Synthesizing Switching Logic Using Constraint Solving*

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

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

Abstract. A new approach based on constraint solving techniques was recently proposed for verification of hybrid systems. This approach works by searching for inductive invariants of a given form. In this paper, we extend that work to automatic synthesis of safe hybrid systems. Starting with a multi-modal dynamical system and a safety property, we present a sound technique for synthesizing a switching logic for changing modes so as to preserve the safety property. By construction, the synthesized hybrid system is well-formed and is guaranteed safe. Our approach is based on synthesizing a controlled invariant that is sufficient to prove safety. The generation of the controlled invariant is cast as a constraint solving problem. When the system, the safety property, and the controlled invariant are all expressed only using polynomials, the generated constraint is an $\exists\forall$ formula in the theory of reals, which we solve using SMT solvers. The generated controlled invariant is then used to arrive at the maximally liberal switching logic.

1 Introduction

Formal verification is beginning to play an important role in the process of building reliable and certifiable complex engineered systems. A different approach to building correct systems is to automatically synthesize safe 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.

Recently, Gulwani and Tiwari [7] introduced an approach for verification of (hybrid) systems that reduces the safety verification problem to satisfiability of $\exists\forall$ formulas over some theory (the theory of reals). Their method is based on finding an inductive invariant that proves the safety of the system. The "unbounded" search for invariants is "bounded" by fixing some templates for the

* Research supported in part by the National Science Foundation under grant CNS-0720721 and by NASA under Grant NNX08AB95A. Work done when the first author was visiting SRI International.

N.D. Jones and M. Müller-Olm (Eds.): VMCAI 2009, LNCS 5403, pp. 305–319, 2009.

invariants. The existence ($\exists$) of an appropriate instance of the template that is also an inductive invariant ($\forall$) naturally maps to an $\exists\forall$ formula. If the $\exists\forall$ formula is valid (over the underlying theory), then it means that there exists an inductive invariant (of the form of the chosen template) that proves safety.

In theory, the constraint-based approach for verification described in Gulwani and Tiwari [7] can be generalized to solving the synthesis problem as well. Given an under-specified system, we can choose templates for the unknown parts of the system and the unknown inductive invariant. We can then obtain a $\exists\forall$ formula where all the unknowns are existentially quantified. The constraint solver then searches for instances of *all* these unknowns 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 [7], to solve the switching logic synthesis problem. The novelty in our approach here is that we do not search for the switching conditions directly. 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.

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.

Contribution and Outline of the Paper. In this paper, we present a formalization of 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 relies on the deductive verification approach and does not use the usual game theoretic approach for controller synthesis, or the controlled reachability approach (See Section 7 for more discussion). We also describe several sufficient conditions for a set to be an inductive controlled invariant set. These conditions enable practical implementations for synthesizing controllers using template-based techniques (Section 4). Finally, we describe some heuristics to generate large controlled invariant sets, that lead to synthesis of the weak controllers (Section 5). We have performed preliminary experimental evaluation of our approach and presented some of the results as examples in the paper. We start by formally describing and motivating the problem in Section 2.

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/states of the system, and its dynamics, which defines how the system changes states (with time). Formally, a *continuous dynamical system* is a tuple $\langle X, f \rangle$ where X is a finite set of real-valued variables that define the state space $\mathbb{R}^X$ and $f : \mathbb{R}^X \mapsto \mathbb{R}^X$ is a vector field that specifies the continuous dynamics (as $\frac{d\boldsymbol{x}}{dt} = f(\boldsymbol{x})$). We assume that f is Lipschitz, which guarantees the existence and uniqueness of solutions to the ordinary differential equations.

