Synthesizing Switching Logic using Constraint Solving

Ankur Taly

Dept. of Computer Science, Stanford University

(Joint work with Ashish Tiwari and Sumit Gulwani)

1. What are Hybrid systems ?
   - Formal framework
   - Example : Train gate controller
   - Desired Properties
2. Synthesis
   - Semantic procedure
   - Practical implementations
   - Illustration : Train gate controller
3. Conclusions and Future work

What are Hybrid systems ?

- Dynamical systems with both discrete and continuous behavior.
- Multiple modes each with its own differential equation which governs the dynamics in that mode.
- A switching logic which governs the discrete mode changes.
- Example : Thermostat - *on* and *off* mode.
- Interested in safety and stability properties of such systems. Does the thermostat maintain the temperature between 70 F and 80 F ?

Notation and Definitions

$\mathtt{HS}(\mathtt{MDS}, \mathtt{Init}, \mathtt{SwL})$

- Set of variables $X = \{x_1, \ldots, x_n\}$, each x_i taking values in $\mathbb{R}$. The vector of values $\vec{x} \in \mathbb{R}^n$ at any instant represents the continuous state of the system.
- Multi-modal Dynamical System ($\mathtt{MDS}$) : A set of modes $I = \{1, \ldots, k\}$ representing the discrete state.
  - Dynamics in mode i, $\frac{d\vec{x}}{dt} = f_i(\vec{x})$ (where f_i is a lipschitz field)
  - $F_i(\vec{x}_0, t)$ denotes the solution of the above differential equation with initial state $\vec{x}_0$.
- Set of initial states $\mathtt{Init} \subseteq \mathbb{R}^n$
- Switching Logic ($\mathtt{SwL}$) : $\mathtt{SwL} := \langle (g_{ij})_{i \neq j; i,j \in I}, (\mathtt{StateInv}_i)_{i \in I} \rangle$ where
  - $\mathtt{StateInv}_i$: state invariant for mode i (closed set).
  - g_{ij} : guard for transition from mode i to j. Identity resets

Example : Train gate controller

Consider a train approaching a railroad crossing.

- Let x be the distance of the train from the gate and g be the gate angle.
- Three modes : Normal, About to lower and Lowering.

Normal

$$\frac{dx}{dt} = -50, \frac{dg}{dt} = 0$$

$\texttt{StateInv} := x > 1000$

About to lower

$$\frac{dx}{dt} = -50, \frac{dg}{dt} = 0$$

$\texttt{StateInv} := 1000 \leq x \leq 500$

Lowering

$$\frac{dx}{dt} = -50, \frac{dg}{dt} = -10$$

$\texttt{StateInv} := x < 500$

- $\texttt{Init} : x = 1000 \bigwedge g = 90$, $g_{12} : x = 1000$ and $g_{23} : x = 500$.

Desired Properties

Safety

A hybrid system is safe with respect to a safety property $\texttt{SafeProp} \subseteq \mathbb{R}^n$ if all reachable continuous states $\vec{x} \in \texttt{SafeProp}$.

Non Blocking

For every mode i, for all $\vec{x} \in \partial\texttt{StateInv}_i$, there should be a mode j (may be same as mode i) such that $\exists\epsilon > 0 : (F_j(\vec{x}, [0, \epsilon]) \in \texttt{StateInv}_j \bigwedge \vec{x} \in g_{ij})$.

Min. Dwell time

There exists a fixed time duration t_a such that on entering a mode, the continuous flow can evolve within that mode for at least time t_a.

Two Problems

Verification Problem

Given a hybrid system HS(MDS, Init, SwL) and a safety property SafeProp, the problem is to verify that HS is safe with respect to SafeProp.

Synthesis Problem - This talk !

Given a MDS, Init and a safety property SafeProp, the problem is to synthesize the switching logic SwL so that the resulting hybrid system HS(MDS, Init, SwL) is safe and non-blocking with respect to SafeProp.

Two Problems

Verification Problem

Given a hybrid system $\mathtt{HS}(\mathtt{MDS}, \mathtt{Init}, \mathtt{SwL})$ and a safety property $\mathtt{SafeProp}$, the problem is to verify that $\mathtt{HS}$ is safe with respect to $\mathtt{SafeProp}$.

Synthesis Problem - This talk !

Given a $\mathtt{MDS}, \mathtt{Init}$ and a safety property $\mathtt{SafeProp}$, the problem is to synthesize the switching logic $\mathtt{SwL}$ so that the resulting hybrid system $\mathtt{HS}(\mathtt{MDS}, \mathtt{Init}, \mathtt{SwL})$ is safe and non-blocking with respect to $\mathtt{SafeProp}$.

