Switching Logic Synthesis for Reachability[*]

Ankur Taly
Computer Science Dept., Stanford University
ataly@stanford.edu

Ashish Tiwari
SRI International, Menlo Park, CA 94025
tiwari@csl.sri.com

ABSTRACT

We consider the problem of driving a system from some initial configuration to a desired configuration while avoiding some unsafe configurations. The system to be controlled is a dynamical system that can operate in different modes. The goal is to synthesize the logic for switching between the modes so that the desired reachability property holds.

In this paper, we first present a sound and complete inference rule for proving reachability properties of single mode continuous dynamical systems. Next, we present an inference rule for proving controlled reachability in multi-modal continuous dynamical systems. From a constructive proof of controlled reachability, we show how to synthesize the desired switching logic. We show that our synthesis procedure is sound and produces only non-zeno hybrid systems.

In practice, we perform a constructive proof of controlled reachability by solving an Exists-Forall formula in the theory of reals. We present an approach for solving such formulas that combines symbolic and numeric solvers. We demonstrate our approach on some examples. All results extend naturally to the case when, instead of reachability, interest is in *until* properties.

Categories and Subject Descriptors

C.3 [**Special-Purpose and Application-Based Systems**]: Real-time and embedded systems; D.2.4 [**Software Engineering**]: Software/Program Verification—*correctness proofs, formal methods*; J.7 [**Computers in Other Systems**]: Process control; I.2.3 [**Artificial Intelligence**]: Deduction and Theorem Proving—*Deduction*

General Terms

Design,Verification

[*]Research supported in part by NSF grants CNS-0720721, CSR-EHCS-0834810 and CSR-0917398 and NASA grant NNX08AB95A.

1. INTRODUCTION

Several physical systems have multiple modes of operation. In each mode, the behavior of the system is different. The dynamics of such a system is described by a *collection* of differential equations – one system of different equations for each mode. We call such systems multimodal dynamical systems. Multimodal dynamical systems are fairly common. For example, whenever a physical device or plant is controlled by actuators that can be in one of finitely many different states, we get a multimodal system. In many multimodal systems, the switch from one mode to another mode is controllable. An important problem then is designing this switching logic – when to switch from one mode of operation to another mode of operation – so as to achieve some desired safety and/or reachability goal.

In this paper, we consider the problem of automatically synthesizing a controller for a multimodal system that will drive the system from an initial state to a desired final state while avoiding unsafe states on the way. In other words, we are interested in *until* properties: all trajectories of the synthesized system should remain in a safe region *until* they reach the desired goal state. We aim to synthesize controllers that are formally correct.

Our approach for solving the synthesis problem is based on generating proofs of "controlled properties" for the multimodal system. We explain this a bit more now. First, consider the verification problem. How do we prove that a system has some safety property? We do so by discovering an appropriate certificate for safety in the form of an inductive invariant (also known as a barrier certificate) [31, 25]. How do we prove that a system has some reachability or stability property? We do so by discovering an appropriate certificate for reachability (stability) in the form of a ranking function (Lyapunov function). Now, consider the synthesis problem. How do we prove that a multimodal system can be controlled to have some safety property? We do so by discovering an appropriate certificate for *controlled safety* in the form of a *controlled inductive invariant* [29]. How do we prove that a multimodal system can be controlled to have some reachability or some until property? We do so by discovering an appropriate certificate for *controlled reachability*, or *controlled until*, property in the form of a *controlled ranking function*, or a combination of *controlled ranking function* and *controlled inductive invariant.* This is the essence of our approach for synthesis.

We are now left with two questions. First, what is the formal definition of a *controlled ranking function*? Second, how do we discover an appropriate controlled ranking function and a controlled inductive invariant for a given multimodal system? We answer these questions in the rest of this paper.

The answer to the first question is presented in the form of a logical (deductive) inference rule. A deductive inference rule states a foundational fact about when a system has a property. For example, consider the following foundational inference rule for safety verification of sequential and concurrent programs [9, 13, 15]:

$$\begin{array}{llrcl}
(1) & \forall \vec{x} & \vec{x} \in \texttt{Init} & \Rightarrow & \vec{x} \in \texttt{Inv} \\
(2) & \forall \vec{x}, \vec{y} & \vec{x} \in \texttt{Inv} \wedge \vec{x} \rightarrow \vec{y} & \Rightarrow & \vec{y} \in \texttt{Inv} \\
(3) & \forall \vec{x} & \vec{x} \in \texttt{Inv} & \Rightarrow & \vec{x} \in \texttt{Safe} \\
\hline
 & & \langle \mathbf{X}, \rightarrow, \texttt{Init} \rangle & \models & G(\texttt{Safe})
\end{array}$$

It can be read as follows: if there is a set Inv such that
(1) all initial states are contained in Inv,
(2) all successors $\vec{y}$ of a state $\vec{x} \in$ Inv are also in Inv, and
(3) every state $\vec{x}$ in Inv is contained in the set Safe,
then the discrete system $\langle \mathbf{X}, \rightarrow, \texttt{Init} \rangle$ with state space $\mathbf{X}$, transition relation $\rightarrow$ and initial states Init is safe; that is, all reachable states are contained in the set Safe. The inference rule above says that an inductive invariant (formally defined as any set Inv that satisfies Conditions (1), (2) and (3)) is a certificate for safety verification.

When we wish to synthesize, rather than verify, we need to prove that the (partial) system "can have" the desired property, rather than "will have" the desired property. Hence, we need to search for a "controlled inductive invariant", rather than an "inductive invariant". In previous work, we presented a formal definition of inductive invariants for continuous dynamical systems [30] in the form of a deductive inference rule; see also [3, 31, 25]. To keep the paper self contained, we present that inference rule for safety verification of single mode continuous systems in Section 4. In another earlier paper, we extended the inference rule for safety verification to an inference rule for "controlled safety" verification, which defined the concept of a "controlled inductive invariant" for a multimodal system [29]. We used the controlled inductive invariant to synthesize a switching logic that guaranteed safety in [29].

The first main contribution of this paper is a sound and (relatively) complete inference rule for reachability verification of single mode continuous systems (Section 4). That inference rule can be seen as defining the notion of a ranking function for continuous systems. We then build upon this rule to present the second contribution of this paper, namely a sound rule for verifying *controlled reachability in multimodal systems* (Section 5). In fact, the rule verifies *controlled until* properties – a controlled until property holds for a multimodal system if there exists a controller (switching logic for switching between modes) such that the resulting system has the specified *until* property. We also present a procedure to extract the controller from the above proof of controlled reachability/until property (Section 6). An important feature of our rule for controlled reachability (controlled until) verification is that it uses a combination of "controlled bounded-time invariants" and "controlled progress invariants" to support the "controlled ranking function" for controlled reachability verification.

Now we come to the second question: how do we discover the controlled invariants and controlled ranking functions that are required to prove controlled reachability in multimodal systems? Here we use the idea of templates [28, 25, 32, 11]. We assume the user provides the *form* of the controlled invariants and the controlled ranking function. Now the applicability of the inference rule reduces to the satisfiability of an $\exists\forall$ formula. A third contribution of the paper is an outline of a new approach for solving such formulas that combines numeric simulations and symbolic reasoning and works for nonlinear expressions (Section 6.3). We illustrate our overall approach on two examples.

2. MOTIVATING EXAMPLES

2.1 Driving a Robot down an Alley

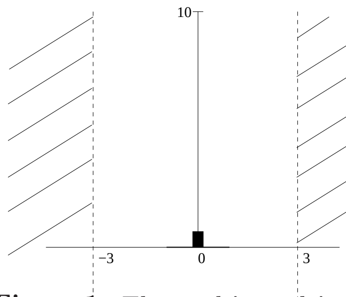

Figure 1: The goal is to drive the robot starting from $x \in [-1, 1], y = 0$ to $y \geq 10$ while avoiding the unsafe region and using the 2 modes.

Consider the problem of steering a robot (car) to keep it on a designated path. The state is described by four continuous variables, x, y, v_x, v_y, where x, y describe the position of the robot on the plane and v_x, v_y describe its velocity in the x- and y-directions. In the scenario in Fig. 1, initially the robot is in the region Init $:= \{(x_0, y_0, v_{x0}, v_{y0}) \mid x_0 \in [-1, 1], y_0 = 0, v_{x0} = v_{y0} = 0\}$ and the goal is to drive it to the region R $:= \{(x, y, v_x, v_y) \mid y \geq 10\}$. Along the way, we wish to avoid the walls Unsafe $:= \{(x, y) \mid x < -3 \vee x > 3\}$. Assume that the robot can be in one of two modes, whose dynamics are:

$$\frac{dx}{dt} = v_x, \quad \frac{dv_x}{dt} = U - v_x, \quad \frac{dy}{dt} = v_y, \quad \frac{dv_y}{dt} = 1 - v_y$$

where $U = 1$ in Mode 1 and $U = -1$ in Mode 2. Note that we are controlling the (x, y) position of the robot by using force, U, that affects the acceleration. The problem is to find the correct switching conditions – the conditions for switching from one mode to the other – that will ensure that the robot reaches the region R while avoiding the region Unsafe.

