$. Unfortunately, it also holds in some degenerate cases. One such degenerate case is when $\boldsymbol{\nabla} p = 0$, because when $\boldsymbol{\nabla} p(\mathbf{x}) = 0$ then $L_f(p)(\mathbf{x})$ will be zero *irrespective* of the value of f at $\mathbf{x}$.

The new test, shown below as Test (11), explicitly requires that $\boldsymbol{\nabla} p$ be nonzero whenever $L_f(p)$ is zero at any point $\mathbf{x}$. Geometrically, $\boldsymbol{\nabla} p$ is nonzero at a point whenever the surface is "smooth" at that point. The new test is formally written as:

$$L_f(p)(\mathbf{x}) \geq 0 \wedge \boldsymbol{\nabla} p(\mathbf{x}) \neq 0 \tag{11}$$

The additional "smoothness" requirement prevents us from (incorrectly) concluding that the trajectory starting from $x = 0$ under dynamics $dx/dt = 1$ will remain inside $-x^2 \geq 0$.

Example 5. Consider the dynamical system from Example 4. We notice that Test (11) fails at point $x = 0$. In fact if we use $-x^2$ as the value of p, Test (11) reduces to checking if, at $x = 0$, it is the case that $-2x \geq 0 \wedge -2x \neq 0$. This is false and hence Test (11) correctly fails. □

Although it works on the above example, Test (11) is still not sufficient for concluding Condition (6).

Example 6. Consider the dynamics $dx/dt = 1, dy/dt = 0$ and the set $x^2 + y^2 \leq 1$. Consider the point $(0, 1)$. The reader can verify that the trajectory starting from this point will move in the x-direction, and hence it will move "out" of the circle. However, Test (11) evaluates to true at the point $(0, 1)$: at $(0, 1)$, we need to check if $-2x \geq 0 \wedge (-2x, -2y) \neq (0, 0)$, which is true since $0 \geq 0$ and $(0, -2) \neq (0, 0)$. Thus, if we only check for validity of Test (11) at the point $(0, 1)$, then we can incorrectly conclude that the trajectory from $(0, 1)$ will remain inside the circle for some time. □

This problem is fixed by requiring that Test (11) holds *not just* at $\mathbf{x}$, but also at points around $\mathbf{x}$. With this modification, we can now state that the new test implies Condition (6). We need some notation first. We use $|\mathbf{y}|$ to denote the magnitude or length (L_2 norm) of the vector $\mathbf{y}$. We use $B_{\mathbf{x},\epsilon}$ to denote an ϵ-ball around $\mathbf{x}$:

$$B_{\mathbf{x},\epsilon} := \{\mathbf{y} \in \mathbb{R}^n \mid |\mathbf{y} - \mathbf{x}| < \epsilon\}$$

Lemma 2. *Let F be a solution of the differential equation $dX/dt = f(X)$. Let $\texttt{IndInv} := \{\mathbf{y} \mid p(\mathbf{y}) \geq 0\}$. Let $\mathbf{x}$ be a point such that $p(\mathbf{x}) = 0$. If there is a $\delta > 0$ s.t.*

$$\forall \mathbf{y} : (p(\mathbf{y}) = 0 \wedge \mathbf{y} \in B_{\mathbf{x},\delta}) \Rightarrow (L_f(p)(\mathbf{y}) \geq 0 \wedge \boldsymbol{\nabla}(p)(\mathbf{y}) \neq 0),$$

then, there is an $\epsilon > 0$ such that $F(\mathbf{x}, (0, \epsilon)) \subseteq \texttt{IndInv}$.

Proof. Let $D_{\mathbf{x},\delta}$ denote the disk on the surface $p = 0$ around $\mathbf{x}$; formally,

$$D_{\mathbf{x},\delta} := \{\mathbf{y} \mid p(\mathbf{y}) = 0 \wedge \mathbf{y} \in B_{\mathbf{x},\delta}\}$$

We are given that $L_f(p)(\mathbf{y}) \geq 0$ and $\boldsymbol{\nabla}(p)(\mathbf{y}) \neq 0$ for all points $\mathbf{y} \in D_{\mathbf{x},\delta}$.

For $k = 1, 2, 3, \ldots$, define new vector fields f_k as follows:

$$f_k(\mathbf{y}) = f(\mathbf{y}) + \frac{1}{k} \hat{\boldsymbol{\nabla}}(p)(\mathbf{y})$$

where $\hat{\boldsymbol{\nabla}}(p)(\mathbf{y})$ denotes the unit vector in the direction of $\boldsymbol{\nabla}(p)(\mathbf{y})$. Note that the vector fields $f_1, f_2, \ldots$ converge to the vector field f in the limit. Note also that all vector fields are Lipschitz continuous, since we have assumed that f is Lipschitz continuous and since $\boldsymbol{\nabla}(p)$ is Lipschitz continuous (as p is a polynomial).

Let S be the "hemisphere" inside $B_{\mathbf{x},\delta}$ where p is non-negative:

$$S := \{\mathbf{y} \mid p(\mathbf{y}) \geq 0 \wedge \mathbf{y} \in B_{\mathbf{x},\delta}\}$$

For any fixed k and any $\mathbf{y} \in D_{\mathbf{x},\delta}$,

$$\begin{aligned} L_{f_k}(p)(\mathbf{y}) &= L_f(p)(\mathbf{y}) + \frac{1}{k} L_{\hat{\boldsymbol{\nabla}}(p)} p(\mathbf{y}) \\ &\geq 0 + \frac{1}{k} (\boldsymbol{\nabla}(p) \cdot \hat{\boldsymbol{\nabla}}(p))(\mathbf{y}) \ > \ 0 \end{aligned}$$

Hence, using Lemma 1, we conclude that, for each k, the trajectory under f_k starting from $\mathbf{y}$ remains inside S for some nonzero time.

Let t_k and t, respectively, be the time it takes the trajectory starting from $\mathbf{x}$ under dynamics f_k and f, respectively, to exit S. We just showed that $t_k > 0$. We wish to prove that $t > 0$. Let us find a better lower bound for t_k. Since the trajectory can not exit S from any point on D, it needs to reach some other boundary point of S before it can exit S. All other boundary points of S are at least δ distance away from $\mathbf{x}$.

