ECE 584: Embedded System Verification
Constructing Invariants for Hybrid Automata.

Slides from Sriram Sankaranarayanan
University of Colorado, Boulder, CO.

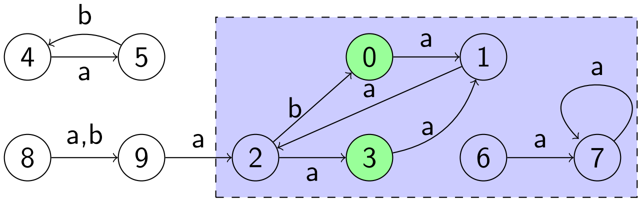
November 8, 2012

Outline

1. Invariant Synthesis
2. Hybrid Systems
3. Invariants For Hybrid Systems
4. Constraint-Based Invariant Generation

Invariants

Finite State Automaton:



Initial States: $\{0, 3\}$.
Invariant Set: Subset of states that contains

1. All initial states, and
2. All states reachable in one or more steps from initial.

Example:

$\{0, 1, 2, 3, 6, 7\}$ $\{0, 1, 2, 3, 4, 5\}$
$\{0, 1, \ldots, 9\}$ $\{0, 1, 2, 3\}$ $\leftarrow$ **Strongest Invariant**

Proving Programs Correct

Compute $\lceil \sqrt{n} \rceil$, for $n \geq 0$.

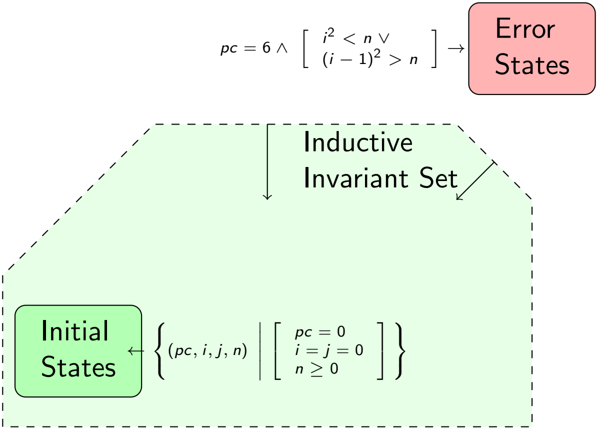
```
int computeSqrt ( int n )
    @Pre: n ≥ 0
    int i, j = (0, 0);
    while ( j ≤ n ) {
        i := i + 1;
        j := j + 2 * i − 1;
    }
    @Post: i^2 ≥ n ∧ (i − 1)^2 ≤ n
```

Infinite State Automaton

$$(pc, i, j, n) \longrightarrow (pc', i', j', n')$$

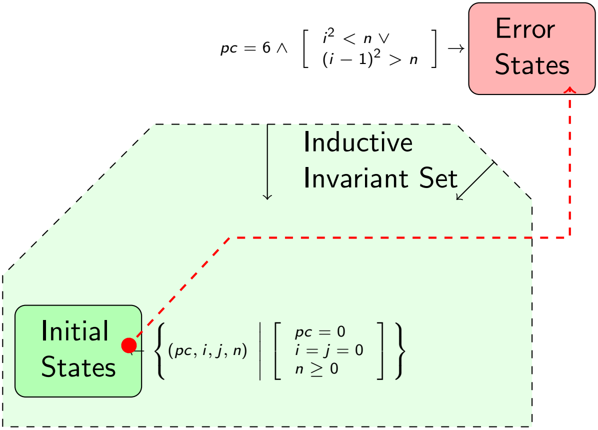
Inductive Invariant Set

Start inside invariant set $\Rightarrow$ remain in invariant set.



Inductive Invariant Set

Start inside invariant set ⇒ remain in invariant set.



Proving Programs Correct

Example: Compute $\lceil \sqrt{n} \rceil$, for $n \geq 0$.

```
int computeSqrt ( int n )
    @Pre: n ≥ 0
1:  int i, j = (0, 0);
2:  while ( j ≤ n ) {
3:      i := i + 1;
4:      j := j + 2 ∗ i − 1;
5:  }
    @Post: i^2 ≥ n ∧ (i − 1)^2 ≤ n
```

Ind. Invariant at location 2:

$$j = i^2,\ n \geq 0,\ j \leq n + 1,\ j \leq n - 2i - 1\,.$$

Q: How do we find the invariant above?