2.2 Inverted Pendulum

A classic problem in control pertains to maintaining an inverted pendulum around its unstable equilibrium. The state of the inverted pendulum can be described using two continuous variables, x and y, where x denotes the deviation of the pendulum from its unstable equilibrium point $x = 0$ and y denotes the rate of change of x. The dynamics of the inverted pendulum are given by[1]

$$\frac{dx}{dt} = y, \ \frac{dy}{dt} = 20x - 16y + 4u$$

where $u \in \{-16, 16\}$ is the force we can apply to control the inverted pendulum. Thus, again we have two modes. Assume that initially the pendulum is in the state $x = 2, y = 0$. The goal is to design a controller that will take the system to the region $-1 \leq x \leq 1$ and keep it there. Again, the

[1]Dynamics are linearized about the equilibrium point. Our technique also works for a polynomial nonlinear model, but we limit the illustration in this paper to the linearized model.

controller takes the form of a switching logic – the conditions for switching between the two modes that would guarantee the pendulum eventually reaches and thereafter stays in the region $-1 \leq x \leq 1$.

Our goal in this paper is to present an approach for solving such switching logic synthesis problems.

3. CONTINUOUS DYNAMICAL SYSTEMS

In this section, we define a continuous dynamical system (CDS, in short) and then we define when such a continuous system is said to satisfy a safety or reachability property.

DEFINITION 1 (CONTINUOUS DYNAMICAL SYSTEM). *A continuous dynamical system* $\texttt{CDS}$ *is a tuple* $(\texttt{X}, \texttt{Init}, f)$ *where* $\texttt{X}$ *is a finite set of variables interpreted over the reals* $\mathbb{R}$, $\mathbf{X} = \mathbb{R}^{\texttt{X}}$ *is the set of all valuations of the variables* $\texttt{X}$, $\texttt{Init} \subseteq \mathbf{X}$ *is the set of initial states, and* $f : \mathbf{X} \mapsto \mathbf{X}$ *is a vector field that specifies the continuous dynamics.*

Note that $\mathbb{R}^{\texttt{X}}$ is isomorphic to the n-dimensional real space $\mathbb{R}^n$ where $n = |\texttt{X}|$ is the number of variables in $\texttt{X}$. Note also that the continuous dynamical systems we consider here are autonomous, that is, they have no inputs. We assume that f is locally Lipshitz continuous everywhere on $\mathbf{X}$, which guarantees that the ordinary differential equations $\frac{d\texttt{X}}{dt} = f(\texttt{X})$ have a unique solution for every initial condition [5].

The meaning of a continuous dynamical system is simply the collection of all possible trajectories starting from an initial state and “flowing along” the vector field. Formally, if $F(\vec{\texttt{x}}_0, t)$ is the solution of $\frac{d\texttt{X}(t)}{dt} = f(\texttt{X}(t))$, $\texttt{X}(0) = \vec{\texttt{x}}_0$, then the semantics, $[[\texttt{CDS}]]$, of a continuous dynamical system $\texttt{CDS} = (\texttt{X}, \texttt{Init}, f)$ is given as

$$[[\texttt{CDS}]] := \{F_1 : [0, \infty) \mapsto \mathbf{X} \mid F_1(t) = F(\vec{\texttt{x}}_0, t), \quad \vec{\texttt{x}}_0 \in \texttt{Init}\}$$

The above semantics using flow functions is broadly referred to as the *flow semantics* [35]. One can also give a *transition semantics* using discrete state transition systems [12], but the distinction [7] is not relevant for this paper.

The set of reachable states for a continuous dynamical system $\texttt{CDS}$, $\texttt{Reach}(\texttt{CDS})$, is given by $\{\vec{\texttt{x}} \in \mathbf{X} \mid \exists F \in [[\texttt{CDS}]], \exists t \geq 0 : \vec{\texttt{x}} = F(t)\}$. In this paper, we are interested in *until* properties of the form $\texttt{Safe}\,\mathcal{U}\,\texttt{R}$. A CDS has the property $\texttt{Safe}\,\mathcal{U}\,\texttt{R}$ if every initial state can reach some state in $\texttt{R}$ and all intermediate states remain inside $\texttt{Safe}$.

DEFINITION 2 (PROPERTIES). *Given sets* $\texttt{Safe}$ *and* $\texttt{R}$ *of states, a continuous dynamical system* $\texttt{CDS} = (\texttt{X}, \texttt{Init}, f)$ *is said to satisfy the property* $\texttt{Safe}\,\mathcal{U}\,\texttt{R}$*, denoted by* $\texttt{CDS} \models \texttt{Safe}\,\mathcal{U}\,\texttt{R}$*, if*

$$\forall F \in [[\texttt{CDS}]] : \exists t : [F(t) \in \texttt{R} \wedge \forall t' : (0 \leq t' < t \Rightarrow F(t') \in \texttt{Safe})]$$

We use $\mathbf{F}(\texttt{R})$ as a shorthand for $\texttt{true}\,\mathcal{U}\,\texttt{R}$ and $\mathbf{G}(\texttt{Safe})$ for $\neg\mathbf{F}(\neg\texttt{Safe})$. We will use the same notation to denote a set and its characteristic predicate. For example, $\texttt{Init}$ denotes a set of states and $\texttt{Init}(\vec{\texttt{x}})$ denotes $\vec{\texttt{x}} \in \texttt{Init}$.

EXAMPLE 1 (CDS). *Each mode of the robot can be described as a continuous dynamical system* $(\texttt{X}, \texttt{Init}, f)$*, where* $\texttt{X} := \{x, y, v_x, v_y\}$, $\texttt{Init} := \{(x, y, v_x, v_y) \mid x \in [-1, 1], y = v_x = v_y = 0\}$*, and* $f : \mathbb{R}^{\texttt{X}} \mapsto \mathbb{R}^{\texttt{X}}$ *is defined by*

$$f((x, y, v_x, v_y)) = (v_x, v_y, U - v_x, 1 - v_y).$$

In Mode 1, $U = 1$ *and in Mode* 2, $U = -1$. ■

4. VERIFYING SAFETY AND REACHABILITY OF CDS

In this section, we consider the problem of safety and reachability verification of continuous systems. We present sound and complete inference rules (necessary and sufficient characterizations) for safety and reachability verification of continuous systems.

Henceforth, we use $F(\vec{\texttt{x}}, t)$ to denote the unique solution to the initial value problem $\frac{d\texttt{X}(t)}{dt} = f(\texttt{X}(t))$, $\texttt{X}(0) = \vec{\texttt{x}}_0$.

Our inference rules for verification of safety and reachability are based on finding appropriate “witnesses” or “certificates” that are sufficient (and even necessary) for proving safety and reachability respectively. For safety verification, the witness is an inductive invariant and for reachability verification, the witness is a ranking function. (In the same spirit, the witness for stability verification would be a Lyapunov function.)

4.1 Safety Verification of CDSs

Figure 2 presents an inference rule for safety verification of continuous dynamical systems. We can prove a CDS safe by finding a closed set $\texttt{Inv}$ such that (1) all initial states are contained in $\texttt{Inv}$ (Initial), (2) every state in $\texttt{Inv}$ is safe (Property), and (3) at all points $\vec{\texttt{x}}$ on the boundary $\partial\texttt{Inv}$ of $\texttt{Inv}$, the flow F moves *inwards* into the set $\texttt{Inv}$ (Induct). The predicate $\texttt{Inwards}$ is defined below.

DEFINITION 3 (INWARDS). *Given a closed set* $\texttt{Inv}$*, a Lipschitz continuous vector field* $f : \mathbf{X} \mapsto \mathbf{X}$ *and a point* $\vec{\texttt{x}} \in \mathbf{X}$*, the predicate* $\texttt{Inwards}(\texttt{Inv}, \vec{\texttt{x}}, f)$ *holds iff* $\exists h > 0 : \forall 0 \leq t < h : F(\vec{\texttt{x}}, t) \in \texttt{Inv}$.

Intuitively, the predicate $\texttt{Inwards}(\texttt{Inv}, \vec{\texttt{x}}, f)$ is true if, starting from $\vec{\texttt{x}}$, the flow F (corresponding to the vector field f) stays inside the set $\texttt{Inv}$ for *some* $h > 0$ time. If $\vec{\texttt{x}}$ is in the interior of $\texttt{Inv}$, then $\texttt{Inwards}(\texttt{Inv}, \vec{\texttt{x}}, f)$ is always true for any f. However, if $\vec{\texttt{x}}$ is on the boundary of $\texttt{Inv}$, then $\texttt{Inwards}(\texttt{Inv}, \vec{\texttt{x}}, f)$ is true only if the vector field f is pointing “inwards” (into $\texttt{Inv}$) at $\vec{\texttt{x}}$.

The inference rule in Figure 2 is read as follows: if we can find a closed set $\texttt{Inv}$ that satisfies Conditions (Initial), (Induct) and (Property), then we can conclude that $\texttt{CDS}$ is safe. In other words, safety is established by showing the existence of a witness set, namely an inductive invariant $\texttt{Inv}$, which is any set $\texttt{Inv}$ that satisfies Conditions (Initial), (Induct) and (Property). The crucial condition is the inductiveness check (Induct), which is based on checking that the vector field points inwards at all points on the *boundary* $\partial\texttt{Inv}$ of the invariant set $\texttt{Inv}$.