Now we use the crucial assumption that f, and hence each f_k, is Lipschitz. Hence, there is a constant M[1] s.t. for all $\mathbf{y} \in S$, $|f(\mathbf{y})| < M$. Now let $M_k = M + 1/k$. By definition of f_k, it follows that $|f_k(\mathbf{y})| < M_k$ for all $\mathbf{y} \in S$. Hence, $t_k > \delta/M_k$. It then follows that $t_k > \delta/(M + 1/k)$. As k becomes larger, f_k's tend to f and t_k's tend to t (by definition of t_k's). But we just proved a lower bound of δ/M for t and hence $t > 0$. This is exactly what we wanted to prove. □

The fix suggested by Lemma 2 causes the new test to correctly fail on Example 6.

[1] Technically, when we say "Lipschitz", we mean "locally Lipschitz at all points" and hence, M works for only some neighbourhood of $\mathbf{x}$. Hence, we need to intersect the set S with this neighbourhood for the argument to go through. We ignore this technicality here.

Example 7. Reconsider the scenario from Example 6. Recall that Test (11) holds for the point $(0, 1)$. However, it does not hold for points near $(0, 1)$, such as $\mathbf{y} := (\epsilon, \sqrt{(1-\epsilon^2)})$. At this point, Test (11) fails to hold because, while $\boldsymbol{\nabla}(p)$ continues to be nonzero, the Lie derivative $L_f(p)(\mathbf{y}) = -2\epsilon$ is strictly negative. □

The assumption that f is Lipschitz is important for the above lemma, as demonstrated by the following example.

Example 8. Consider the differential equation $dx/dt = 3x^{2/3}$. The vector field $f(x) = 3x^{2/3}$ *is not Lipschitz* at $x = 0$. Suppose we wish to determine if starting from $x = 0$, under the above dynamics, the trajectories remain inside $-x \geq 0$. The set $-x \geq 0$ has only one boundary point $x = 0$ and hence there are no other "nearby" boundary points where we need to check Test (11). Let $p(x) = -x$. At $x = 0$, Test (11) holds because

$$\boldsymbol{\nabla} p|_{x=0} = \frac{\partial(-x)}{\partial x}|_{x=0} = -1|_{x=0} = -1 \neq 0$$
$$L_f(p)|_{x=0} = -x^{2/3}|_{x=0} = 0 \geq 0$$

This seems to suggest that trajectories starting from $x = 0$ will remain inside $-x \geq 0$ for some time. However, this is not true *for all solutions of the differential equation.* Starting from the point $x = 0$, there are two possible trajectories, $x(t) = 0$ and $x(t) = t^3$. The reader can verify that both are solutions of the differential equation. However the trajectory $x(t) = t^3$ exits the region $-x \geq 0$ immediately. □

Note that if the vector field f is specified using polynomials and if `IndInv` is a semi-algebraic set of the form $p \geq 0$, the Lie derivative of p is also a polynomial, and hence all the above the tests are formulas in the first-order theory of the reals, which is decidable [26].

5 Implementing the Procedure

The procedure for solving the switching logic synthesis problem was described at a semantic level in Section 3. In the last section, we presented some sufficient tests that can be used to implement the semantic synthesis procedure. In this section, we present further details on how the semantic synthesis procedure can be concretely implemented.

Recall that the semantic procedure, presented in Figure 2, is based on finding a closed set `CInv` that satisfies Conditions `(A1)`, `(A2)` and `(A3)`. In other words, we need to decide the satisfiability of the following formula:

$$\exists \texttt{CInv} : \forall X : \texttt{(A1)} \wedge \texttt{(A2)} \wedge \texttt{(A3)}$$

Since `CInv` is a set, the above formula has a second-order quantifier. We use the idea of templates to search for `CInv`. A template is a formula (in the theory of reals) with free variables $X \cup U$. Here U are the (real-valued) unknown coefficients that need to be instantiated to yield the desired `CInv`. We use boolean combinations of polynomial equalities and inequalities (semi-algebraic sets) as the formulas. Once a template is fixed, we can write Conditions `(A1)`, `(A2)` and `(A3)` as a *first-order* $\exists\forall$ formula over the theory of reals [9]. Concretely, let $p(U, X)$ be a polynomial and let $p(U, X) \geq 0$ be the chosen template for (searching for) `CInv`. Again, we restrict ourselves to the case of a single inequality $p(U, X) \geq 0$ for simplicity of presentation. For example, $u_1x_1 + u_2x_2 \geq u_3$ is a linear template over 2 variables $X = \{x_1, x_2\}$ and 3 unknown coefficients $U = \{u_1, u_2, u_3\}$. The following formula replaces the existential quantification on `CInv` in the above formula by existential quantification on U and states that there is a choice of values for U such that the resulting set, $p(U, X) \geq 0$, is a controlled invariant sufficient to prove safety.

$$\begin{aligned}\exists U \forall X : &(X \in \texttt{Init} \Rightarrow p(U, X) \geq 0) \ \wedge \\ &(p(U, X) \geq 0 \Rightarrow X \in \texttt{Safe}) \ \wedge \\ &(p(U, X) = 0 \Rightarrow \bigvee_{i \in I} \exists \epsilon > 0 : F_i(X, (0, \epsilon)) \subseteq \texttt{CInv})\end{aligned}$$

The only remaining hurdle in solving the above formula is dealing with F_i. Now we use the results developed in the previous section and replace the reasoning on F_i with reasoning on L_{f_i}.

5.1 The $\exists\forall$ Constraint

We first use the result from Section 4.1. We get the following $\exists\forall$ formula:

$$\begin{aligned}\exists U : \forall X : \ &(X \in \texttt{Init} \Rightarrow p(U, X) \geq 0) \ \wedge \\ &(p(U, X) \geq 0 \Rightarrow X \in \texttt{Safe}) \ \wedge \\ &(p(U, X) = 0 \Rightarrow \bigvee_{i \in I} \mathcal{L}_{f_i} p(U, X) > 0) \quad (12)\end{aligned}$$

If `Init` and `Safe` are semi-algebraic sets, then the membership tests ($X \in$ `Init` and $X \in$ `Safe`) can be written as formulas using polynomials. If each vector field, f_i, is specified using polynomials (i.e., in each mode, $\frac{dX}{dt}$ is a vector of polynomials), then $\mathcal{L}_{f_i} p$ is simply a polynomial. Thus, Formula (12) is a $\exists\forall$ formula consisting only of polynomial expressions.