Automatic Invariant Synthesis Techniques

(Reasonably) well-understood problem for programs:

- Abstract Interpretation with Widening Framework:
  - Intervals: [Cousot+Cousot'78]
  - Polyhedral Invariants: [Cousot+Halbwachs'78]
  - Octagons [Miné'02]
  - Templates [S.+Sipma+Manna'05]
  - Symbolic Ranges [S.+Ivancic+Gupta'07]
  - Algebraic/Semi-algebraic [Carbonell+Others'05]
- Constraint-based Invariant Generation:
  - Linear Invariants: [Colón+S.+Sipma'03, S.+Sipma+Manna'04, Gulwani+Others'07, Gupta+Rybalchenko+Majumdhar'08,...]
  - Algebraic (Polynomial) Invariants: [S.+Sipma+Manna'04, Carbonell+Kapur'04, Seidl+Muller-Olm'04, Colón'04, ...]

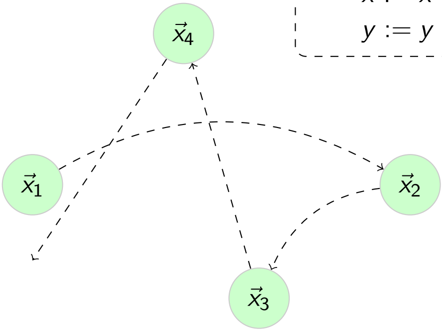
Challenge: Invariants for *Hybrid Systems*.

Dynamical Systems

Discrete dynamical systems: defined by maps.

$$\vec{x}(n+1) = F(n, \vec{x}(n))\,.$$

$$\begin{aligned}&\text{while}(x^2 + y^2 \leq 3)\\ &\qquad x := x + y^{\frac{1}{2}}\\ &\qquad y := y - x^{\frac{1}{2}}\end{aligned}$$



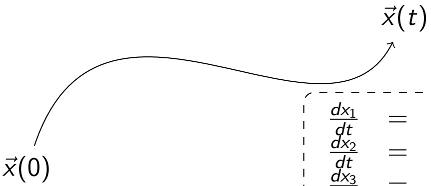
Examples:

Automata, Programs, Digital Circuits

Continuous Systems

Continuous dynamical systems: defined by flows.

$$\frac{d\vec{x}}{dt} = F(t, \vec{x})\,.$$



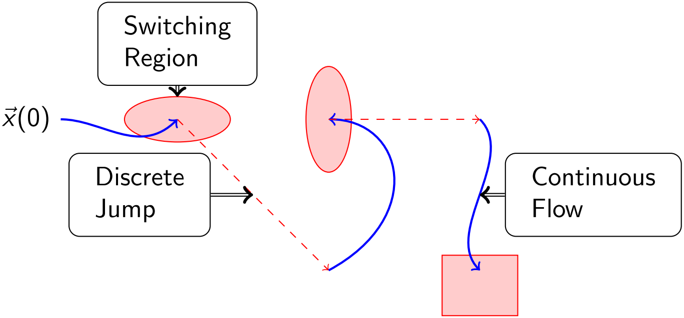
$$\begin{aligned}
\frac{dx_1}{dt} &= x_1 \sin\psi - x_2 \cos\psi \\
\frac{dx_2}{dt} &= x_1^2 - x_1 x_2 \sin(2\psi) \\
\frac{dx_3}{dt} &= cos\psi \\
\frac{d\psi}{dt} &= 0
\end{aligned}$$

Examples:

Mechanical systems, analog circuits, biological systems (cell signalling mechanisms).