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

Often a single ordinary differential equation is insufficient to describe the 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 differential equations. We call such system multi-modal dynamical systems. 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 continuous 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 $\langle X, f_1, f_2, \ldots, f_k, \mathit{Init}\rangle$, where $\langle X, f_i\rangle$ is a continuous dynamical system (representing the i-th mode) and $\mathit{Init} \subseteq \mathbb{R}^X$ is the set of initial states. Given an initial state $\boldsymbol{x_0} \in \mathit{Init}$, we say that a function $\boldsymbol{x}(t) : [0, \infty) \to R^X$ 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*

- *$\boldsymbol{x}(0) = \boldsymbol{x_0}$ and $\boldsymbol{x}(t)$ is continuous over $t \geq 0$, and*
- *for each interval (t_i, t_{i+1}), there is a mode $j \in I$ such that $\boldsymbol{x}(t)$ is smooth and $\frac{d\boldsymbol{x}(t)}{dt}(t') = f_j(\boldsymbol{x}(t'))$ 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.*

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 $\mathit{MDS} := \langle X, (f_i)_{i\in I}, \mathit{Init}\rangle$ is a tuple $\langle (g_{ij})_{i\neq j; i,j\in I}, (\mathit{Inv}_i)_{i\in I}\rangle$, containing guards $g_{ij} \subseteq \mathbb{R}^X$ and state (location) invariants $\mathit{Inv}_i \subseteq \mathbb{R}^X$.*

Informally, the guard g_{ij} specifies the condition under which the system *could* switch from mode i to mode j and the state invariant $\mathtt{Inv}_i$ 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 $\mathtt{HS} := \mathtt{HS}(\mathtt{MDS}, \mathtt{SwL})$ in the following natural way: the hybrid system `HS` has $\|I\|$ modes with dynamics given by $\frac{dX}{dt} = f_i$ in mode i, and with g_{ij} being the guard on the discrete transition from mode i to mode j and $\mathtt{Inv}_i$ being the state invariant in mode i. The initial states are $\{1\} \times \mathtt{Init}$. 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].

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 $\boldsymbol{x}$ on the boundary of $\mathtt{Inv}_i$, but there is no valid trajectory from $\boldsymbol{x}$; that is, there is no discrete transition enabled at $\boldsymbol{x}$, and following mode i dynamics takes the system out of $\mathtt{Inv}_i$. The non-blocking requirement disallows such cases. We are interested in synthesizing non-blocking hybrid systems.

Definition 3. *A hybrid system* $\mathtt{HS}$ *is said to be* non-blocking *if for every mode* i*, and for every point* $\boldsymbol{x}$ *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)* $\boldsymbol{x} \in g_{ij}$ *whenever* $i \neq j$*, 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 $\mathtt{HS}$ is safe with respect to a safety property $\mathtt{Safe} \subseteq \mathbb{R}^X$ if the set of its reachable states is contained in $\mathtt{Safe}$. Formally, we define the logic synthesis problem as follows:

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

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 $\mathtt{MDS}$, its initial set of states by $\mathtt{Init}$ and the given safety property by $\mathtt{Safe}$.

We first define the notion of a controlled invariant set.

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

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

Example 1. Let $\dot{x}$ denote $\frac{dx}{dt}$. Consider a multi-modal system with two modes. In mode 1, $\dot{x} = 1, \dot{y} = 0$, while in mode 2, $\dot{x} = 0, \dot{y} = 1$. If $x = 0, y = 0$ is the only initial state, then $x \geq 0 \wedge y \geq 0$ is an invariant, whereas $x \geq 0 \wedge y = 0$ is a controlled invariant that is not an invariant. The set $x + y \leq 0$ is not 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 Lipshitz and hence, by Proposition 1, we have a unique trajectory $F_i(\boldsymbol{x_0}, t)$ in mode i. By $F_i(\boldsymbol{x_0}, (0, \epsilon))$ we denote the set of all points reached in the time interval $(0, \epsilon)$; that is, $F_i(\boldsymbol{x_0}, (0, \epsilon)) := \{\boldsymbol{x} \mid \boldsymbol{x} = F_i(\boldsymbol{x_0}, t), 0 < t < \epsilon\}$. For a set $S \subseteq \mathbb{R}^n$, let ∂S denotes 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* $\mathtt{CInv}$ *is an* inductive controlled invariant *for* $\mathtt{MDS} := \langle X, (f_i)_{i\in I}, \mathtt{Init}\rangle$ *if*