Corollary 1. *If Formula (12) is valid in the theory of reals, then there is a controlled invariant `CInv` that proves safety.*

Proof. The corollary will follow from Theorem 2 if we show that the condition

$$(p(U, X) = 0 \Rightarrow \bigvee_{i \in I} \mathcal{L}_{f_i} p(U, X) > 0)$$

`SynthSwitchLogicImpl(MDS,Init,Safe)`
`0. Choose a template for the controlled invariant,`
`   say` $p(U, X) \geq 0$
`1. Generate` $\exists\forall$ `constraint for the template to be`
`   a controlled invariant`
$\exists U : \forall X : (X \in \texttt{Init} \Rightarrow p \geq 0) \wedge (p \geq 0 \Rightarrow X \in \texttt{Safe}) \wedge$
$(p = 0 \Rightarrow \bigvee_{i \in I} \mathcal{L}_{f_i} p > 0)$
`1. Solve the` $\exists\forall$ `constraint and get values` $\mathbf{u}$ `for` U
`2. Let` $\texttt{bdry}_i := (p(\mathbf{u}, X) = 0 \ \wedge \ \mathcal{L}_{f_i} p > 0)$ `for all` $i \in I$
`3. Let` $\texttt{Inv}_i := (p(\mathbf{u}, X) \geq 0)$ `for all` $i \in I$
`4. Let` $g_{ij} := \texttt{bdry}_j \ \vee \ (p(\mathbf{u}, X) > 0)$ `for all` $i \neq j; i, j \in I$`,`
`Return SwL` $:= \langle (g_{ij})_{i \neq j; i,j \in I}, (\texttt{Inv}_i)_{i \in I} \rangle$

Fig. 3. A sound procedure for solving the switching logic synthesis problem.

implies Condition (A2). Let $\mathbf{u}$ be the value of the parameters that make Formula (12) true. Let $\mathbf{x}$ be a point on the boundary of the set $\{X \mid p(\mathbf{u}, X) \geq 0\}$. Clearly, $p(\mathbf{u}, \mathbf{x}) = 0$. The above condition immediately implies that the following formula evaluates to true:

$$\bigvee_{i \in I} \mathcal{L}_{f_i} p(\mathbf{u}, \mathbf{x}) > 0$$

From Lemma 1, we know that $\mathcal{L}_{f_i} p(\mathbf{u}, \mathbf{y}) > 0$ implies $\exists \epsilon : F_1(\mathbf{y}, (0, \epsilon)) \subseteq (p \geq 0)$ and we are done. □

Corollary 1 immediately gives us a sound procedure that reduces the switching logic synthesis problem to solving of an $\exists\forall$ constraint in the theory of reals. This procedure is shown in Figure 3. On Lines 2,3 and 4, we extract the guards and the state invariants from the set CInv as we did in Figure 2. We illustrate the procedure on the following example.

Example 9. Consider a train gate controller with two modes: In the *about to lower* mode (1), distance x of the train from the gate decreases according to $\dot{x} = -50$ and the gate angle g does not change. In the *gate lowering* mode (2), we have $\dot{x} = -50$ and $\dot{g} = -10$. The initial state is $g = 90 \wedge x = 1000$. We wish to synthesize the switching logic so that the system always stays in the safe region $x > 0 \vee g \leq 0$. We assume a template of the form $x + a_1 g \geq a_2$ for the controlled invariant. Writing out Formula (12), we get:

$$\begin{aligned}
&\exists a_1, a_2 : \forall x, g : \\
&\quad (x = 1000 \wedge g = 90 \ \Rightarrow \ x + a_1 g \geq a_2) \wedge && \texttt{(A1)} \\
&\quad (x + a_1 g \geq a_2 \ \Rightarrow \ x > 0 \vee g \leq 0) \wedge && \texttt{(A3)} \\
&\quad (x + a_1 g = a_2 \ \Rightarrow \ -50 + 0 > 0 \vee -50 - 10 a_1 > 0) && \texttt{(A2)}
\end{aligned}$$

Our solver returns $a_1 = -10, a_2 = 50$; that is, we get $x - 10g \geq 50$ as the controlled invariant. The resulting hybrid system has $x - 10g \geq 50$ as the state invariant for each mode. The guards for transitions are $g_{12} = x - 10g \geq 50$ (as dynamics for mode 2 points inwards everywhere on the boundary) and $g_{21} = x - 10g > 50$ (dynamics for mode 1 never points inwards on the boundary, so no boundary point gets assigned to g_{21}). So, if the system starts in mode 1, it can continue in 1 until $x - 10g = 50$ is true, whence the system will have to shift to mode 2. The resulting hybrid system is safe and non-blocking. □

5.2 Another $\exists\forall$ Constraint

We use the test in Section 4.2 to get a different $\exists\forall$ constraint for synthesizing controlled invariants. Instead of requiring some vector field to point strictly inwards ($\mathcal{L}_{f_i} p > 0$), we now allow vector fields to be tangential to the controlled invariant ($\mathcal{L}_{f_i} p \geq 0$). However, to be sound, we need the controlled invariant to have a smooth surface (as we noted in Section 4.2). Thus, we have the following weakened version of Formula (12):

$$\begin{aligned}
\exists U \forall X : \ & (X \in \texttt{Init} \Rightarrow p(U, X) \geq 0) \ \wedge \\
& (p(U, X) \geq 0 \Rightarrow X \in \texttt{Safe}) \ \wedge \\
& (p(U, X) = 0 \Rightarrow \bigvee_{i \in I} \mathcal{L}_{f_i} p \geq 0) \ \wedge \\
& (p(U, X) = 0 \Rightarrow \boldsymbol{\nabla} p(U, X) \neq 0) \ \wedge \\
& p(U, X) = 0 \wedge \bigwedge_{i \in I} \mathcal{L}_{f_i} p \leq 0 \Rightarrow \exists! i \in I : \mathcal{L}_{f_i} p = 0
\end{aligned}$$

The notation $\exists!$ denotes the "exists exactly one" quantifier.

Let us analyze the above constraint in more detail. Consider any point $\mathbf{x}$ on the boundary. Suppose no vector field points strictly inwards at $\mathbf{x}$. Therefore, all vector fields point tangential or outwards ($\mathcal{L}_{f_j} p(\mathbf{x}) \leq 0$ for all j). Since our constraint above says that some vector should be tangential or inwards ($\vee_{i \in I} \mathcal{L}_{f_i} p \geq 0$), it follows that there will be a vector field, say f_i, that is tangential at $\mathbf{x}$ ($\mathcal{L}_{f_i} p(\mathbf{x}) = 0$). Since we wish to apply Lemma 2 for proving correctness, we additionally need that *the same vector field*, namely f_i, remain "tangential or inwards" on points on the surface that are in some *neighbourhood* of $\mathbf{x}$. How to guarantee this? *If* all other vector fields point strictly outwards at $\mathbf{x}$ ($\mathcal{L}_{f_j} p(\mathbf{x}) < 0$, for all $j \neq i$), then there will be a *small enough* neighbourhood of $\mathbf{x}$ where these other vector fields will continue to point strictly outwards, and hence, by our requirement that some vector field be "tangential or inwards", it will follow that f_i will remain "tangential or inwards". Thus, the requirement that f_i be the unique vector field that is tangential at $\mathbf{x}$ enables us to prove correctness of the new constraint in Corollary 2 below.