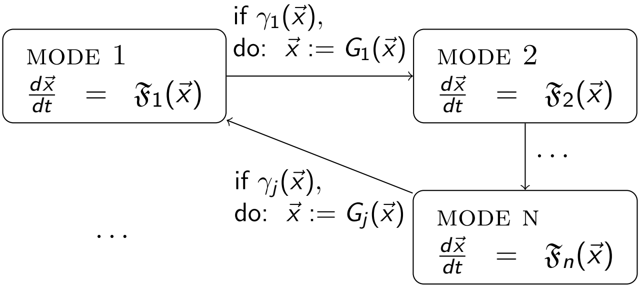
Hybrid Systems

1. Continuous-time Flows + Discrete-time Transitions.
2. *Multi-Modal*:Continuous-time dynamics depend on the state.



Hybrid Automaton

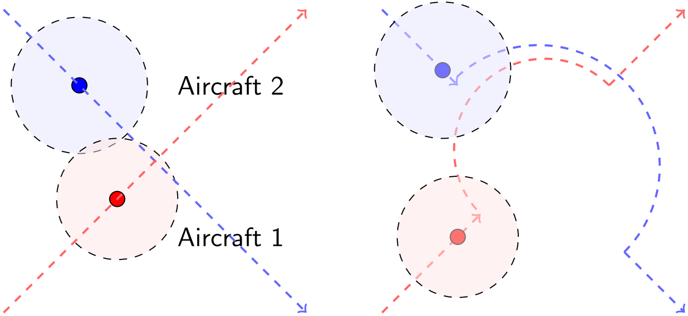
[Alur et al.'96; ...]



- Finite set of *modes*. $Q : \{q_1, \ldots, q_m\}$
- Continuous state variables. $\vec{x} : (x_1, \ldots, x_n)$.
- Dynamics for each mode.
- Discrete Transitions between modes.

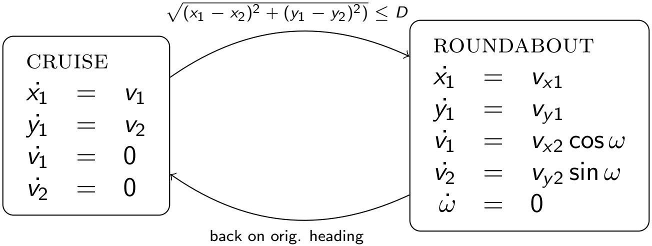
Example # 2: Conflict Resolution Maneuvers

Conflict resolution protocol. [Tomlin et al.'98]



Collision Avoidance Model

Hybrid automaton for each aircraft:



Hybrid Automata: Verification Challenges

Hybrid automata are Turing complete.

Even the simplest ones:

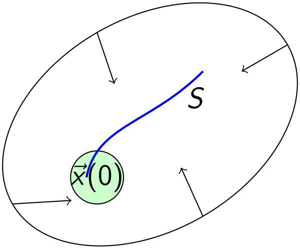
- Constant ODEs. $\frac{d\vec{x}}{dt} = \vec{c}$.
- Small number of variables ~ 3.

Verification is therefore **undecidable** even for simpler cases.

Problem: *Interesting* hybrid automata models do not fall into a known decidable class.

Invariants for Hybrid Systems.

Invariants: Differential Equations

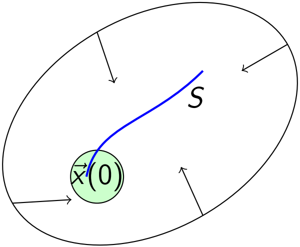


Set S is *positive invariant* for flow φ iff

$$\forall\ \vec{x}(0) \in S,\ \ \varphi(\vec{x}(0), t) \in S\,.$$

Start inside set $S \Rightarrow$ flow remains in S.

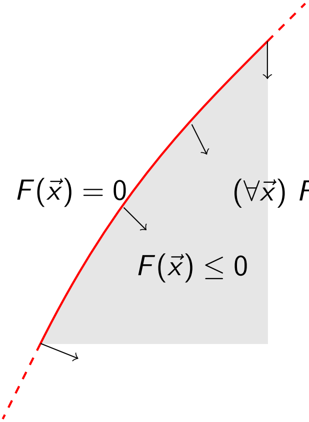
Positive Invariant Checking



Need to find *closed form* solutions.

- Not available in general for non-linear systems.
- Even for linear systems, closed form involves transcendental functions: $\sin, \cos, exp$.