$$\mathtt{(A1)}\ \ \mathtt{Init} \subseteq \mathtt{CInv} \text{ and}$$

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

```
SynthSwitchLogic(MDS, Safe) :
1. Find a closed set CInv that satisfies Conditions (A1) and (A2)
   from Definition 6 and Condition (A3) below
     (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. 1. 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.*

The complete procedure, at a semantic level, for solving the switching logic synthesis problem is presented in Figure 1. The key idea behind the synthesis procedure is to find an inductive controlled invariant set CInv and then design the guarded transitions so that the resulting hybrid system always remains in CInv. Conditions (A1), (A2), and (A3) imply that CInv is an inductive controlled invariant that proves safety. It follows from the definition of CInv that its boundary ∂CInv can be written as a union

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

such that $\forall \boldsymbol{x} \in \texttt{bdry}_i$, it is the case that $\exists \epsilon > 0 : F_i(\boldsymbol{x}, (0, \epsilon)) \subseteq \texttt{CInv}$. 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 various discrete transitions.

We next state and prove some properties of the procedure SynthSwitchLogic in Figure 1. 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, the hybrid system HS(MDS, SwL) is non-blocking and safe.*

We prove completeness under a technical assumption. We say a hybrid system HS has the min-dwell-time property if there exists a fixed time duration t_a such that for all reachable states $\boldsymbol{x}$, if the hybrid system permits a mode switch from i to j at $\boldsymbol{x}$, then there must exist a mode k such that the hybrid system permits a mode switch from i to k at $\boldsymbol{x}$ and the system can stay in mode k for at least t_a

units of time starting at $\boldsymbol{x}$. The min-dwell-time property implies that successive mode switchings can be forced to be t_a units apart.

Theorem 2 (Completeness). *For any switching logic* `SwL`*, if a hybrid system* $\mathtt{HS} = \mathtt{HS}(\mathtt{MDS}, \mathtt{SwL})$ *is safe,* `HS` *satisfies the min-dwell-time property and* `Safe` *is a closed set, then 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 replace Condition `(A2)` by something stronger that can be easily computed. This causes loss of completeness, but it preserve soundness.

4 Implementing the Procedure

The procedure for solving the switching logic synthesis problem was described at a semantic level in the previous section. In this section, we show how that procedure can be concretely implemented.

Recall that a set `CInv` is an inductive controlled invariant if it satisfies Conditions `(A1)` and `(A2)`. Condition `(A2)` is not easy to check as F_i's are solutions of differential equations. We solve this problem by replacing this condition by a stronger condition, `(B2)`, which, as we show later, can be tested *without* explicitly computing F_i. Let $\mathtt{Interior}(\mathtt{CInv}) := \mathtt{CInv} - \partial\mathtt{CInv}$. We ensure that `CInv` is an inductive controlled invariant (that proves safety) by checking:

`(B1)` $\mathtt{Init} \subseteq \mathtt{CInv}$

`(B2)` $\forall \boldsymbol{x} \in \partial\mathtt{CInv} : \exists i \in I : \exists \epsilon > 0 : F_i(\boldsymbol{x}, (0, \epsilon)) \subseteq \mathtt{Interior}(\mathtt{CInv})$

`(B3)` $\mathtt{CInv} \subseteq \mathtt{Safe}$

We will now present a condition that is equivalent to `(B2)` and that can be easily computed. We first need to fix a representation for `CInv`.

We use semi-algebraic sets as candidates for $\mathtt{CInv} \subseteq \mathbb{R}^X$. Since `CInv` is unknown, we use the idea of templates. 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 `(B1)`, `(B2)` and `(B3)` as an $\exists\forall$ formula over the theory of reals [7]. Concretely, let $p(U, X)$ be a polynomial and $p(U, X) \geq 0$ be the chosen template for searching for `CInv`. 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 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.

$$\exists U : \forall X : \ (X \in \mathtt{Init} \Rightarrow p(U, X) \geq 0) \ \wedge \ (p(U, X) \geq 0 \Rightarrow X \in \mathtt{Safe}) \ \wedge \\ (p(U, X) = 0 \Rightarrow \bigvee_{i \in I} \mathcal{L}_{f_i} p(U, X) > 0) \tag{2}$$