The inference rule in Figure 2 is sound and complete for proving safety, as shown in previous work [3, 30]. As stated, the inference rule in Figure 2 is impractical because the definition of $\texttt{Inwards}$ (Definition 3) uses the solution F of the differential equation. The important point to note here is that there are several efficiently computable sufficient checks for $\texttt{Inwards}$ – based on using the Lie derivatives – some of which are also necessary [30]; see also Section 6.2. We will use one such “implementation” for $\texttt{Inwards}$ in the examples.

4.2 Reachability Verification of CDSs

We extend our earlier work for safety [11, 29, 30], by considering reachability. In this section, we consider the reachability verification problem for CDSs.

$$\begin{array}{lcc}
\text{(Initial)} & \mathtt{Init}(\vec{\mathtt{x}}) \Rightarrow \mathtt{Inv}(\vec{\mathtt{x}}) & \mathtt{Init}(\vec{\mathtt{x}}) \Rightarrow V(\vec{\mathtt{x}}) \leq 0 \ \wedge \ \mathtt{Inv}(\vec{\mathtt{x}}) \\
\text{(Induct)} & \partial\mathtt{Inv}(\vec{\mathtt{x}}) \Rightarrow \mathtt{Inwards}(\mathtt{Inv}, \vec{\mathtt{x}}, f) & \partial\mathtt{Inv}(\vec{\mathtt{x}}) \wedge V(\vec{\mathtt{x}}) < 0 \Rightarrow \mathtt{Inwards}(\mathtt{Inv}, \vec{\mathtt{x}}, f) \\
\text{(Property)} & \mathtt{Inv}(\vec{\mathtt{x}}) \Rightarrow \mathtt{Safe}(\vec{\mathtt{x}}) & \mathtt{Inv}(\vec{\mathtt{x}}) \ \wedge \ V(\vec{\mathtt{x}}) = 0 \Rightarrow \mathtt{R}(\vec{\mathtt{x}}) \\
\text{(Progress)} & & \mathtt{Inv}(\vec{\mathtt{x}}) \wedge V(\vec{\mathtt{x}}) < 0 \Rightarrow \mathtt{Progress}(V, \vec{\mathtt{x}}, f, \epsilon) \\
\hline
 & \mathtt{CDS} \models \mathbf{G}(\mathtt{Safe}) & \mathtt{CDS} \models \mathbf{F}(\mathtt{R})
\end{array}$$

Figure 2: Inference rule for safety and reachability verification of continuous system $\mathtt{CDS} := (\mathtt{X}, \mathtt{Init}, f)$. The inference rule on right actually proves $(V < 0)\,\mathcal{U}\,\mathtt{R}$.

Figure 2(right) presents an inference rule for reachability verification. It uses the `Progress` predicate, apart from the `Inwards` predicate.

DEFINITION 4 (PROGRESS). *Given a state $\vec{x} \in \mathbf{X}$, a Lipschitz continuous vector field $f : \mathbf{X} \mapsto \mathbf{X}$, a positive constant $\epsilon \in \mathbb{R}^+$, and a function $V : \mathbf{X} \mapsto \mathbb{R}$, the predicate $\mathtt{Progress}(V, \vec{x}, f, \epsilon)$ holds if there is a positive $h \in \mathbb{R}^+$ s.t.*

1. $V(F(\vec{x}, h)) - V(\vec{x}) \geq \epsilon \cdot h$
2. $\forall 0 \leq t \leq h : V(F(\vec{x}, t)) \geq V(\vec{x})$
3. $V(F(\vec{x}, t))$ *is continuous over t for all $t \geq 0$.*

Intuitively, the predicate $\mathtt{Progress}(V, \vec{\mathtt{x}}, f, \epsilon)$ is true at a point $\vec{\mathtt{x}}$ for the vector field f if the function V changes continuously along the flow F (defined by the vector field f), V does not drop below its value at $\vec{\mathtt{x}}$ for at least some h time, and after h time it is at least $\epsilon \cdot h$ more than its initial value.

The inference rule in Figure 2 establishes reachability of a set `R` of states by showing the existence of a ranking function V such that (1) V is initially non-positive (Initial), (2) V increases while it is non-positive (Progress), and (3) when V is zero, then it indicates that some state in `R` has been reached (Property).

A notable enhancement in the inference rule for reachability verification in Figure 2 is that reachability verification is supported with "time-bounded invariants" `Inv`. Specifically, in Figure 2(right), the Conditions (Progress) and (Property) are checked *only for points $\vec{x}$ that are also in the set `Inv`*. The set `Inv` is an over-approximation of the set of states that are reached *before* a state in `R` is reached. Note that `Inv` need not be an invariant of `CDS`. Adding the set `Inv` to the inference rule in Figure 2 simplifies the choice of the ranking function V since weaker ranking functions could suffice for proving reachability if stronger `Inv` can be found. This observation was also made by Platzer [21].

We can state and prove the soundness (sufficiency) and completeness (necessity) of the inference rule in Figure 2 for proving reachability, but we omit the proofs here.

The soundness theorem is proved by showing that for any system which satisfies the inference rules, it is never the case that a flow moves out of the set `Inv` before the ranking function V becomes 0. Further, as long as $V < 0$ and the flow is inside `Inv`, the value of the ranking function can increase by at least $\epsilon \cdot h$ during a duration h.

THEOREM 1. *(Soundness) Let $\mathtt{CDS} := (X, \mathtt{Init}, f)$ be a continuous dynamical system and let $R \subseteq \mathbf{X}$ be a set of states. If there is an $\epsilon > 0$, a function $V : \mathbf{X} \mapsto \mathbb{R}$ and a closed set `Inv` that satisfy the conditions (Initial), (Progress), (Induct) and (Property) in Fig. 2, then $\mathtt{CDS} \models \mathbf{F}(R)$.*

The completeness theorem says that if $\mathbf{F}(\mathtt{R})$ holds for a continuous dynamical system `CDS`, then the inference rules in Figure 2 can also prove that $\mathtt{CDS} \models \mathbf{F}(\mathtt{R})$. The completness theorem is proved by showing that for any `CDS` which satisfies $\mathbf{F}(\mathtt{R})$, we can find $\epsilon > 0$, a function $V : \mathbf{X} \mapsto \mathbb{R}$, and a closed set `Inv`, which satisfy conditions (Initial), (Progress), (Induct) and (Property) in Fig. 2. Informally, we take the closure of the complete reach set of `CDS` as the set `Inv` and the negative of *time-to-reach-R* as the value of the function V at any point $\vec{\mathtt{x}}$ that is going to reach `R`. Intuitively, it is clear that, by definition, *time-to-reach-R* will always make progress along a flow.

THEOREM 2. *(Completeness) Let $\mathtt{CDS} := (X, \mathtt{Init}, f)$ be a continuous dynamical system and let $R \subseteq \mathbf{X}$ be a set such that $\mathbf{F}(R)$ holds. Suppose that `Init` is closed. Then there exists an $\epsilon > 0$, a function $V : \mathbf{X} \mapsto \mathbb{R}$, and a closed set `Inv` that satisfy conditions (Initial), (Progress), (Induct) and (Property) in Fig. 2.*

We present a simple example to illustrate the inference rule in Figure 2 for reachability verification.

EXAMPLE 2. *Consider a ball moving with constant speed in the x-direction and falling under gravity in the negative y-direction. We can model this system as the $\mathtt{CDS} := (X = \{x, y, z\}, \mathtt{Init} = \{(0, 10, 0)\}, f)$ where f defines the following dynamics: $\dot{x} = 1, \dot{y} = z, \dot{z} = -10$. Suppose we wish to prove that the set $R = \{(1, y, z) \mid y \geq 5, z \in \Re\}$ is reachable.*

We can apply the rule in Figure 2 to prove that R is reachable from the initial state by choosing $\mathtt{Inv} := (10x^2 + 2y - 20 \geq 0 \wedge 10x + z \geq 0)$, $V := (x - 1)$, and $\epsilon = 1$. We use the following practical checks for the predicates `Inwards` and `Progress` (see also Section 6.2). We check $\mathtt{Inwards}(p \geq 0, \vec{x}, f)$ by checking $\frac{dp}{dt}(\vec{x}) \geq 0$, whenever $\nabla(p)(\vec{x}) \neq 0$. Here $\nabla(p)$ denotes the gradient of p. We check $\mathtt{Progress}(V, \vec{x}, f, \epsilon)$ by checking $\frac{dV}{dt}(\vec{x}) \geq \epsilon$. For these choices, we get the proof obligations shown in Figure 3 by applying the rule in Figure 2. It is easy to see that each formula in Figure 3 is a valid (universally quantified) fact. This proves that the region R is reachable. ■

We remark that the inference rule in Figure 2(right) actually proves that "$V < 0$ holds true until the system reaches `R`" ($(V < 0)\,\mathcal{U}\,\mathtt{R}$), which is stronger than just the reachability claim that "the system eventually reaches `R`" ($\mathbf{F}(\mathtt{R})$).