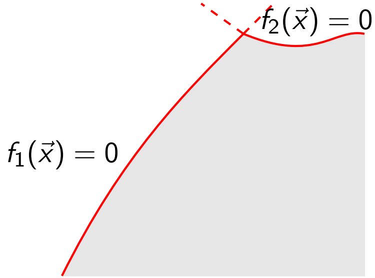
Positive Invariance Principle



$$(\forall \vec{x})\ F(\vec{x}) = 0 \ \Rightarrow\ \frac{dF}{dt} = \frac{\partial F}{\partial x}\frac{dx}{dt} < 0$$

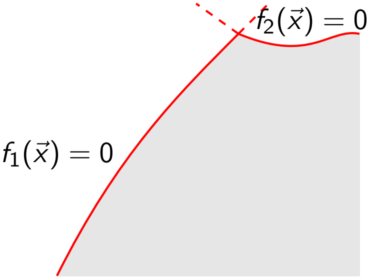
Intuition: Flow on the boundary points "inwards".
Formal Statement: Nagumo Viability Theorem/Subtangential condition.

Checking Positive Invariance



$$
\begin{array}{lll}
f_1(\vec{x}) = 0 \ \wedge \ f_2(\vec{x}) \leq 0 & \models & \frac{df_1}{dt} < 0 \\
f_1(\vec{x}) \leq 0 \ \wedge \ f_2(\vec{x}) = 0 & \models & \frac{df_2}{dt} < 0
\end{array}
$$

Checking Positive Invariance

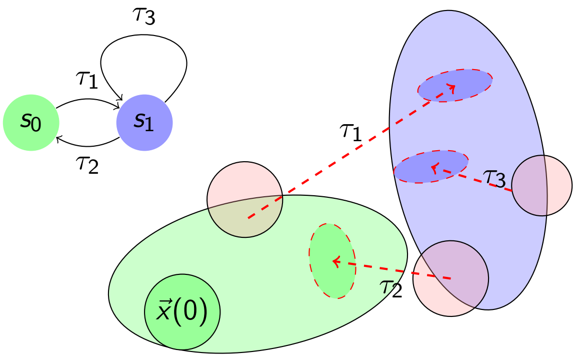


$$f_1(\vec{x}) = 0 \;\wedge\; f_2(\vec{x}) \leq 0 \quad \models \quad \frac{df_1}{dt} < 0$$
$$f_1(\vec{x}) \leq 0 \;\wedge\; f_2(\vec{x}) = 0 \quad \models \quad \frac{df_2}{dt} < 0$$

Soundness Warning: Unsound for "pathological" functions f_i.

- Restrict applications to well-behaved f_i (convex functions). [Blanchini+Miani]
- Or use a weaker version that works for corner cases. [Platzer+Clarke'07]

Invariants for Hybrid Systems

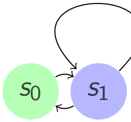


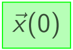
- Positive invariant for each discrete modes.
- "Preserved" by discrete transitions.
- Invariant checking reduced to checking entailments.
  Efficient decision procedures for linear systems.
  Theorem proving in the general case.

Invariant Synthesis for Hybrid Systems

Invariant Synthesis.

[Henzinger et al.'96,...]

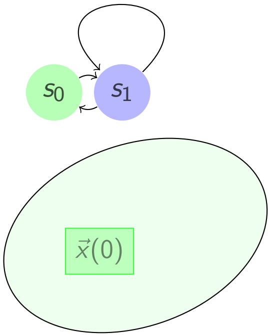




- Start with Initial States.

Invariant Synthesis.

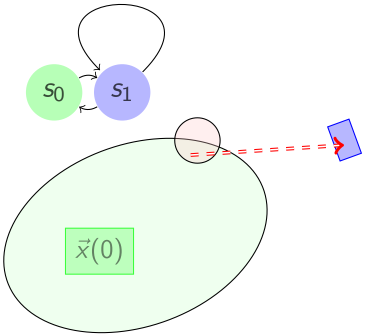
[Henzinger et al.'96,...]



- Start with Initial States.
- Positive Invariant computation for differential equations.

Invariant Synthesis.

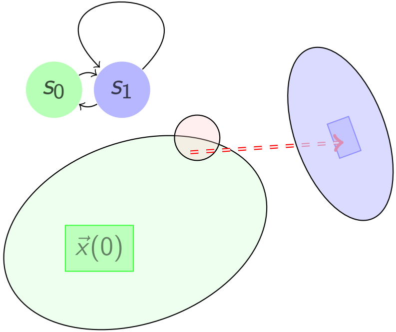
[Henzinger et al.'96,...]