Synthesizing switching logic

Related Work : Fixed point based approaches :

- Involves computing a safe subset of the"reachable states" closed under reduction.
- Cannot handle non trivial continuous dynamics as there is no effective notion of "next" state unless suitable abstractions are applied.

Our Approach : Deductive Verification + Constraint Solving.

- Catch : Direct constraint solving with templates for the unknowns in the switching logic and for the safety invariant for each mode, may lead to degenerate systems (zeno or deadlocked).
- Idea : Synthesize Inductive Controlled Invariants instead of safety invariants.

Synthesizing switching logic

Related Work : Fixed point based approaches :

- Involves computing a safe subset of the"reachable states" closed under reduction.
- Cannot handle non trivial continuous dynamics as there is no effective notion of "next" state unless suitable abstractions are applied.

Our Approach : Deductive Verification + Constraint Solving.

- Catch : Direct constraint solving with templates for the unknowns in the switching logic and for the safety invariant for each mode, may lead to degenerate systems (zeno or deadlocked).
- Idea : Synthesize Inductive Controlled Invariants instead of safety invariants.

Trajectories

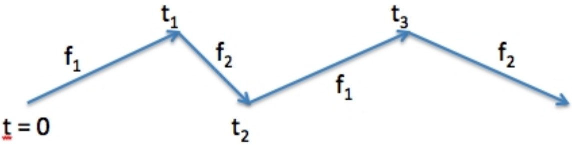

Figure: Trajectory of $\vec{x}(t)$

Given an initial state $\vec{x}_0$, $\mathbf{x}(t)$ is a *trajectory* of an MDS if

- $\mathbf{x}(0) = \vec{x}_0$ and $\mathbf{x}(t)$ is continuous.
- There exists an increasing sequence $0 \leq t_1 < t_2 < \ldots$ such that for each t_i, there is a mode j such that $\frac{dx}{dt} = f_j(x(t))$ for all $t_i < t < t_{i+1}$.

Inductive Controlled Invariant

Inductive Controlled Invariant

A closed set CInv is said to be an inductive controlled invariant iff for each point $\vec{x} \in \partial\texttt{CInv}$, there exists a vector field f_i such that $\exists\epsilon > 0 : F_i(\vec{x}, (0, \epsilon)) \in \texttt{CInv}$.

Illustration :

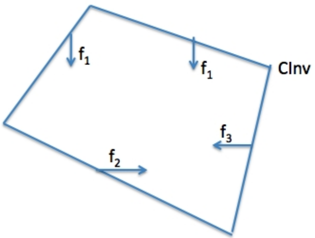

Figure: Trajectory of $\vec{x}(t)$

State variables : x, y Dynamics :

- $f_1 : \dot{x} = 0, \dot{y} = -1$
- $f_2 : \dot{x} = 1, \dot{y} = 0$
- $f_3 : \dot{x} = -1, \dot{y} = 0$ '

The synthesis procedure (at a semantic level)

SynthSwitchLogic(MDS, SafeProp) :

1. Find a closed set CInv such that the following conditions hold

   (A1) Init $\subseteq$ CInv

   (A2) CInv $\subseteq$ SafeProp

   (A3) for all $\vec{x} \in \partial$CInv, there exists an $i \in I$ such that $\exists \epsilon : F_i(\vec{x}, (0, \epsilon)) \subseteq$ CInv

2. Let $\mathtt{bdry}_i := \{\vec{x} \in \partial\mathtt{CInv} \mid \exists \epsilon > 0 : F_i(\vec{x}, (0, \epsilon)) \subseteq \mathtt{CInv}\}$ for all $i \in I$,

3. Let $\mathtt{StateInv}_i := \mathtt{CInv}$ for all $i \in I$,

4. Let $g_{ij} := \mathtt{bdry}_j \cup \mathtt{Interior}(\mathtt{CInv})$ for all $i \neq j; i, j \in I$,

Return $\mathtt{SwL} := \langle (g_{ij})_{i \neq j; i,j \in I}, (\mathtt{StateInv}_i)_{i \in I} \rangle$

Properties

Theorem 1

For every switching logic $\mathtt{SwL}$ returned by $\mathtt{SynthSwitchLogic}$, the hybrid system $\mathtt{HS}(\mathtt{MDS}, \mathtt{SwL})$ is non-blocking.