The above formula can be succinctly rewritten as

$$\begin{aligned}
\exists U \forall X : \ & (X \in \texttt{Init} \Rightarrow p(U, X) \geq 0) \ \wedge \\
& (p(U, X) \geq 0 \Rightarrow X \in \texttt{Safe}) \ \wedge \\
& p(U, X) = 0 \Rightarrow (\boldsymbol{\nabla} p(U, X) \neq 0 \ \wedge \\
& \quad (\exists i \in I : \mathcal{L}_{f_i} p > 0 \vee \exists! i \in I : \mathcal{L}_{f_i} p = 0))
\end{aligned} \tag{13}$$

Formula (13) says that at the boundary ($p = 0$) of the controlled invariant ($p \geq 0$), the gradient of p is nonzero

and either some vector field, say f_i, points strictly inwards ($\mathcal{L}_{f_i}p > 0$), or *exactly one* vector field is tangential ($\mathcal{L}_{f_i}p = 0$).

We can now replace the constraint in Step (1) of the procedure in Figure 3 by Formula (13) and get a new and more powerful procedure for solving the switching logic synthesis problem. We can again prove soundness of the technique.

Corollary 2. *If Formula (13) is valid in the theory of reals, then there is a controlled invariant* `CInv` *that proves safety.*

Proof. We only need to argue that the conjunction

$$(p(U,X) = 0 \Rightarrow \boldsymbol{\nabla} p(U,X) \neq 0) \ \wedge$$
$$p(U,X) = 0 \Rightarrow (\exists i \in I : \mathcal{L}_{f_i} p > 0 \vee \exists! i \in I : \mathcal{L}_{f_i} p = 0)$$

implies Condition `(A2)`. Let $\mathbf{u}$ be the value of the parameters that make Formula (13) true. Define `IndInv` to be the set $\{\mathbf{x} \mid p(\mathbf{u},\mathbf{x}) \geq 0\}$. From the formula above, we know that $\boldsymbol{\nabla} p \neq 0$ on all points on the boundary of `IndInv`. Let $\mathbf{y}$ be a point on the boundary of `IndInv`. Clearly, $p(\mathbf{u},\mathbf{y}) = 0$. The above formula also immediately implies that either (i) there is a mode i s.t. $\mathcal{L}_{f_i}p(\mathbf{u},\mathbf{y}) > 0$, or (ii) there is exactly one mode i s.t. $\mathcal{L}_{f_i}p = 0$ and for all modes $j \neq i$, $\mathcal{L}_{f_j}p(\mathbf{u},\mathbf{y}) < 0$.

Now consider the two cases. If some $\mathcal{L}_{f_i}p$ evaluates to a strictly positive real at $\mathbf{y}$, then we are done by Lemma 1. Therefore, assume that for all $i \in I$, $\mathcal{L}_{f_i}p \leq 0$. Case (ii) guarantees that there is an i such that

$$\mathcal{L}_{f_i}p = 0 \wedge \bigwedge_{j \neq i} \mathcal{L}_{f_j}p < 0.$$

Since $\mathcal{L}_{f_j}p < 0$ for all $j \neq i$ at $\mathbf{y}$, it implies that there is a small neighbourhood around $\mathbf{y}$ s.t. $\mathcal{L}_{f_j}p < 0$, for all $j \neq i$, at all points in this small neighbourhood. On all points in this neighbourhood that are also on the surface $p = 0$, by the formula above, we necessarily have $\mathcal{L}_{f_i}p = 0$. Now, using Lemma 2, we conclude that $\exists \epsilon$: $F_i(\mathbf{y},(0,\epsilon)) \subseteq \{\mathbf{x} \mid p(\mathbf{u},\mathbf{x}) \geq 0\}$. □

The following example illustrates the advantage of using Formula (13) over using Formula (12).

Example 10. Consider a system with continuous variable x and y and two modes. In mode 1, $\dot{x} = 0$, $\dot{y} = -1$ and in mode 2, $\dot{x} = -1$, $\dot{y} = 0$. The initial state is $x = 10, y = 10$ and the desired safety property is $y \geq 0$. We start with the template $a_1x + a_2y \geq a_3$. When there are more than one mode, the requirement that $\boldsymbol{\nabla} p \neq 0$ on all boundary points is implied by the other requirements and hence we ignore it. Formula (13) then becomes:

$$\begin{aligned}
&\exists a_1, a_2, a_3 : \forall x, g : \\
&\quad (x = 10 \wedge y = 10 \Rightarrow a_1x + a_2y \geq a_3) \wedge && \texttt{(A1)} \\
&\quad (a_1x + a_2y \geq a_3 \ \Rightarrow y \geq 0) \wedge && \texttt{(A3)} \\
&\quad (a_1x + a_2y = a_3 \ \Rightarrow (-a_1 > 0 \vee -a_2 > 0 \vee && \texttt{(A2)} \\
&\qquad (-a_1 = 0 \wedge -a_2 \neq 0) \vee (-a_2 = 0 \wedge -a_1 \neq 0)))
\end{aligned}$$

We get a solution $a_1 = 0, a_2 = 1, a_3 = 1$. So the invariant obtained is $y \geq 1$. Note that on the boundary of the controlled invariant, the dynamics in mode 2 moves along the boundary and that of mode 1 points outwards. On the same problem and using the same template, if we use Formula (12), the resulting $\exists\forall$ formula is unsatisfiable. Hence, the previous method would fail to find a controlled invariant for this example. □

Example 11. Consider the train gate controller model from Example 9. Observe that the controller synthesized is very conservative and forces the system to switch from mode 1 to 2 in less-than 1 time units. Applying the variant procedure on this example, we get the following $\exists\forall$ formula:

$$\begin{aligned}
&\exists a_1, a_2 : \forall x, g : \\
&\quad (x = 1000 \wedge g = 90 \Rightarrow x + a_1 g \geq a_2) \wedge && \texttt{(A1)} \\
&\quad (x + a_1 g \geq a_2 \Rightarrow x > 0 \vee g \leq 0) \wedge && \texttt{(A3)} \\
&\quad (x + a_1 g = a_2 \Rightarrow -50 > 0 \vee -50 - 10a_1 > 0 \vee \\
&\qquad (-50 = 0 \wedge -50 - 10a_1 < 0) \vee \\
&\qquad (-50 - 10a_1 = 0 \wedge -50 < 0)) && \texttt{(A2)}
\end{aligned}$$