5. CONTROLLED REACHABILITY IN MDS

In this section, we define a multi-modal system (MDS, in short) and then present a sound inference rule for checking controlled reachability in multi-modal systems.

A multi-modal system has a finite number of different modes and in each mode, it behaves like a different continuous dynamical system.

$$\begin{array}{cc}
\text{(Initial)} & (x=0 \wedge y=10 \wedge z=0) \Rightarrow (x-1 \leq 0 \;\wedge\; 10x^2+2y-20 \geq 0 \wedge 10x+z \geq 0) \\
\text{(Progress)} & (10x^2+2y-20 \geq 0 \wedge 10x+z \geq 0 \;\wedge\; x-1<0) \Rightarrow (1 \geq 1) \\
\text{(Induct)} & (10x^2+2y-20 = 0 \wedge 10x+z \geq 0 \;\wedge\; x-1<0) \Rightarrow (20x+2z \geq 0) \\
\text{(Induct)} & (10x^2+2y-20 \geq 0 \wedge 10x+z = 0 \;\wedge\; x-1<0) \Rightarrow (10-10 \geq 0) \\
\text{(Property)} & (10x^2+2y-20 \geq 0 \wedge 10x+z \geq 0 \;\wedge\; x-1=0) \Rightarrow (x=1 \wedge y \geq 5)
\end{array}$$

Figure 3: Proof obligations arising from Example 2.

Definition 5 (Multi-modal CDS/MDS). *A multi-modal continuous dynamical system, MDS, is a tuple $\langle X, f_1, \ldots, f_k, \mathit{Init}\rangle$, where $\langle X, f_i, \mathit{Init}\rangle$ is a continuous dynamical system (representing the i-th mode). Let $I = \{1, \ldots, k\}$ be the mode indices. Given an initial state $\vec{x}_0 \in \mathit{Init}$, we say that a function $F(\vec{x}_0, t) : \mathbf{X} \times [0, \infty) \rightarrow \mathbf{X}$ is a* trajectory *for MDS, if there is an increasing sequence $0 = t_0 < t_1 < t_2 < \cdots$ (either finite or diverging to ∞) such that*

- *$F(\vec{x}_0, 0) = \vec{x}_0$ and $F(\vec{x}_0, t)$ is continuous over $t \geq 0$, and*
- *for each interval (t_i, t_{i+1}), there is a mode $j \in I$ such that $F(\vec{x}_0, t)$ is smooth and $\frac{dF(\vec{x}_0,t)}{dt}(t') = f_j(F(\vec{x}_0, t'))$ for all t' in the range $t_i < t' < t_{i+1}$.*

The semantics $[[MDS]]$ of MDS is defined as $\{F_1 : [0, \infty) \mapsto \mathbf{X} \mid F_1(t) = F(\vec{x}_0, t)$ for some $\vec{x}_0 \in \mathit{Init}\}$.

Following Definition 5, 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 reach some desired state while maintaining safety. We are interested in designing controllers that can guarantee both safety and reachability properties. In an earlier paper [29], we synthesized controllers that guaranteed safety. Here we consider reachability.

We solve the controller synthesis problem in two steps. In the first step, presented in this section, we consider the controlled-reachability verification problem. In the next section, we will show how to extract a controller and synthesize a hybrid system using the "proof" of controlled-reachability.

A set of states R is controlled reachable if for each initial state, there is *some* flow starting from that initial state that reaches the set R. We define the more general notion of "controlled until" property.

Definition 6 (Controlled Until). *Let MDS be a multi-modal system $\langle X, f_1, \ldots, f_k, \mathit{Init}\rangle$ and let $\mathit{Safe}, R \subset \mathbf{X}$ be sets of states. We say $MDS \models \mathit{Safe}\, \mathcal{U}^c\, R$, if for every $\vec{x}_0 \in \mathit{Init}$, there exists a flow $F \in [[MDS]]$ such that $F(0) = \vec{x}_0$, $F(t_1) \in R$ for some $t_1 \geq 0$ and $F(t) \in \mathit{Safe}$ for all $0 \leq t < t_1$. We say that R is controlled reachable in MDS, denoted by $MDS \models \mathbf{F^c}(R)$, if $MDS \models \mathtt{true}\, \mathcal{U}^c\, R$.*

Note that $\mathtt{MDS} \models \mathbf{F}(\mathtt{R})$ holds when *all* flows starting from every initial state reach R, whereas $\mathtt{MDS} \models \mathbf{F^c}(\mathtt{R})$ holds when, starting from any initial state, *some* flow reaches R.

Figure 4 presents an inference rule for proving controlled reachability for multi-modal systems. To prove controlled reachability, we need to find
(a) an invariant set $\mathtt{Inv}_i(\vec{x})$ for each Mode i,
(b) a progress invariant $\mathtt{PInv}_i(\vec{x}, \overline{\vec{x}})$ for each Mode i,
(c) a function $V(\vec{x})$, and
(d) positive real numbers $(p_i)_{i \in I}$ and ϵ
such that the six conditions in Figure 4 hold. Note that the last five conditions need to be checked for each Mode i. The "progress invariant" predicates $\mathtt{PInv}_i(\vec{x}, \overline{\vec{x}})$ are defined over pairs of states, $\vec{x}$ and $\overline{\vec{x}}$. The progress invariant captures the relationship between the current value $\vec{x}$ of the state variables and their value $\overline{\vec{x}}$ at the last entry into the current Mode i. The progress invariant helps us in proving that every discrete transition makes some progress toward reaching our goal. For improving presentation of the inference rule, we introduce the macros, $\mathtt{Entry}_i$ and $\mathtt{Exit}_i$, that are defined as follows:

$$\begin{array}{rcl}
\mathtt{Exit}_i(\vec{x}) & := & \mathtt{Inv}_i(\vec{x}) \wedge \neg\mathtt{Inwards}(\mathtt{Inv}_i, \vec{x}, f_i) \\
\mathtt{Entry}_i(\vec{x}) & := & \mathtt{Inv}_i(\vec{x}) \wedge \mathtt{Inwards}(\mathtt{Inv}_i, \vec{x}, f_i) \wedge \vee_{j \in I} \mathtt{Exit}_j(\vec{x})
\end{array}$$

Intuitively, $\mathtt{Entry}_i$ is the set of states from where we enter Mode i and $\mathtt{Exit}_i$ is the set of states from where we exit Mode i. Now look back at the rule in Figure 4. In the first reading, the reader may wish to ignore the progress invariants. Roughly speaking, Condition (DProgress) ensures that ϵ progress is made during every discrete transition, Condition (CProgress) ensures that the rate of progress is at least p_i during the stay in Mode i, Condition (Induct) ensures that we stay inside the sets $\mathtt{Inv}_i$'s as long as $V < 0$, and Condition (PInduct) ensures that $\mathtt{PInv}_i(\vec{y}, \vec{x})$ holds for all pairs of states such that $\vec{y}$ is reachable from $\vec{x}$ following dynamics of Mode i.

If we add the following extra check in the antecedent of the rule in Figure 4,

$$\text{(Property')} \qquad \mathtt{Inv}_i(\vec{x}) \wedge V(\vec{x}) < 0 \Rightarrow \mathtt{Safe}(\vec{x}),$$

then we can make the conclusion of the rule stronger to conclude the controlled until property $\mathtt{Safe}\, \mathcal{U}^c\, \mathtt{R}$; that is, from every initial state there is a trajectory that reaches R and that remains inside Safe until it reaches R.

Example 3 (Figure 4). *Consider the MDS from Example 1 that modeled a robot moving down an alley (Section 2.1). We will prove that the property $\mathit{Safe}\, \mathcal{U}^c\, R$, where $\mathit{Safe} := \{(x, y, v_x, v_y) \mid -3 < x < 3\}$ and $R := \{(x, y, v_x, v_y) \mid y \geq 10\}$ using the inference rule in Figure 4. To use the rule, we need a function V, and invariants $\mathit{Inv}_1, \mathit{Inv}_2$ and $\mathit{PInv}_1, \mathit{PInv}_2$ for the two modes. Consider the following choices for V, Inv_i and $P\mathit{Inv}_i$.*

$$\begin{array}{rcl}
V & := & (y + v_y - 11) \\
\mathit{Inv}_1 := \mathit{Inv}_2 := \mathit{Inv} & := & |x + v_x| \leq 2 \wedge |v_x| \leq 1 \wedge 0 \leq v_y \leq 1 \\
\mathit{PInv}_1 & := & x + v_x - y - v_y = \overline{x} + \overline{v_x} - \overline{y} - \overline{v_y} \\
\mathit{PInv}_2 & := & x + v_x + y + v_y = \overline{x} + \overline{v_x} + \overline{y} + \overline{v_y}
\end{array}$$

where $\overline{x}$ in PInv_i denotes the value of x when Mode i was entered. Let $\vec{x}$ denote the tuple x, y, v_x, v_y of variables. The