Soundness and Completeness under a technical side condition.

Theorem 2

If $\mathtt{SynthSwitchLogic}$ returns the switching logic $\mathtt{SwL}$, then the hybrid system $\mathtt{HS}(\mathtt{MDS}, \mathtt{SwL})$ is safe. If $\mathtt{HS} = \mathtt{HS}(\mathtt{MDS}, \mathtt{SwL})$ is a safe hybrid system that satisfies the min-dwell-time property and if $\mathtt{SafeProp}$ is a closed set, then procedure $\mathtt{SynthSwitchLogic}$ will return a switching logic.

Issues

Second order quantifier

The procedure SynthSwitchLogic(MDS,SafeProp) naturally gives a $\exists$CInv : $\forall \vec{x}$: formula. Need to get rid of the second order quantifier.

Solution :

- Restrict to Polynomial hybrid systems.
- Use a template for CInv. Simple case : CInv $:= P(u, \vec{x}) \geq 0$ ∂CInv $:= P(u, \vec{x}) = 0$. This gives the first order formula $\exists u \forall \vec{x}$.
- Write effective logical formulas for conditions A1 (easy), A2(easy) and A3 (tricky !)
- Check if the $\exists\forall$ formula is valid over the theory of reals (Decidable). Also Gulwani et al propose sound heuristics for efficiently deciding validity of such formulas.

Issues

Second order quantifier

The procedure SynthSwitchLogic(MDS,SafeProp) naturally gives a $\exists \texttt{CInv} : \forall \vec{x} :$ formula. Need to get rid of the second order quantifier.

Solution :

- Restrict to Polynomial hybrid systems.
- Use a template for CInv. Simple case : $\texttt{CInv} := P(u, \vec{x}) \geq 0$ $\partial\texttt{CInv} := P(u, \vec{x}) = 0$. This gives the first order formula $\exists u \forall \vec{x}$.
- Write effective logical formulas for conditions A1 (easy), A2(easy) and A3 (tricky !)
- Check if the $\exists\forall$ formula is valid over the theory of reals (Decidable). Also Gulwani et al propose sound heuristics for efficiently deciding validity of such formulas.

Issues

Encoding A3 is tricky

How do we decide $\exists \epsilon : F_i(\vec{x}, (0, \epsilon)) \subseteq \texttt{CInv}$ without computing the closed form solution F_i of the differential equation ?

Solution:

- Sound Approximation (A3') : $\exists \epsilon : F_i(\vec{x}, (0, \epsilon)) \subseteq \texttt{Interior(CInv)}$
- Make use of Lie Derivates to encode the above condition.
- $\mathcal{L}_{f_i} p := \frac{dp}{dt} = \sum_{x \in X} \frac{\partial p}{\partial x} \frac{dx}{dt}$.
- $(\bigvee_{i \in I} \mathcal{L}_{f_i} P(u, \vec{x}) > 0)) \Longrightarrow (A3')$

Issues

Encoding A3 is tricky

How do we decide $\exists\epsilon : F_i(\vec{x}, (0, \epsilon)) \subseteq \texttt{CInv}$ without computing the closed form solution F_i of the differential equation ?

Solution:

- Sound Approximation (A3') : $\exists\epsilon : F_i(\vec{x}, (0, \epsilon)) \subseteq \texttt{Interior}(\texttt{CInv})$
- Make use of Lie Derivates to encode the above condition.
- $\mathcal{L}_{f_i} p := \frac{dp}{dt} = \sum_{x \in X} \frac{\partial p}{\partial x} \frac{dx}{dt}$.
- $(\bigvee_{i \in I} \mathcal{L}_{f_i} P(u, \vec{x}) > 0)) \Longrightarrow (A3')$

A sound and practical procedure

$$\exists u \forall \vec{x} : \quad \begin{array}{l} (\vec{x} \in \texttt{Init} \Rightarrow P(u,\vec{x}) \geq 0) \ \wedge \\ (P(u,\vec{x}) \geq 0 \Rightarrow \vec{x} \in \texttt{SafeProp}) \ \wedge \\ (P(u,\vec{x}) = 0 \Rightarrow \bigvee_{i \in I} \mathcal{L}_{f_i} P(u,\vec{x}) > 0) \end{array}$$