One possible solution now is $a_1 = -5$, $a_2 = 50$, which gives $x - 5g \geq 50$ as the controlled invariant. These values for a_1 and a_2 do not make Formula (12) satisfiable and hence this solution could not have been discovered by the previous method. Corresponding to these values for a_1 and a_2, the resulting hybrid system has $x - 5g \geq 50$ as the state invariant for each mode and the guards are computed as $g_{12} = x - 5g \geq 50$ and $g_{21} = x - 5g > 50$. In this case, the switch from mode 1 to mode 2 could be delayed by as much as 10 time units. □

6 Synthesizing a Good Controller

In the previous section, two sound approaches were presented for solving the switching logic synthesis problem. Neither method gives any guarantee on the quality of the generated controller. A controller that minimally restricts the dynamics – and consequently results in a system with a maximal reach set – is preferable since it provides more opportunities for being refined later for other requirements. In this section, we present heuristics that improve the quality of solution generated by the two approaches presented in Section 5.

The size of the generated controlled invariant is a good measure of the quality of the solution. We desire to synthesize the largest possible inductive controlled invariant `CInv` because this would allow the maximal possible behaviors. It is not immediately clear how this can be achieved in our approach. Intuitively, the problem of finding the largest inductive controlled invariant is naturally seen as an *optimization* problem, whereas in our approach of using constraints, we are casting the problem as a *satisfiability* problem that asks for some solution and not the "best" solution. We now present three different ways to address the above problem.

6.1 Binary Search

The first solution for finding good controllers is based on iteratively searching for larger controlled invariants. In the first iteration, we use one of the methods from Section 5 to compute `CInv`. In each subsequent iteration, we add an additional constraint that forces search for a larger set `CInv`. For example, if we use the template $p(U, X) \geq 0$, and the first iteration returns the controlled invariant $p(\mathbf{u}, X) \geq 0$, then in the next iteration we use the template $p'(v, X) := p(\mathbf{u}, X) \geq v$ (containing only one parameter v) and add an additional constraint $v \leq -1$. If the second iteration is successful, then the controlled invariant generated in the second iteration will necessarily contain the controlled invariant generated in the first iteration. In the case when we know a lower bound on v, say $lb < 0$, then we can search for the optimal v by using a binary search in the interval $[lb, 0]$. This approach can be used to find the largest controlled invariant in the set $\{p(\mathbf{u}, X) \geq v \mid v \in [lb, 0],\ v \text{ an integer}\}$ in $O(\log \|lb\|)$ calls to the procedure in Section 5.

6.2 Encoding Optimality Constraints Directly

We now present a different technique for capturing the optimality requirement. It is based on adding more constraints to the $\exists\forall$ formula. Intuitively, the new constraints say that at least one of the implications in the $\exists\forall$ formula is tight.

A reasonable heuristic for identifying if `CInv` is maximally large is to test if the boundary of `CInv` touches the boundary of the unsafe set $\overline{\texttt{Safe}}$. Hence, we introduce the following additional constraint in the original $\exists\forall$ formula:

$$\partial\texttt{CInv} \cap \partial\texttt{Cl}(\overline{\texttt{Safe}}) \neq \emptyset$$

This constraint can be written as an $\exists$ formula. Since we assume the sets `CInv` and `Safe` are given using polynomial inequalities, the boundaries of these sets can be expressed using polynomial equations and inequalities. The above constraint corresponds to tightening Condition `(A3)`.

Example 12. Consider the train gate controller from example 11. The controlled invariant obtained by using the variant procedure on this example is $x - 5g \geq 50$. Observe that this is not the largest controlled invariant possible because when $x = 0$, this invariant implies $g \leq -10$, whereas safety just requires $g \leq 0$. If we add an additional constraint for tightening condition `(A3)`, which in this case is $\exists x_1, g_1 : x_1 + a_1 g = a_2 \wedge x = 0 \Rightarrow g = 0$, to the $\exists\forall$ formula, we get $x - 5g \geq 0$ as the controlled invariant. This is the largest controlled invariant for the template $x - 5g \geq v$. □

Rather than tightening Condition `(A3)`, we can alternatively tighten Condition `(A2)`. However, we can not do that in general, but only for a specific SMT-based backend $\exists\forall$ solver [9]. The SMT-based $\exists\forall$ solver works in two steps. In the first step, the $\forall$ quantifier is eliminated and replaced by new $\exists$ quantifiers. The result of the first step is a purely existentially quantified formula which is solved using SMT solvers in the second step. The first step is achieved using a variant of Farkas Lemma – which is a technique for replacing $\forall$ by $\exists$ quantification.

Lemma 3. *It is the case that Formula (14) is implied by Formula (15).*

$$\exists U : \forall X : ((\bigwedge_j p_j = 0) \wedge (\bigwedge_k q_k > 0) \Rightarrow p \geq 0) \quad (14)$$

$$\exists U, \nu_j, \lambda_k, \lambda, \mu : \lambda_k \geq 0 \ \wedge\ \lambda \geq 0 \ \wedge\ \mu > 0 \ \wedge$$
$$\forall X : \sum_p \nu_j p_j + \sum_k \lambda_k q_k + \lambda - \mu p = 0 \quad (15)$$

Lemma 3 can be used to eliminate the internal $\forall X$ quantifier by noting that the polynomial in Formula (15) is zero for all X, if and only if, all coefficients of all power products of X in that polynomial are identically 0. We note that the term λ in Formula (15) is a "slack" term. If Formula (15) is satisfied when $\lambda = 0$, then we say that the implication of Formula (14) is *tight*.

Now consider Condition `(A2)` that encodes the inductiveness condition. In Section 5, this condition was approximately captured in Formula (12) and Formula (13). Using elementary logical manipulations, we can rewrite these formulas in the form

$$\exists U : \bigwedge_i (\forall X : (\bigwedge_j p_{ij} = 0) \wedge (\bigwedge_k q_{ik} > 0) \Rightarrow p_i \geq 0) \quad (16)$$

Apply Lemma 3 to each outer conjunct and let λ_i be the slack term for the i-th conjunct.