$$\begin{array}{llcl}
\text{(Initial)} & \texttt{Init}(\vec{x}) & \Rightarrow & V(\vec{x}) \leq 0 \wedge \exists i : (\texttt{Inv}_i(\vec{x}) \wedge \texttt{PInv}_i(\vec{x},\vec{x})) \\
\text{(DProgress)} & \texttt{Entry}_i(\vec{x}) \wedge \texttt{Exit}_i(\vec{y}) \wedge \texttt{PInv}_i(\vec{y},\vec{x}) \wedge V(\vec{x}) < 0 & \Rightarrow & V(\vec{y}) \geq V(\vec{x}) + \epsilon \\
\text{(CProgress)} & \texttt{Inv}_i(\vec{x}) \wedge \neg\texttt{Exit}_i(\vec{x}) \wedge V(\vec{x}) < 0 & \Rightarrow & \texttt{Progress}(V, \vec{x}, f_i, p_i) \\
\text{(Induct)} & \texttt{Exit}_i(\vec{x}) \wedge V(\vec{x}) < 0 & \Rightarrow & \exists j : (\texttt{Entry}_j(\vec{x}) \wedge \texttt{PInv}_i(\vec{x},\vec{x})) \\
\text{(PInduct)} & \texttt{PInv}_i(\vec{y},\vec{x}) \wedge \texttt{Inv}_i(\vec{y}) \wedge V(\vec{y}) < 0 & \Rightarrow & \texttt{Inwards}(\texttt{PInv}_i, \vec{y}, f_i) \\
\text{(Property)} & \texttt{Inv}_i(\vec{x}) \wedge V(\vec{x}) = 0 & \Rightarrow & \texttt{R}(\vec{x}) \\
\hline
 & \texttt{MDS} & \models & \mathbf{F^c}(\texttt{R})
\end{array}$$

Figure 4: Inference rule for controlled-reachability verification of multimodal dynamical system $\texttt{MDS} := (\texttt{X}, \texttt{Init}, f_1, \ldots, f_k)$ and reach set $\texttt{R} \subseteq \mathbf{X}$. Here $\texttt{Exit}_i(\vec{x})$ is defined as $\texttt{Inv}_i(\vec{x}) \wedge \neg\texttt{Inwards}(\texttt{Inv}_i, \vec{x}, f_i)$ and $\texttt{Entry}_i(\vec{x})$ is defined as $\texttt{Inv}_i(\vec{x}) \wedge \texttt{Inwards}(\texttt{Inv}_i, \vec{x}, f_i) \wedge \vee_{j \in I}\texttt{Exit}_j(\vec{x})$. The rule naturally extends to proving $\texttt{Safe}\,\mathcal{U}^c\,\texttt{R}$ too.

values of the predicates $\texttt{Entry}_i$ and $\texttt{Exit}_i$, that are defined using $\texttt{Inv}_i$ and $\texttt{Inwards}$, are computed as:

$$\texttt{Entry}_2 := \texttt{Exit}_1(\vec{x}) \quad := \quad \texttt{Inv}_1(\vec{x}) \wedge (x + v_x = 2)$$
$$\texttt{Entry}_1 := \texttt{Exit}_2(\vec{x}) \quad := \quad \texttt{Inv}_2(\vec{x}) \wedge (x + v_x = -2)$$

To verify $\texttt{Safe}\,\mathcal{U}^c\,\texttt{R}$ for the 2-mode system, we next check the seven conditions in the antecedent by fixing $\epsilon = 1$ and $p_i = 1$. Note that $\texttt{PInv}_i(\vec{x},\vec{x})$ evaluates to `true` *and hence is ignored below. Conditions (Initial) and (Property) expand to the following valid formulas:*

$$-1 \leq x \leq 1 \wedge y = v_x = v_y = 0 \quad \Rightarrow \quad (y + v_y \leq 11) \wedge \texttt{Inv}(\vec{x})$$
$$\texttt{Inv}(\vec{x}) \wedge (y + v_y = 11) \quad \Rightarrow \quad y \geq 10$$

Note that $\frac{dV}{dt} = L_{f_i}(V) = 1$ in both modes and hence Condition (CProgress) is immediately verified as it reduces to the tautology $\texttt{Inv}_i \wedge \neg\texttt{Exit}_i \wedge V < 0 \Rightarrow 1 \geq 1$.

Condition (DProgress) for Mode 1 reduces to:

$$\begin{array}{c}
x + v_x = -2 \wedge |v_x| \leq 1 \wedge 0 \leq v_y \leq 1 \wedge \\
x' + v_x' = 2 \wedge |v_x'| \leq 1 \wedge 0 \leq v_y' \leq 1 \wedge \\
x' + v_x' - y' - v_y' = x + v_x - y - v_y \Rightarrow \quad y' + v_y' - y - v_y \geq 1
\end{array}$$

which is easily verified. Condition (DProgress) for Mode 2 is similarly verified. Condition (Induct) for Mode $i = 1$ reduces to the following tautology:

$$x + v_x = 2 \wedge \texttt{Inv}_1(\vec{x}) \quad \Rightarrow \quad (\texttt{Inv}_2(\vec{x}) \wedge x + v_x = 2)$$

Condition (Induct) for Mode $i = 2$ is similarly verified. Condition (PInduct) also holds in both modes since $\frac{d(x+v_x-y-v_y)}{dt} = 0$ in Mode 1 and $\frac{d(x+v_x+y+v_y)}{dt} = 0$ in Mode 2, and hence $\texttt{Inwards}(\texttt{PInv}_i, \vec{y}, f_i)$ holds for any state $\vec{y}$. We finally verify the extra condition (Property') (works for both modes since they share the regular invariant $\texttt{Inv}$):

$$\texttt{Inv} \wedge (y + v_y < 0) \quad \Rightarrow \quad -3 < x < 3$$

Thus, we conclude that the robot can reach the desired state $\texttt{R}$ while avoiding the unsafe region on the way. ■

If we generalize the definition of $\texttt{Inv}_i$ to include the progress invariants (that is, new $\texttt{Inv}_i$ is $\texttt{PInv}_i \wedge \texttt{Inv}_i$), then the presentation of the rule in Figure 4 gets considerably simplified: all mention of `PInv` are eliminated and Condition (PInduct) is also not required (since Condition (Induct) suffices then).

We can formally state and prove the soundness of the inference rule in Figure 4, but we do not do it here since we will explicitly construct the switching conditions, combine them with the `MDS` to synthesize a hybrid system, and then prove that the hybrid system satisfies $\mathbf{F}(\texttt{R})$ (respectively $\texttt{Safe}\,\mathcal{U}\,\texttt{R}$).

6. CONTROLLER SYNTHESIS

In this section, we formally describe how to use a proof of controlled reachability to synthesize mode-switching conditions. These conditions are composed with the multimodal system to give a hybrid system that satisfies the reachability property.

The mode-switching conditions are specified using a *switching logic.*

Definition 7 (Switching Logic). *A switching logic $\texttt{SwL}$ for a multi-modal dynamical system $\texttt{MDS} := \langle \texttt{X}, (f_i)_{i \in I}, \texttt{Init} \rangle$ is a tuple*

$$\langle B, (g_{ij})_{i \neq j;\, i,j \in I}, (\texttt{Reset}_{ij})_{i \neq j;\, i,j \in I} (\texttt{StateInv}_i)_{i \in I} \rangle$$

where, B is a finite set of Boolean variables that take values in the set $\mathbb{B} := \{\top, \bot\}$, $g_{ij} \subseteq \mathbb{B}^B \times \mathbb{R}^X$ specifies the guard for the discrete transition from mode i to mode j, $\texttt{Reset}_{ij} \subseteq \mathbb{B}^B \times \mathbb{R}^X \times \mathbb{B}^B$ specifies the updates to B during the discrete transition from mode i to mode j, and $\texttt{StateInv}_i \subseteq \mathbb{R}^X$ specifies the state (location) invariants.

A multi-modal system $\texttt{MDS} := \langle \texttt{X}, (f_i)_{i \in I}, \texttt{Init} \rangle$ can be combined with a switching logic $\texttt{SwL} := \langle B, (g_{ij}), (\texttt{Reset}_{ij}),$- $(\texttt{StateInv}_i) \rangle$ to create a hybrid system $\texttt{HS} := \texttt{HS}(\texttt{MDS}, \texttt{SwL})$ in the following natural way:

- the state space of `HS` is $I \times \mathbb{B}^B \times \mathbb{R}^X$,
- the hybrid system `HS` has $|I|$ modes; the vector field in mode $i \in I$ is f_i
- the state invariant in mode i is $\texttt{StateInv}_i$
- the initial states of `HS` is $\{(i, \vec{\texttt{b}}, \vec{\texttt{x}}) \mid \vec{\texttt{x}} \in \texttt{Init}, \vec{\texttt{x}} \in \texttt{StateInv}_i, \vec{\texttt{b}} = \vec{\bot}\}$
- there is a discrete transition from $(i, \vec{\texttt{b}}, \vec{\texttt{x}})$ to $(j, \vec{\texttt{b}}', \vec{\texttt{x}})$ if $(\vec{\texttt{b}}, \vec{\texttt{x}}) \in g_{ij}$ and $(\vec{\texttt{b}}, \vec{\texttt{x}}, \vec{\texttt{b}}') \in \texttt{Reset}_{ij}$

The semantics, $[[\texttt{HS}]]$, of $\texttt{HS} := \texttt{HS}(\texttt{MDS}, \texttt{SwL})$ can be defined as a collection of trajectories, analogously to the definition of $[[\texttt{MDS}]]$ (Definition 5), but with the following two additions: (1) every discrete mode switch in the trajectory should respect the guard and the reset relations (as described above), and (2) if $(i, \vec{\texttt{b}}, \vec{\texttt{x}})$ is ever reached in a trajectory, then $\vec{\texttt{x}}$ should belong to $\texttt{StateInv}_i$. For more formal presentation, see [35, 1]. We can define when a hyrid system satisfies an *until* (safety, reachability) property by generalizing Definition 2.