Here $\mathcal{L}_{f_i}p$ denotes the derivative of p with respect to time t and is called the Lie derivative of p with respect to the vector field f_i. It can be symbolically computed using the chain rule as,

$$\mathcal{L}_{f_i}p := \sum_{x \in X} \frac{\partial p}{\partial x}\frac{dx}{dt}.$$

Note that we have used a test on the Lie derivatives to encode Condition `(B2)`. This test is equivalent to `(B2)` and allows us to verify it without requiring F_i.

Remark 1. It is tempting to think that replacing $>$ by $\geq$ in Formula (2) will make the formula equivalent to checking Condition `(A2)`, but this is not true.

If each of the vector fields, 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. If `Init` and `Safe` are semi-algebraic sets, then the membership tests ($X \in$ `Init` and $X \in$ `Safe`) can also be written as formulas using only polynomials. Thus Formula (2) is a $\exists\forall$ formula consisting only of polynomial expressions.

Corollary 1. *If Formula (2) is valid in the theory of reals, then there is a controlled invariant* `CInv` *that proves safety.*

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. We illustrate the procedure on the following example.

Example 2. 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 (2), we get:

$$\begin{array}{ll}
\exists a_1, a_2 : \forall x, g : & \\
\quad (x = 1000 \wedge g = 90 \;\Rightarrow\; x + a_1 g \geq a_2) \wedge & \text{(Condition (B1))} \\
\quad (x + a_1 g \geq a_2 \;\Rightarrow\; x > 0 \vee g \leq 0) \wedge & \text{(Condition (B3))} \\
\quad (x + a_1 g = a_2 \;\Rightarrow\; -50 + 0 > 0 \vee -50 - 10a_1 > 0) & \text{(Condition (B2))}
\end{array}$$

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. ■

4.1 A Variant Procedure

In the previous section, we approximated the semantic condition (A2) by the constraint $p = 0 \Rightarrow \bigvee_{i \in I} \mathcal{L}_{f_i} p > 0$. As Corollary 1 shows, this is a sound approximation. However, the requirement that a vector field points *strictly* inwards, which is captured by $\mathcal{L}_{f_i} p > 0$, is too strong and leads to incompleteness, which leads to failure in finding suitable controlled invariant sets in practice. In this section, we weaken Formula (2) so that it can be used to handle more examples.

We weaken Condition (B2) and use the following weaker version of Formula (2) to test if $p(X, U) \geq 0$ is an inductive controlled invariant:

$$\exists U \forall X : \ (X \in \texttt{Init} \Rightarrow p(U, X) \geq 0) \ \wedge \ (p(U, X) \geq 0 \Rightarrow X \in \texttt{Safe}) \ \wedge \\ p(U, X) = 0 \Rightarrow \bigvee_{i \in I} (\mathcal{L}_{f_i} p > 0 \vee (\mathcal{L}_{f_i} p = 0 \wedge \bigwedge_{j \neq i} \mathcal{L}_{f_j} p < 0)) \qquad (3)$$

Formula (3) says that at the boundary ($p = 0$) of the controlled invariant ($p \geq 0$), either some vector field, say f_i, points strictly inwards ($\mathcal{L}_{f_i} p > 0$), or exactly one vector field is tangential ($\mathcal{L}_{f_i} p = 0$) and all others point strictly outside ($\mathcal{L}_{f_j} p < 0$). The counterintuitive condition – vector fields pointing strictly outwards – helps in proving that the tangential vector field will keep the system inside the controlled invariant.

We can now replace the constraint in Step (1) of the procedure in Figure 2 by Formula (3) 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 (3) is valid in the theory of reals, then there is a controlled invariant* CInv *that proves safety provided* $\|I\| > 1$.

Remark 2. The procedure could be unsound when $\|I\| = 1$. This unsoundness is related to the comment in Remark 1.

We illustrate the advantage of weakening the constraint for the inductive test by using the following example.

Example 3. 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, $\dot{y} = 0, \ \dot{x} = -1$. The initial state

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

Fig. 2. A sound procedure for solving the switching logic synthesis problem

is $x = 10, y = 10$ and the desired safety property is $y \geq 0$. We start with the template $a_1 x + a_2 y \geq a_3$. Formula (3) then becomes:

$$
\begin{array}{l}
\exists a_1, a_2, a_3 : \forall x, g : \\
\quad (x = 10 \wedge y = 10 \Rightarrow a_1 x + a_2 y \geq a_3) \wedge \qquad \text{(Condition } \texttt{(A1)}\text{)} \\