Now we are ready to state the constraint that enforces tightness on Condition `(A2)`. This new constraint is not added to the $\exists\forall$ formula. It is added to the existential formula generated after the $\forall$ quantifiers have been eliminated using Lemma 3. The constraint we add is the following:

$$\phi_{opt} := \bigvee_i (\lambda_i = 0) \quad (17)$$

If the existential formula, with ϕ_{opt} added, is satisfiable and we get a controlled invariant $p(\mathbf{u}, X) \geq 0$, then we can show the obtained controlled invariant is the "best possible" among the set $\{p(\mathbf{u}, X) \geq \alpha \mid \alpha \in \mathbb{R}\}$.

Theorem 3 (Correctness). *Let* $\mathbf{u}$ *be a set of values for variables* U *that satisfy the existential formula* $\phi_\exists \wedge \phi_{opt}$*, where* $\phi_\exists$ *is the existential formula generated from Formula (12) (or Formula (13)) using Lemma 3. Then, there is no controlled invariant* $p(\mathbf{u}, X) \geq \alpha$ *for any* $\alpha < 0$ *that also satisfies the existential formula generated from Formula (12) (or Formula (13)) using* $p(\mathbf{u}, X) \geq \alpha$ *as a template.*

The above theorem follows from the definition of the slack terms λ_i's above. Its proof is technical and not very illuminating and hence it is not included here.

7 Extensions and Future Work

In our presentation so far, we have restricted all discussion, for simplicity, to simple templates of the form $p(U, X) \geq 0$. However, the two procedures described in Section 5 can be generalized to the case when the template is a boolean combination of nonstrict polynomial inequalities. When the template is a conjunction, say $p_1 \geq 0 \wedge p_2 \geq 0$, then Formula (12) generalizes to

$$\begin{aligned}
&\exists U \forall X : (X \in \texttt{Init} \Rightarrow p_1 \geq 0 \wedge p_2 \geq 0) \ \wedge \\
&\quad (p_1 \geq 0 \wedge p_2 \geq 0 \Rightarrow X \in \texttt{Safe}) \ \wedge \\
&\quad (p_1 = 0 \wedge p_2 > 0 \Rightarrow \bigvee_{i \in I} \mathcal{L}_{f_i} p_1 > 0) \ \wedge \\
&\quad (p_1 > 0 \wedge p_2 = 0 \Rightarrow \bigvee_{i \in I} \mathcal{L}_{f_i} p_2 > 0) \ \wedge \\
&\quad (p_1 = 0 \wedge p_2 = 0 \Rightarrow \bigvee_{i \in I} \mathcal{L}_{f_i} p_1 > 0 \wedge \mathcal{L}_{f_i} p_2 > 0)
\end{aligned}$$

When the template is a disjunction, say $p_1 \geq 0 \vee p_2 \geq 0$, then Formula (12) generalizes to

$$\begin{aligned}
&\exists U \forall X : (X \in \texttt{Init} \Rightarrow p_1 \geq 0 \vee p_2 \geq 0) \ \wedge \\
&\quad (p_1 \geq 0 \vee p_2 \geq 0 \Rightarrow X \in \texttt{Safe}) \ \wedge \\
&\quad (p_1 = 0 \wedge p_2 < 0 \Rightarrow \bigvee_{i \in I} \mathcal{L}_{f_i} p_1 > 0) \ \wedge \\
&\quad (p_1 < 0 \wedge p_2 = 0 \Rightarrow \bigvee_{i \in I} \mathcal{L}_{f_i} p_2 > 0) \ \wedge \\
&\quad (p_1 = 0 \wedge p_2 = 0 \Rightarrow \bigvee_{i \in I} \mathcal{L}_{f_i} p_1 > 0 \vee \mathcal{L}_{f_i} p_2 > 0)
\end{aligned}$$

We can similarly generalize Formula (13) for the case when the template is a disjunction or conjunction of polynomial inequalities. The following example illustrates this case.

Example 13. Consider a thermostat controller with two continuous variables temperature (t) and power (p) and two modes *on* and *off*. In the *on* mode, the dynamics is $\dot{p} = +1 \wedge \dot{t} = p - 10$ and in the *off* mode, it is $\dot{p} = -1 \wedge \dot{t} = p - 10$. The initial state is $p = 10, t = 75$ and the mode is *on*. The desired safety property is $70 \leq t \leq 80$. We start with the following conjunctive template for the controlled invariant: $a_1 p^2 + a_2 p + a_3 t + a_4 \geq 0 \wedge b_1 p^2 + b_2 p + b_3 t + b_4 \leq 0$. Using the generalization of Formula (13) to conjunctive templates, we get a $\exists\forall$ formula. Solving this formula, we get $a_1 = -1, a_2 = 20, a_3 = 2, a_4 = -172, b_1 = 1, b_2 = -20, b_3 = 100, b_4 = 23$ as one possible solution. This gives the invariant $\frac{-(p-10)^2}{2} + t \geq 72 \wedge \frac{(p-10)^2}{2} + t \leq 77$. The switching conditions can be obtained from the controlled invariant by using the procedure `SynthSwitchLogic`. It is easy to see that this is a safe controller. However it is not the most liberal controller. If we add an additional constraint to tighten the Condition `(A3)` (make the controlled invariant touch the boundary of the unsafe set), then we obtain $\frac{-(p-10)^2}{2} + t \geq 70 \wedge \frac{(p-10)^2}{2} + t \leq 80$ as the controlled invariant. In fact, this is the most liberal controller for this system. □

Our basic approach can be adapted to handle natural variants of the switching logic synthesis problem. First, note that we have assumed that each mode of the multi-modal system has the complete state space as its given state invariant. If the given modes have nontrivial state invariants, we can use them in our constraints and the synthesized controller can potentially refine them. Second, our synthesized controller could have zeno behaviors. It appears that making the constraints stronger (as done in Section 5) already reduces the possibility of synthesizing zeno hybrid systems. This aspect needs further investigation.

Our constraint-based technique relies on the user for picking the template for the controlled invariant. We currently start with linear or quadratic templates that have 1 to 4 conjuncts or disjuncts. It will be interesting to find classes of systems for which a given class of templates is complete.

8 Case Study: Inverted Pendulum

A classic benchmark example used in the area of control is the inverted pendulum system. In this system, an inverted pendulum is attached to a cart that can be moved along a horizontal track (by applying external force). The goal is to prevent the inverted pendulum from falling and keeping it around its unstable equilibrium position.