Hybrid systems can have two kinds of undesirable behaviors. First, they can deadlock. This happens when a hybrid system reaches a point from where there is no further valid trajectory; that is, there is no discrete transition enabled at that point and the continuous dynamics can not keep the trajectory inside $\mathtt{StateInv}_i$. The non-blocking requirement defined below disallows such cases.

Definition 8 (Non-Blocking). *A hybrid system* HS *with modes* I*, vector fields* $(f_i)_{i\in I}$*, guards* $(g_{ij})_{i,j\in I}$*, and state invariants* $(\mathit{StateInv}_i)_{i\in I}$*, is said to be* non-blocking, *if for every mode* $i \in I$ *and any state* $\vec{x}$ *in* $\partial \mathit{StateInv}_i$*,*

$$\exists j \in I : (\mathit{Inwards}(\mathit{StateInv}_j, \vec{x}, f_j) \wedge (i \neq j \Rightarrow \vec{x} \in g_{ij}))$$

The second undesirable behavior happens when the frequency of mode switches becomes unbounded and the trajectory makes infinite switches in finite time (zeno behavior). The Min-Dwell requirement eliminates zeno behaviors.

Definition 9 (Min-Dwell). *A hybrid system* HS *satisfies* Min-Dwell *property if there exists a constant* $h > 0$ *such that for all flows* $F \in [[\mathit{HS}]]$*, if* $t_1, t_2, \cdots$ *are the switching times in the flow* F*, then* $\forall i > 1 : t_i - t_{i-1} \geq h$*.*

We are interested in synthesizing non-blocking hybrid systems that have the Min-Dwell property.

Definition 10 (Switching Logic Synthesis Problem). *Given an* $\mathit{MDS} := \langle X, f_1, f_2, \ldots, f_k, \mathit{Init}\rangle$*, a set* $R \subseteq \mathbb{R}^X$ *to reach and a set* Unsafe *to avoid, the* switching logic synthesis problem *seeks to synthesize a switching logic* SwL *such that* $\mathit{HS}(\mathit{MDS}, \mathit{SwL}) \models \neg \mathit{Unsafe}\ \mathcal{U}\ R$*,* HS *is non-blocking, and* HS *satisfies the Min-Dwell property.*

6.1 The Synthesis Procedure

The switching logic synthesis problem (Definition 10) is solved in two steps. In the first step, the inference rule in Figure 4 is used to solve the controlled-reachability (controlled-until) verification problem. Once we have verified controlled-reachability, in this section, we show how we can use the "proof" to construct a controller (mode switching logic) to reach the desired region.

The extraction of the switching logic from the proof of controlled-reachability is fairly intuitive and is shown in Figure 5. We introduce a new Boolean variable, *done*?, which indicates if we have reached $\mathtt{R}$. Initially, *done*? is $\bot$ and it is set to $\top$ only after $V \geq 0$. Note that, as soon as *done*? becomes $\top$, all transitions are disabled and all state invariants accept any state $\vec{x}$. This is done because, after we have reached $\mathtt{R}$, we need to make sure we remain non-blocking and have min-dwell property. (No safety is enforced after goal is reached.) Ignoring the Boolean component of the state space, the switching logic is easy to understand. The sets $\mathtt{Inv}_i$ define the state invariants. The guards are defined so that we have a transition from mode i to mode j at state $(b, \vec{x})$ only if (1) $\vec{x}$ is in $\mathtt{Inv}_i$ and $\mathtt{Inv}_j$, (2) the vector field f_i at $\vec{x}$ is not pointing inwards toward $\mathtt{Inv}_i$, (3) the vector field f_j at $\vec{x}$ is pointing inwards toward $\mathtt{Inv}_j$, and (4) V at $\vec{x}$ is negative.

The synthesis procedure outlined above correctly solves the switching logic synthesis problem.

Theorem 3. *Given any* $\mathit{MDS} := (\mathit{X}, \mathit{Init}, f_1, \ldots, f_k)$*, for every switching logic* SwL *returned by procedure* $\mathit{SynthSwitchLogic}$*, the hybrid system* $\langle \mathit{MDS}, \mathit{SwL}\rangle$ *is non-blocking, it has the min-dwell property, and it satisfies* $\mathbf{F}(\mathit{R})$*.*

6.2 Sound Approximations for the Semantic Predicates

The verification rules for continuous and multi-modal dynamical systems described in this paper rely on the predicates $\mathtt{Inwards}$ and $\mathtt{Progress}$. These predicates are defined using the flow function F, which is the solution of the differential equation $\dot{\vec{x}} = f(\vec{x})$. In this section, we present alternate sound checks for the predicates $\mathtt{Inwards}$ and $\mathtt{Progress}$, which can be computed automatically for the class of polynomial multimodal systems. A polynomial system is one where the vector field $f(x)$ (for each mode) is expressed using polynomial function from $\mathtt{X} \rightarrow R$.

Sound approximations for $\mathtt{Inwards}(\mathtt{Inv}, \vec{x}, f)$ for semi-algebraic $\mathtt{Inv}$ have been discussed extensively in previous work by the authors [30]. Let $\nabla(p)$ denote the gradient of p; that is, $\nabla p := \langle \frac{\partial p}{\partial x_1}, \frac{\partial p}{\partial x_2}, \ldots, \frac{\partial p}{\partial x_n}\rangle$. Let $L_f(p)$ denote the Lie derivative of p with respect to the dynamics f; that is, $L_f(p) := \nabla p \cdot f$, where $\cdot$ denotes the inner product. Clearly, if we know p and the vector field f, we can symbolically compute ∇p and $L_f(p)$. Now, if p is a polynomial, then $\mathtt{Inwards}(p \geq 0, \vec{x}, f)$ is implied by $L_f(p)(\vec{x}) \geq 0$ whenever $\nabla(p)(\vec{x}) \neq 0$, and it is implied by $L_f(p)(\vec{x}) > 0$ in all cases.

Using Lie derivatives, we can similarly get a sound approximation for $\mathtt{Progress}(V, \vec{x}, f, \epsilon)$ as $L_f(V)(\vec{x}) \geq \epsilon$. If the sets $\mathtt{Inv}_i$ in the inference rules in this paper are closed and bounded, then we can argue that we can replace the check $L_f(V)(\vec{x}) \geq \epsilon$ by the simpler check $L_f(V)(\vec{x}) > 0$ without compromising the soundness of the procedure.

The inference rules in Figures 2 and 4 are based on finding a suitable set(s) $\mathtt{Inv}$ ($\mathtt{Inv}_i$'s), a function V, and some positive constants. We have not discussed how to find these sets and functions. We do so by considering semi-algebraic sets as candidates for $\mathtt{Inv}$ and polynomial functions as candidates for V. As demostrated in [11, 29, 30], we can fix a template for the polynomials, thereby restricting the problem to finding the unknown coefficients of the polynomials. Hence, we get an $\exists\forall$ formula – there exist values for the unknown coefficients such that for all values for the state variables X, the conditions in the inference rules hold. Since the theory of reals admits quantifier elimination, we can decide such formulas and discover the unknown coefficients.

6.3 Discovering $\mathtt{Inv}$'s and V's

We briefly illustrate our approach for discovering the various sets required for proving (controlled) reachability.

First, consider the robot example from Section 2.1. In Example 3, we proved a controlled-until property for this example *assuming* that we were given V, $\mathtt{Inv}_i$'s and $\mathtt{PInv}_i$'s. We can weaken this assumption and assume instead that we are only given the *form* of V, $\mathtt{Inv}_i$'s and $\mathtt{PInv}_i$'s, say,

$$\begin{array}{rcl} V & := & (y + v_y - a) \\ \mathtt{Inv}_1 & := & |x + v_x| \leq b \wedge |v_x| \leq c \wedge d \leq v_y \leq e \\ \mathtt{Inv}_2 & := & \mathtt{Inv}_1 \\ \mathtt{PInv}_1 & := & i(x - \overline{x}) + j(v_x - \overline{v_x}) + k(y - \overline{y}) + l(v_y - \overline{v_y}) \\ \mathtt{PInv}_2 & := & m(x - \overline{x}) + n(v_x - \overline{v_x}) + o(y - \overline{y}) + p(v_y - \overline{v_y}) \end{array}$$

where $a, b, \ldots, p$ are unknown real constants. Plugging in the above forms in the formula in Figure 4 gives a big formula of the form