\quad (a_1 x + a_2 y \geq a_3 \ \Rightarrow y \geq 0) \wedge \qquad \text{(Condition } \texttt{(A3)}\text{)} \\
\quad (a_1 x + a_2 y = a_3 \ \Rightarrow -a_1 > 0 \vee (-a_1 = 0 \wedge -a_2 < 0) \vee \\
\qquad\qquad\qquad -a_2 > 0 \vee (-a_2 = 0 \wedge -a_1 < 0)) \quad \text{(Condition } \texttt{(A2)}\text{)}
\end{array}
$$

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. The previous method fails to find a controlled invariant for this example. ■

Example 4. Consider the train gate controller model from Example 2. Observe that the controller synthesized is very conservative and forces the system to switch from mode 1 to 2 in $t \leq 1$ units. Applying the variant procedure on this example, we get the following $\exists\forall$ formula:

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

This time the solver returned $a_1 = -5,\ a2 = 50$ as the solution, which gives $x - 5g \geq 50$ as the controlled invariant. So the resulting hybrid system has $x - 5g \geq 50$ as the state invariant for each mode and the guards $g_{12} = x - 5g \geq 50$ and $g_{21} = x - 5g > 50$ are computed. In this case, the switch from mode 1 to mode 2 could be delayed by as much as 10 units. ■

5 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 4.

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.

5.1 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 4 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 $p(\boldsymbol{u}, X) \geq 0$, then in the next iteration we use the template $p'(v, X) := p(\boldsymbol{u}, X) \geq v$ (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 $\{p(\boldsymbol{u}, X) \geq v \mid v \in [lb, 0],\ v \text{ an integer}\}$ in $O(\log \|lb\|)$ iterations.

5.2 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 (A2).

Example 5. Consider the train gate controller from example 4. The controlled invariant obtained by using the variant procedure on this example is $x - 10g \geq 50$. Observe that this is not the largest controlled invariant possible because when $x = 0$, this invariant implies $g \leq -5$, whereas safety just requires $g \leq 0$. If we add an additional constraint for tightening condition $A2$, 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 $x - 10g \geq 0$ as the controlled invariant. This is the largest controlled invariant for the template $x - 10g \geq v$. ■

Tightening Condition (A3). Before we describe the constraint for encoding tightness of Condition (A3), we need a few details on the procedure we use to solve the $\exists\forall$ formulas from [7]. The $\exists\forall$ formulas are solved 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 1. *It is the case that Formula (4) is implied by Formula (5).*

$$\exists U : \forall X : \ ((\bigwedge_j p_j = 0) \wedge (\bigwedge_k q_k > 0) \Rightarrow p \geq 0) \tag{4}$$

$$\exists U, \nu_j, \lambda_k, \lambda, \mu : \ \lambda_k \geq 0 \ \wedge \ \lambda \geq 0 \ \wedge \ \mu > 0 \ \wedge \ (\forall X : (\sum_p \nu_j p_j + \sum_k \lambda_k q_k + \lambda - \mu p = 0)) \tag{5}$$

Lemma 1 can be used to eliminate the internal $\forall X$ quantifier by noting that the polynomial in Formula (5) 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 (5) is a "slack" term. If Formula (5) is satisfied when $\lambda = 0$, then we say that the implication of Formula (4) is *tight*.

Now consider Condition (A3) which encodes the boundary condition. In Section 4, this condition was approximately captured in Formula (2) and Formula (3). Using elementary logical manipulations, we can rewrite these formulas in the form

$$\exists U : \bigwedge_i (\forall X : (\bigwedge_j p_{ij} = 0) \wedge (\bigwedge_k q_{ik} > 0) \Rightarrow p_i \geq 0). \tag{6}$$

Apply Lemma 1 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 (A3). 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 1. The constraint we add is the following:

$$\phi_{opt} := \bigvee_i (\lambda_i = 0) \tag{7}$$

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

Theorem 3 (Correctness). *Let $\boldsymbol{u}$ 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 (2) (or Formula (3)) using Lemma 1. Then, there is no controlled invariant $p(\boldsymbol{u}, X) \geq \alpha$ for any $\alpha < 0$ that also satisfies the existential formula generated from Formula (2) (or Formula (3)) using $p(\boldsymbol{u}, X) \geq \alpha$ as a template.*