The system is nonlinear and can be described using two state variables, $X = [\theta, \omega]$, denoting the angle the pendulum makes with the vertical axis (θ) and its angular velocity (ω). The dynamics are given by the following differential equations:

$$\frac{d\theta}{dt} = \omega$$

$$\frac{d\omega}{dt} = \frac{g \sin(\theta)}{l} + \frac{\cos(\theta)}{l} \frac{F - ml\omega^2 \sin(\theta) + mg \cos(\theta) \sin(\theta)}{M + m - m \cos^2(\theta)}$$

where $l = 0.3$ is the length of the pendulum, $m = 0.5$ is its mass, $g = 10$ is the acceleration due to gravity, $M = 0.5$ is the mass of the cart, and F is the external force on the cart. Let us assume that F is the control input and it can take values from the finite set $\{0, -1.5, 1.5, -2, 2\}$. Thus, we have *five* modes and let $f_i, i = 1, \ldots, 5$ denote the five vector fields. The goal is to control the switching between the five modes (by setting the value of F) so that the system remains inside the safe set $\texttt{Safe} := \{\langle \theta, \omega \rangle \mid -1/20 \leq \theta \leq 1/20\}$. Suppose the set `Init` of initial states contains all states where $\omega = 0$ and $-1/40 \leq \theta \leq 1/40$.

We first note that the differential equations above involve trigonometric functions $(\sin, \cos)$, which our constraint solvers can not handle. Hence, we approximate

these functions by the first term of their Taylor series expansion; so, for θ close to zero, we use the approximations $\sin(\theta) \approx \theta$ and $\cos(\theta) \approx 1$, which gives us

$$\frac{d\omega}{dt} = \frac{g\theta}{l} + \frac{F - ml\omega^2\theta + mg\theta}{lM}$$

Let us pick a quadratic template $\theta^2 + b\omega^2 \leq c$ for the controlled invariant `CInv`. Let $p := (-\theta^2 - b\omega^2 + c)$. Using Test (10) for checking inductiveness, we get the following $\exists\forall$ constraint for this example:

$$\begin{array}{l}\exists b, c : \forall \theta, \omega : \ (\frac{-1}{40} \leq \theta \leq \frac{1}{40} \wedge \omega = 0 \Rightarrow \theta^2 + b\omega^2 \leq c) \wedge \\ \qquad\qquad (\theta^2 + b\omega^2 \leq c \Rightarrow \frac{-1}{20} \leq \theta \leq \frac{1}{20}) \wedge \\ (\theta^2 + b\omega^2 = c \Rightarrow \bigvee_{i=1}^{5}(\mathcal{L}_{f_i} p > 0 \vee \mathcal{L}_{f_i} p = 0 \wedge \mathcal{L}_{f_i}^{(2)} p > 0))\end{array}$$

where $\mathcal{L}_{f_i} p$ is calculated as

$$-2\theta\frac{d\theta}{dt} - 2b\omega\frac{d\omega}{dt} = -2\theta\omega - 2b\omega(\frac{g\theta}{l} + \frac{F_i - ml\omega^2\theta + mg\theta}{lM})$$

where $(F_i)_{i=1,\ldots,5} = \langle 0, -1.5, \ldots, 2\rangle$. After replacing the parameters (l, m, M, g) by their values, the above $\exists\forall$ formula can be solved using a tool for performing quantifier elimination (using cylindrical algebraic decomposition) `QEPCAD` [10] and we get $b = 1/200$ and $c = 1/500$ as a possible solution for the existentially quantified variables.

Using the controlled invariant `CInv` $:= (\theta^2 + 1/200\omega^2 \leq 1/500)$, we can construct a switching logic to preserve safety as shown in Figure 3. Recall that when we are on the boundary of `CInv`, we need to switch to a mode that takes the system back into the set `CInv`, and when we are in the interior, we are free to be in any mode. For one deterministic implementation of the synthesized controller, we simulated the *original* model of the inverted pendulum system (model with trigonometric functions) starting from a particular initial state. For this simulation, Figure 4 shows the angle and (scaled) angular velocity of the pendulum against time.

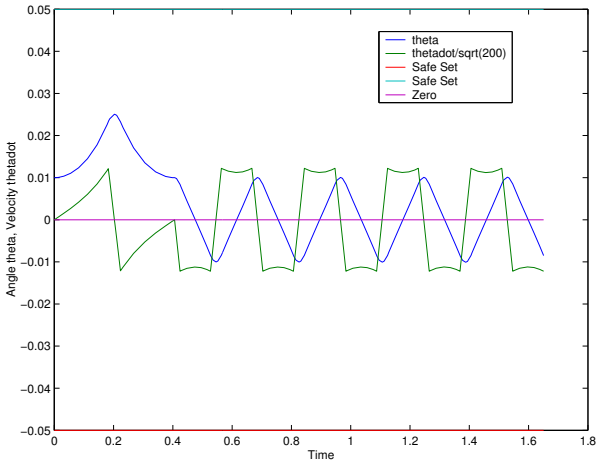

Fig. 4. A simulation of the synthesized inverted pendulum system. Note that the trajectory remains inside the safe set.

9 Related Work

Constraint-based techniques have been used for safety *verification* of hybrid systems [9,21,19], wherein $\exists\forall$ constraints are generated from the user-provided invariant templates. The various approaches differ in the form of the invariants considered, the technique used to generate the $\exists\forall$ formula, and the approach for solving it. In this paper, we present a constraint-based technique for the *synthesis* problem that also involves generating and solving a $\exists\forall$ formula from template controlled invariants. The novelty of our work lies in the formalization of inductive controlled invariant approach for solving synthesis problem and showing that it can be reduced to solving $\exists\forall$ constraints.

There is a lot of work on synthesis of controllers for hybrid systems, which can be broadly classified into two categories. The first category finds controllers that meet some liveness specifications, such as synthesizing a trajectory to drive a hybrid system from an initial state to a desired final state [16,12]. The second category finds controllers that meet some safety specification. Our work falls in this category. For a detailed discussion on the related work in this category, we refer the reader to Asarin et.al. [2]. There are two main approaches for synthesis: direct approaches that compute the controlled reachable states in the style of solving a game [2,27], and abstraction-based approaches that do the same, but on an abstraction or approximation of the system [17,7]. Some of these approaches are limited in the kinds of continuous dynamics they can handle. They all require some form of iterative fixpoint computation. Our work here, based on synthesizing inductive controlled invariants, is an entirely different approach for controller synthesis that does not require any fixpoint computation.