$$\exists a, \ldots, p : \forall \mathtt{X}, \overline{\mathtt{X}} : \bigwedge_i (\phi_{i1} \Rightarrow \phi_{i2})$$

```
SynthSwitchLogic(MDS, R) :
1.  Find closed sets $(\texttt{Inv}_i)_{i\in I}$, function $V : \mathbf{X} \mapsto \mathbb{R}$, and constants $\epsilon, (p_i)_{i\in I} > 0$
    s.t.  (Initial), (DProgress), (CProgress), (Induct) and (Property) hold
2.  $B := \{done?\}$
3.  $\texttt{StateInv}_i(b, \vec{\mathtt{x}}) := \exists\overline{\vec{\mathtt{x}}} : \texttt{Inv}_i(\vec{\mathtt{x}}, \overline{\vec{\mathtt{x}}}) \vee b = \top$, for all $i \in I$
4.  $\texttt{Exit}_i(\vec{\mathtt{x}}) := \texttt{StateInv}_i(\bot, \vec{\mathtt{x}}) \wedge \neg\texttt{Inwards}(\texttt{Inv}_i, \vec{\mathtt{x}}, f_i)$;
5.  $\texttt{Entry}_i(\vec{\mathtt{x}}) := \texttt{StateInv}_i(\bot, \vec{\mathtt{x}}) \wedge \texttt{Inwards}(\texttt{Inv}_i, \vec{\mathtt{x}}, f_i) \wedge \vee_{j\in I}\texttt{Exit}_j(\vec{\mathtt{x}})$
6.  $g_{ij}(b, \vec{\mathtt{x}}) := b = \bot \wedge \texttt{Exit}_i(\vec{\mathtt{x}}) \wedge \texttt{Entry}_j(\vec{\mathtt{x}}) \wedge V(\vec{\mathtt{x}}) < 0 \;\; \forall i \neq j \in I$;
7.  $g_{ii}(b, \vec{\mathtt{x}}) := b = \bot \wedge V(\vec{\mathtt{x}}) \geq 0$
8.  $\texttt{Reset}_{ij}(b, \vec{\mathtt{x}}, b') := (b = \bot \wedge V(\vec{\mathtt{x}}) \geq 0 \wedge b' = \top)$
Return SwL := $\langle (B, (g_{ij})_{i,j\in I}, (\texttt{Reset}_{ij})_{i,j\in I}, (\texttt{StateInv}_i)_{i\in I}\rangle$
```

Figure 5: Procedure for synthesizing switching logic

In theory, the above formula falls in a decidable class and hence a decision procedure implementation, such as QEPCAD [14] could decide its validity. In practice, QEPCAD fails (runs out of memory) because it cannot handle formulas with more than 6-7 variables (the above formula has 21 variables – 13 existentially quantified and 8 universally quantified, assuming p_i and ϵ are known constants).

We use a combination of numeric solving – in the form of Matlab ODE simulators – and symbolic solving – in the form of QEPCAD – to find values for $a, \ldots, p$; see also [34]. First, we use QEPCAD on each individual conjunct $\forall \mathtt{X}, \overline{\mathtt{X}} : \phi_{i1} \Rightarrow \phi_{i2}$ to eliminate the universally quantified variables and get a constraint purely on the existential variables. QEPCAD is able to handle these subformulas because (i) only a small subset of the variables appear in it, and (ii) the subformula is much smaller. For example, QEPCAD can simplify the subformula corresponding to Condition (Property), namely

$$\forall \mathtt{X} : |x{+}v_x| \leq b \wedge |v_x| \leq c \wedge d \leq v_y \leq e \wedge y{+}v_y < a \Rightarrow -3 < x < 3,$$

to $c + b - 3 \leq 0$. In general, QEPCAD returns a disjunction of conjunction of such (maybe nonlinear) constraints on the existentially quantified variable. Thus, we reduce the problem of solving the original $\exists\forall$ formula to solving an $\exists$ formula. We turn the formula into disjunctive normal form and then solve each existentially quantified conjunction of polynomial inequalities and equations separately.

We solve the $\exists$ formula in two steps. We first obtain an estimate for the values of the existential variables by using a Matlab ODE simulator. We describe the approach using a small example. Suppose we wish to find values for b, c such that $c{+}b{-}3 \leq 0 \wedge b \geq 0 \wedge c \geq 0$. We define a continuous function $p(b, c) = \max(c{+}b{-}3, 0){+}\max(-b, 0){+}\max(-c, 0)$ with the property that p is always non-negative and it is zero for any satisfying assignment. We then define a 2-dimensional CDS whose vector field is the negative of (an approximation of) the gradient of p. In the above example, the vector field at a point (b, c) is given by $(-\frac{p(b+\epsilon,c)+p(b,c)}{\epsilon}, \frac{-p(b,c+\epsilon)+p(b,c)}{\epsilon})$, where ϵ is some small positive constant. We perform multiple simulations of the resulting CDS and find different approximate equilibrium points of this CDS. The equilibrium point that minimizes p is chosen as our first estimate for b, c. We finally use QEPCAD again on the pure existential formula, but after instantiating a few variables by their estimated values (to reduce the number of variables for QEPCAD), to get a final sound answer for the existential variables.

We generated the sets $\texttt{Inv}_i$, $\texttt{PInv}_i$, and V for the robot example shown in Example 3 using the above approach. In Figure 6, we show *one possible* trajectory of the hybrid system (using light-colored line) that is synthesized from the above proof of controlled reachability. Note that the dynamics are not stright lines. We show a second trajectory in the plot too, which is obtained by non-deterministically switching before the $\texttt{Exit}_i$ condition is true, but after ensuring a dwell time of 1 unit in each mode (using dark-colored/blue line).

The same approach was also used on the pendulum example (Section 2.2). We solved the problem of synthesizing switching logic for the pendulum in two steps. In the first step, we showed that a set Inv can be reached using Mode 2 ($u = -16$) of the MDS. In the second step, we designed a controller to keep the MDS in the set Inv. We used the inference rule in Figure 2 for the first step. We used the following templates for V and Inv:

$$\begin{array}{rcl} V(x, y) & := & ax + by + c \\ \texttt{Inv}(x, y) & := & x^2 + dy^2 \leq e \end{array}$$

We thus get an $\exists a, \ldots, e : \forall x, y : \Phi$ formula using the rule in Figure 2(right). The above approach yielded the solution $a = -16, b = -1, c = 10, d = 1/12, e = 4$. This shows that we can use Mode 2 to reach Inv, where

$$\texttt{Inv}(x, y) := (x^2 + y^2/12 \leq 4 \wedge |16x + y| \leq 10)$$

Note that being in Inv implies that $-1 \leq x \leq 1$.

In the second step, we verify that we can use the two modes to stay inside Inv. In other words, we check that Inv is a controlled invariant [29]. This is achieved by checking that on the boundary of Inv, there is a mode whose dynamics is inwards. The resulting formulas are verified using QEPCAD. This concludes the proof that Inv is a controlled invariant. Thus, we have implicitly designed switches that achieve the desired goal of getting the inverted pendulum to be in $-1 \leq x \leq 1$.

Figure 6(right) shows a simulation of the synthesized hybrid system from the initial state. Note that the system reaches the desired region ($-1 \leq x \leq 1$) and stays there.

7. RELATED WORK

Sufficient (and sometimes even necessary) conditions for verifying safety and reachability properties of continuous and hybrid systems have been extensively studied [31, 25, 27, 21, 36]. The inference rule for reachability in Platzer [21] is similar to our inference rule (Figure 2(right)) and also uses time-bounded invariants Inv to support reachability proofs. However, the check for invariance is different and our check

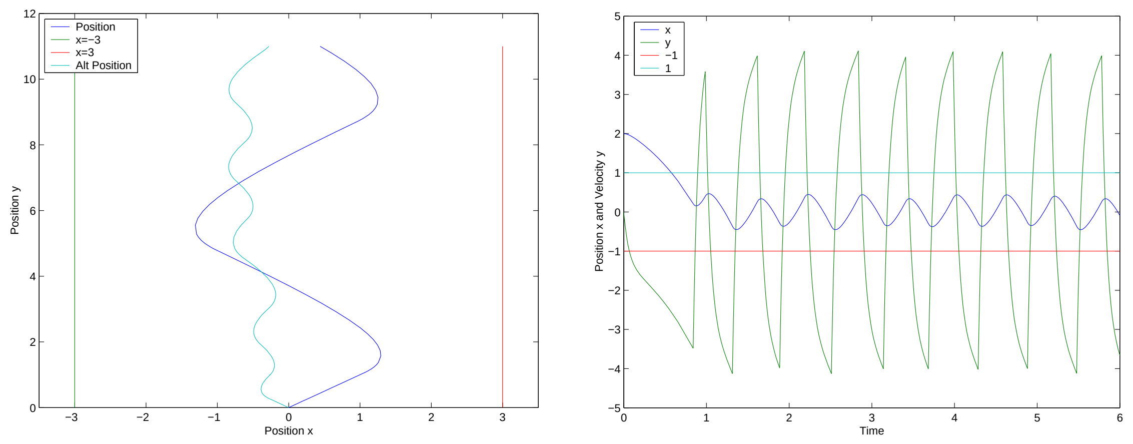

Figure 6: Simulation plots for the two examples. The left plot (y-position versus x-position) shows two trajectories taken by the robot to go from $(0, 0)$ to a destination point in the region $y \geq 10$. The right plot shows the value of the position x (dark/blue) and velocity y (light) of the pendulum (versus time) when it is controlled using the synthesized controller.

is much weaker. We check that vector fields are pointing "inwards" only at the boundary of Inv, whereas [21] checks that at all points. For example, the inference rules in [21] can not prove that $|v_x| \leq 1$ is an invariant under the dynamics $\dot{v_x} = 1 - v_x$ (as in the robot example). Furthermore, we also consider the mode switching logic synthesis problem in this paper. Our inference rule for multimodal and hybrid system works by finding (controlled) invariants and ranking functions that *simultaneously* work for all modes. The inference rules in [21] reason about different modes separately and their application can involve fixpoint iterations.