- Above procedure is sound but incomplete for polynomial hybrid systems.
- Incomplete for cases where controlled invariant has a point on $\vec{x}$ on the boundary where $\mathcal{L}_{f_i} P(u,\vec{x}) \leq 0$ for all i.
- Relatively more complete (and sound) encoding of A3 :

$$\bigvee_{i \in I} (\mathcal{L}_{f_i} p(U,X) > 0 \vee (\mathcal{L}_{f_i} p = 0 \wedge \bigwedge_{j \neq i} \mathcal{L}_{f_j} p < 0).$$

A sound and practical procedure

$$\exists u \forall \vec{x} : \quad \begin{array}{l} (\vec{x} \in \texttt{Init} \Rightarrow P(u,\vec{x}) \geq 0) \ \wedge \\ (P(u,\vec{x}) \geq 0 \Rightarrow \vec{x} \in \texttt{SafeProp}) \ \wedge \\ (P(u,\vec{x}) = 0 \Rightarrow \bigvee_{i \in I} \mathcal{L}_{f_i} P(u,\vec{x}) > 0) \end{array}$$

- Above procedure is sound but incomplete for polynomial hybrid systems.
- Incomplete for cases where controlled invariant has a point on $\vec{x}$ on the boundary where $\mathcal{L}_{f_i} P(u,\vec{x}) \leq 0$ for all i.
- Relatively more complete (and sound) encoding of A3 :

$$\bigvee_{i \in I} (\mathcal{L}_{f_i} p(U,X) > 0 \vee (\mathcal{L}_{f_i} p = 0 \wedge \bigwedge_{j \neq i} \mathcal{L}_{f_j} p < 0).$$

Synthesizing the Train Gate controller

Synthesize the switching logic

Init : $g = 90 \wedge x = 1000$ and SafeProp : $x > 0 \vee g \leq 0$.

About to lower	Gate lowering
$\frac{dx}{dt} = -50 \wedge \frac{dg}{dt} = 0$	$\frac{dx}{dt} = -50 \wedge \frac{dg}{dt} = -10$

Assume a template of the form $x + a_1 g \geq a_2$ for CInv.

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

Synthesizing the Train Gate controller

Synthesize the switching logic

`Init` : $g = 90 \wedge x = 1000$ and `SafeProp` : $x > 0 \vee g \leq 0$.

About to lower	Gate lowering
$\frac{dx}{dt} = -50 \wedge \frac{dg}{dt} = 0$	$\frac{dx}{dt} = -50 \wedge \frac{dg}{dt} = -10$

Assume a template of the form $x + a_1 g \geq a_2$ for `CInv`.

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

Synthesizing the Train Gate Controller

Solver returns $a_1 = -10, a_2 = 50$.

- Therefore, controlled invariant is $x - 10g \geq 50$.
- At all points on the boundary of the state invariant : $x - 10g = 50$, dynamics of mode 2(gate lowering) points inwards and that of mode 1(About to lower) points outwards.
- Therefore $g_{12} := x - 10g \geq 50$, $g_{21} = \phi$ and $\mathtt{StateInv}_1 = \mathtt{StateInv}_2 := x - 10g \geq 50$ is an admissible switching logic.

Synthesizing a good controller

- Larger CInv = more liberal controller
- Tighten condition $A2$.

$$\partial\texttt{CInv} \cap \partial\texttt{SafeProp} \neq \emptyset.$$

  Gives the largest possible controlled invariant $(x - 10g \geq 0)$ for the train gate example !
- Binary Search to optimize the constant term α in invariants of the form $P(u, \vec{x}) \geq \alpha$.
- More heuristics in the paper

Conclusions and Future work

Conclusions :

- We propose a sound and complete (in theory) procedure based on inductive controlled invariants for synthesizing switching logic for Hybrid systems.
- We propose several sound practical implementation of this procedure for polynomial hybrid systems.
- We propose heuristics for generating optimal controlled invariants.

Future Work :

- Extend the synthesis procedure to more complicated systems with implicit state invariants.
- Strengthen the constraints so that the synthesized systems have non-zeno behavior.
- Synthesize systems that have certain liveness and stability properties : Synthesize Lyapunov functions ?

Conclusions and Future work

Conclusions :

- We propose a sound and complete (in theory) procedure based on inductive controlled invariants for synthesizing switching logic for Hybrid systems.
- We propose several sound practical implementation of this procedure for polynomial hybrid systems.
- We propose heuristics for generating optimal controlled invariants.

Future Work :

- Extend the synthesis procedure to more complicated systems with implicit state invariants.
- Strengthen the constraints so that the synthesized systems have non-zeno behavior.
- Synthesize systems that have certain liveness and stability properties : Synthesize Lyapunov functions ?

Thank You !