There is a large body of work in the area of program synthesis [15,22,23]. These works differ in the kind of program synthesized and the techniques used. The only work that uses a constraint-based approach is that of Colón, who synthesizes imperative programs computing polynomial functions from partially specified programs and their invariants [6]. The formulation of our "switching logic synthesis" problem is reminiscent of the work on synthesis from component libraries [14] and Sketching [23]. In Sketching, programs are synthesized starting from a template program. Our approach for solving the synthesis problem also bears a high-level resemblance to Sketching, which also follows a constraint-based approach. However, the details are different: our focus is on dynamical systems and switching logics, our property of interest is safety properties, and our constraint encodes existence of an (controlled) inductive invariant. In Sketching, the focus is on programs and instantiating unknown constants in the program sketches, and the constraint does not encode a search for an inductive in-

variant. Recently, the Sketching approach has been extended to perform parameter synthesis for control programs [5] bringing it closer to our application area.

One natural extension of our work is to go beyond safety properties. In fact, the traditional goal for synthesizing switching logics has been to guarantee asymptotic stability [13]. Recently, we have extended our work to also consider dwell time requirements and proposed a different approach – based on simulation and fixpoint computation – for solving the switching logic synthesis problem [11]. Another possible future extension concerns adding performance requirements. There is plenty of work on synthesis for optimizing some function using finite horizon or gradient descent techniques [5].

10 Conclusion

This paper formalized the notion of inductive controlled invariants and showed that inductive controlled invariants can be used to synthesize controllers that satisfy some safety requirements. Theoretically, this approach is sound and complete. We adapted this approach to the problem of synthesizing switching logic for multi-modal systems. We presented several sufficient conditions for a set to be an inductive (controlled) invariant set for a (multi-modal) dynamical system. These sufficient conditions were used to synthesize controllers using template-based techniques, which were then adapted to generate optimal controlled invariants.

References

1. R. Alur, C. Courcoubetis, N. Halbwachs, T. A. Henzinger, P.-H. Ho, X. Nicollin, A. Olivero, J. Sifakis, and S. Yovine. The algorithmic analysis of hybrid systems. *Theoretical Computer Science*, 138(3):3–34, 1995.
2. E. Asarin, O. Bournez, T. Dang, O. Maler, and A. Pnueli. Effective synthesis of switching controllers for linear systems. *Proc. IEEE*, 88(7):1011–1025, 2000.
3. F. Blanchini. Set invariance in control. *Automatica*, 35:1747–1767, 1999.
4. K. Burns and M. Gidea. *Differential Geometry and Topology: With a view to dynamical systems.* Chapman & Hall, 2005.
5. S. Chaudhuri and A. Solar-Lezama. Smooth interpretation. In *ACM Conference on Programming Language Design and Implementation PLDI*, 2010.
6. M. Colón. Schema-guided synthesis of imperative programs by constraint solving. In *LOPSTR*, pages 166–181, 2004.
7. J. Cury, B. Krogh, and T. Niinomi. Supervisory controllers for hybrid systems based on approximating automata. *IEEE Trans. Aut. Control*, 43:564–568, 1998.
8. S. Gulwani, S. Srivastava, and R. Venkatesan. Program analysis as constraint solving. In *Proc. ACM Conf. on Prgm. Lang. Desgn. and Impl. PLDI*, pages 281–292, 2008.
9. S. Gulwani and A. Tiwari. Constraint-based approach for analysis of hybrid systems. In *CAV*, volume 5123 of *LNCS*, pages 190–203. Springer, 2008.
10. H. Hong. Quantifier elimination procedure by cylindrical algebraic decomposition. 1995. Available at `http://www.gwdg.de/~cais/systeme/saclib,http://www.eecis.udel.edu/~saclib/`.
11. S. Jha, S. Gulwani, S. Seshia, and A. Tiwari. Synthesizing switching logic for safety and dwell-time requirements. In *ACM/IEEE Intl. Conf. on Cyber-Physical Systems, ICCPS*, 2010.
12. T. Koo and S. Sastry. Mode switching synthesis for reachability specification. In *Proc. HSCC 2001*, LNCS 2034, pages 333–346, 2001.
13. D. Liberzon and A. S. Morse. Benchmark problems in stability and design of switched systems. *IEEE Control Systems Magazine*, pages 59–70, October 1999.
14. Y. Lustig and M. Vardi. Synthesis from component libraries. In *Proc. FoSSaCS*, pages 395–409, 2009.
15. Z. Manna and R. Waldinger. A deductive approach to program synthesis. *ACM TOPLAS*, 2(1):90–121, 1980.
16. P. Manon and C. Valentin-Roubinet. Controller synthesis for hybrid systems with linear vector fields. In *Proc. IEEE Symp. on Intell. Control*, pages 17–22, 1999.
17. T. Moor and J. Raisch. Discrete control of switched linear systems. In *Proc. Eur. Control Conf. ECC'99*, 1999.
18. André Platzer. Differential-algebraic dynamic logic for differential-algebraic programs. *J. Log. Comput.*, 20(1):309–352, 2010. Advance Access published on November 18, 2008.
19. S. Prajna and A. Jadbabaie. Safety verification of hybrid systems using barrier certificates. In *HSCC*, volume 2993 of *LNCS*, pages 477–492, 2004.
20. S. Prajna, A. Jadbabaie, and G. Pappas. A framework for worst-case and stochastic safety verification using barrier certificates. *IEEE Trans Aut Control*, 52(8), 2007.
21. S. Sankaranarayanan, H. Sipma, and Z. Manna. Constructing invariants for hybrid systems. In *HSCC*, volume 2993 of *LNCS*, pages 539–554, 2004.
22. Ehud Y. Shapiro. *Algorithmic Program DeBugging.* MIT Press, Cambridge, MA, USA, 1983.
23. Armando Solar-Lezama, Liviu Tancau, Rastislav Bodík, Sanjit Seshia, and Vijay Saraswat. Combinatorial sketching for finite programs. In *ASPLOS*, 2006.
24. A. Taly, S. Gulwani, and A. Tiwari. Synthesizing switching logic using constraint solving. In *Proc. 10th Intl. Conf. on Verification, Model Checking and Abstract Interpretation, VMCAI*, volume 5403 of *LNCS*, pages 305–319. Springer, 2009.
25. A. Taly and A. Tiwari. Deductive verification of continuous dynamical systems. In *IARCS Annual Conf. on Foundations of Software Technology and Theoretical Computer Science (FSTTCS 2009)*, volume 4 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 383–394. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2009.
26. A. Tarski. *A Decision Method for Elementary Algebra and Geometry.* University of California Press, 1948. Second edition.
27. C. Tomlin, L. Lygeros, and S. Sastry. A game-theoretic approach to controller design for hybrid systems. *Proc. of the IEEE*, 88(7):949–970, 2000.