In previous work [30], we presented sound and relatively complete inference rules for safety verification of continuous systems (Figure 2(left)). In [29], we generalized the safety verification inference rule to an inference rule for controlled safety in multimodal systems and used it for controller synthesis. In this paper, we consider reachability properties (and *until* properties).

The idea of defining "progress" invariant predicates over $\vec{x}, \vec{\overline{x}}$ also appears in the work on discrete systems. Specifically, it has been proposed for verifying liveness properties in the form of ranking functions, transition invariants and progress invariants [23, 22, 6, 24, 10]. However, there are some important distinctions. Transition invariants [23] capture the relationship between the current state and *any* previous state (at a particular program location). Progress invariant in [10] capture the relationship between the current state and the *immediately* previous state (at a particular program location). Our progress invariant captures the relationship between the current state and the *particular* previous state when the current mode was entered. Combining standard invariants with progress invariants appears to be indispensable for proving *until* properties.

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 [17, 16]. The second category finds controllers that meet some safety specification. Our work falls in both categories. For a detailed discussion on the related work in the second 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, 33], and abstraction-based approaches that do the same, but on an abstraction or approximation of the system [18, 8]. 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 time-bounded and progress invariants and ranking functions directly, is an entirely different approach for controller synthesis that does not require any fixpoint computation. We used the same overall approach in our previous work [29] for synthesizing switching logic for purely *safety* requirements, but here we show that the approach also works for *reachability*, and *until*, properties. Moreover, we use a different, and an entirely novel, method to solve the ∃∀ constraints that is more general and scalable than the approach used before [11, 29].

Our approach of using templates to search for certificates (inductive invariants or ranking functions) is closer to the approach used in control for proving stability. Templates are used to search for Lyapunov functions. Due to the way Lyapunov functions are defined, the resulting ∃∀ formula is of a much simpler form, namely $\exists\vec{a}\forall\vec{x}(p > 0 \wedge q < 0)$ (that is, there are no implications (disjunctions) in the formula). Such formulas can be effectively solved using sum-of-squares (SOS) technique based on semidefinite programming and convex optimization [20, 19, 26, 4]. Disjunctions, however, introduce technical difficulties. One way of overcoming them is based on combining simulations with SOS programming [34]; see also Section 6.3.

8. CONCLUSIONS AND FUTURE WORK

We presented inference rules for verification of reachability for continuous systems and controlled reachability, and controlled-until properties, for multi-modal systems. We used the proof of the controlled reachability (controlled-until property) to synthesize a controller for achieving a given reachability (until) specification.

Discovering effective, sound and relatively complete (necessary and sufficient) inference rules for verification of continuous and hybrid systems is challenging. Nevertheless, this is an active area of research; primarily because of its use in verification and synthesis via the bounded verification and bounded synthesis paradigm. In bounded verification and bounded synthesis, templates are used to search for witnesses for proofs. The use of templates leads to $\exists\forall$ formulas. We outlined an approach for solving such formulas that combines numeric and symbolic reasoning and works for nonlinear expressions.

Possible avenues for future work include evaluating the effectiveness of our approach for solving $\exists\forall$ formulas, and automatically discovering the right templates.

Acknowledgements. We thank the referees for their insightful comments.

9. 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):1011–25, 2000.

[3] F. Blanchini. Set invariance in control. *Automatica*, 35:1747–1767, 1999.

[4] S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.

[5] K. Burns and M. Gidea. *Differential Geometry and Topology: With a view to dynamical systems*. Chapman & Hall, 2005.

[6] M. Colon and H. Sipma. Synthesis of linear ranking functions. In *TACAS*, pages 67–81, 2001. LNCS 2031.

[7] P. Cuijpers and M. Reniers. Lost in translation: Hybrid-time flows vs real-time transitions. In *Proc. 11th HSCC*, pages 116–129, 2008. LNCS 4981.

[8] J. Cury, B. Krogh, and T. Niinomi. Supervisory controllers for hybrid systems based on approximating automata. *IEEE Trans. Aut. Ctrl*, 43:564–568, 1998.

[9] R. W. Floyd. Assigning meaning to programs. In *Proc. Symp. in Appl. Math*, 1967.

[10] S. Gulwani, S. Jain, and E. Koskinen. Control-flow refinement and progress invariants for bound analysis. In *PLDI*, 2009.

[11] S. Gulwani and A. Tiwari. Constraint-based approach for analysis of hybrid systems. In *CAV*, volume 5123 of *LNCS*, pages 190–203. Springer, 2008.

[12] T. A. Henzinger. A theory of hyrid automata. In *Proc. 11th IEEE LICS*, pages 278–292, 1996.

[13] C. A. R. Hoare. An axiomatic basis of computer programming. *Comm. ACM*, 12(10):576–580, 1969.

[14] H. Hong and C. Brown. Quantifier elimination procedure by cylindrical algebraic decomposition. `www.usna.edu/Users/cs/qepcad/B/QEPCAD.html`.

[15] R. M. Keller. Formal verification of parallel programs. *Comm. of the ACM*, 19(7), 1976.

[16] T. Koo and S. Sastry. Mode switching synthesis for reachability specification. In *Proc. HSCC 2001*, LNCS 2034, pages 333–346, 2001.

[17] P. Manon and C. Valentin-Roubinet. Controller synthesis for hybrid systems with linear vector fields. In *Proc. IEEE Intl. Symp. on Intelligent Ctrl/Intell. Systems and Semiotics*, pages 17–22, 1999.

[18] T. Moor and J. Raisch. Discrete control of switched linear systems. In *Proc. Eur. Ctrl Conf. ECC'99*, 1999.

[19] P. Parrilo and S. Lall. Sdp relaxations and algebraic optimization in control. *European J of control*, 9(2-3):307–321, 2003.

[20] P. A. Parrilo. *Structured semidefinite programs and semialgebraic geometric methods in robustness and optimization*. PhD thesis, CalTech, 2000.

[21] A. Platzer. Differential dynamic logic for hybrid systems. *J. Autom. Reasoning*, 41(2):143–189, 2008.

[22] A. Podelski and A. Rybalchenko. A complete method for synthesis of linear ranking functions. In *VMCAI*, LNCS. Springer, 2004.

[23] A. Podelski and A. Rybalchenko. Transition invariants. In *LICS*. IEEE Computer Society, 2004.

[24] A. Podelski and S. Wagner. Model checking of hybrid systems: from reachabilty towards stabilty. In *HSCC*, LNCS 3927. Springer, 2006.

[25] S. Prajna and A. Jadbabaie. Safety verification of hybrid systems using barrier certificates. In *HSCC*, volume 2993 of *LNCS*, pages 477–492, 2004.

[26] S. Prajna, A. Papachristodoulou, P. Seiler, and P. A. Parrilo. SOSTOOLS and its control applications. *Positive Polynomials in Control*, pages 273–292, 2005.

[27] S. Prajna and A. Rantzer. Primal-dual tests for safety and reachability. In *HSCC*, volume 3414 of *LNCS*, pages 542–556. Springer, 2005.

[28] S. Sankaranarayanan, H. Sipma, and Z. Manna. Constructing invariants for hybrid systems. In *HSCC*, volume 2993 of *LNCS*, pages 539–554, 2004.

[29] A. Taly, S. Gulwani, and A. Tiwari. Synthesizing switching logic using constraint solving. In *VMCAI*, volume 5403 of *LNCS*, pages 305–319. Springer, 2009.

[30] A. Taly and A. Tiwari. Deductive verification of continuous dynamical systems. In *FST&TCS*, 2009.

[31] A. Tiwari. Approximate reachability for linear systems. In *HSCC*, pages 514–525, 2003. LNCS 2623.

[32] A. Tiwari and G. Khanna. Nonlinear Systems: Approximating reach sets. In *HSCC*, volume 2993 of *LNCS*, pages 600–614. Springer, Mar. 2004.

[33] 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.

[34] U. Topcu, A. Packard, P. Seiler, and T. Wheeler. Stability region analysis using simulations and sum-of-squares programming. In *Proc. American Control Conference, ACC*, pages 6009–6014, 2007.

[35] A. J. van der Schaft and J. M. Schumacher, editors. *An introduction to hybrid dynamical systems*, volume 251 of *Lect. Notes in Ctrl. and Inf. Sci.* Springer, 2000.

[36] T. Wongpiromsarn, S. Mitra, R. M. Murray, and A. G. Lamperski. Periodically controlled hybrid systems. In *HSCC*, volume 5469 of *LNCS*, pages 396–410. Springer, 2009.