6 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 4 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 (2) generalizes to

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

When the template is a disjunction, say $p_1 \geq 0 \vee p_2 \geq 0$, then Formula (2) generalizes to

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

We can similarly generalize Formula (3) for the case when the template is a disjunction or conjunction of polynomial inequalities. The following example illustrates this case.

Example 6. 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 (3) 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 (A2) (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. ■

The methods discussed in Section 5 can also be extended to the case of conjunctive and disjunctive templates, but we do not discuss the details here.

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 4) already reduces the possibility of synthesizing zeno hybrid systems. This aspect needs further investigation.

We have a preliminary implementation of the approaches described in Section 4, along with the optimality variants of Section 5. The input is a multi-modal system, a safety property, and a template – all specified using only polynomials – and the output is a switching logic, if it exists. This implementation was used to solve the examples in the paper and their variants. We currently use the technique from [7] for solving the $\exists\forall$ constraints. Future work involves improving this technique using a symbolic nonlinear solver. This will enable applicability to larger and more complex examples. Our constraint-based technique relies heavily on the choice of the template. 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.

7 Related Work

Constraint-based techniques have been used for safety verification of hybrid systems [7, 11, 12], 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 [8, 9]. 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, 13], and abstraction-based approaches that do the same, but on an abstraction or approximation of the system [6, 10]. 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. 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 [5].

8 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] Alur, R., Courcoubetis, C., Halbwachs, N., Henzinger, T.A., Ho, P.-H., Nicollin, X., Olivero, A., Sifakis, J., Yovine, S.: The algorithmic analysis of hybrid systems. Theoretical Computer Science 138(3), 3–34 (1995)
[2] Asarin, E., Bournez, O., Dang, T., Maler, O., Pnueli, A.: Effective synthesis of switching controllers for linear systems. Proc. IEEE 88(7), 1011–1025 (2000)
[3] Blanchini, F.: Set invariance in control. Automatica 35, 1747–1767 (1999)
[4] Burns, K., Gidea, M.: Differential Geometry and Topology: With a view to dynamical systems. Chapman & Hall, Boca Raton (2005)
[5] Colón, M.: Schema-guided synthesis of imperative programs by constraint solving. In: LOPSTR, pp. 166–181 (2004)
[6] Cury, J., Brogh, B., Niinomi, T.: Supervisory controllers for hybrid systems based on approximating automata. IEEE Trans. Aut. Control 43, 564–568 (1998)
[7] Gulwani, S., Tiwari, A.: Constraint-based approach for analysis of hybrid systems. In: Gupta, A., Malik, S. (eds.) CAV 2008. LNCS, vol. 5123, pp. 190–203. Springer, Heidelberg (2008)
[8] Koo, T., Sastry, S.: Mode switching synthesis for reachability specification. In: Di Benedetto, M.D., Sangiovanni-Vincentelli, A.L. (eds.) HSCC 2001. LNCS, vol. 2034, pp. 333–346. Springer, Heidelberg (2001)
[9] Manon, P., Valentin-Roubinet, C.: Controller synthesis for hybrid systems with linear vector fields. In: Proc. IEEE Symp. on Intell. Control, pp. 17–22 (1999)
[10] Moor, T., Raisch, J.: Discrete control of switched linear systems. In: Proc. Eur. Control Conf. ECC 1999 (1999)
[11] Prajna, S., Jadbabaie, A.: Safety verification of hybrid systems using barrier certificates. In: Alur, R., Pappas, G.J. (eds.) HSCC 2004. LNCS, vol. 2993, pp. 477–492. Springer, Heidelberg (2004)
[12] Sankaranarayanan, S., Sipma, H., Manna, Z.: Constructing invariants for hybrid systems. In: Alur, R., Pappas, G.J. (eds.) HSCC 2004. LNCS, vol. 2993, pp. 539–554. Springer, Heidelberg (2004)
[13] Tomlin, C., Lygeros, L., Sastry, S.: A game-theoretic approach to controller design for hybrid systems. Proc. of the IEEE 88(7), 949–970 (2000)