- Start with Initial States.
- Positive Invariant computation for differential equations.
- Computing images across discrete transitions.

Invariant Synthesis.

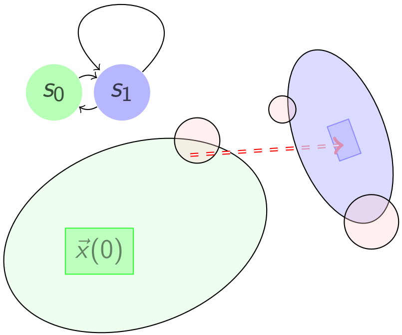
[Henzinger et al.'96,...]



- Start with Initial States.
- Positive Invariant computation for differential equations.
- Computing images across discrete transitions.
- Iterate until convergence (Widening/Extrapolation)

Invariant Synthesis.

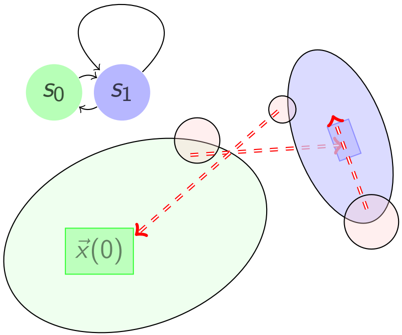
[Henzinger et al.'96,...]



- Start with Initial States.
- Positive Invariant computation for differential equations.
- Computing images across discrete transitions.
- Iterate until convergence (Widening/Extrapolation)

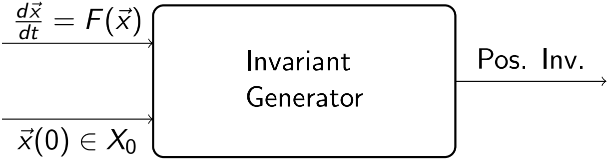
Invariant Synthesis.

[Henzinger et al.'96,...]



- Start with Initial States.
- Positive Invariant computation for differential equations.
- Computing images across discrete transitions.
- Iterate until convergence (Widening/Extrapolation)

Invariants for Continuous Systems



Search for smallest set I such that:

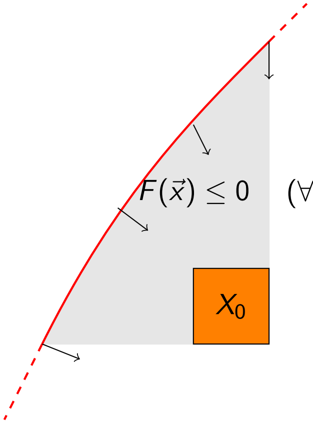
1. $X_0 \subseteq I$.
2. I is a positive invariant. For each time trajectory $\vec{x} : [0, T] \mapsto \mathcal{R}^n$, we have

$$(\vec{x}(0) \in I \;\Rightarrow\; \forall\ t \in [0, T],\ \vec{x}(t) \in I)\ .$$

Invariant Synthesis for Continuous Systems

- **Constraint-Based** techniques: [S.+others'04, Prajna+Jadbabaie'04, Gulwani+Tiwari'09, Platzer+Clarke'09,...]
  1. Fix a family of invariants by choosing an *ansatz*.
  2. Deduce constraints for invariants.
  3. Solve to obtain invariants.
- **Iterative** techniques: [S.+Others'06 - '11]
  1. Express positive invariants as fixed point of monotone operator.
  2. Use Kleene iteration to compute fixed points.
- **Flowpipe** construction techniques: [Zhao'93, Asarin+Others'00, Krogh+Others'98, Kurzhanski +Varaiya'07, Girard'05, ...]
  1. Time bounded invariants.
  2. Rigorous set-valued integration for a class of sets and ODEs.
- **Other techniques**: Optimal control theory, interval Taylor methods and many others. [Berz+Makino, Mitchell+Tomlin,...]

Constraint-Based Method



$$(\forall \vec{x})\ F(\vec{x}) = 0 \ \Rightarrow \ \frac{dF}{dt} < 0$$

Goal: Search for positive invariants F.

Constraint-Based Invariant Synthesis

1. Fix a family of candidate invariants: $F(\vec{c}, \vec{x}) \leq d$.

$$c_1 x_1 + \cdots + c_n x_n \leq d\,.$$

2. Derive constraints on $\vec{c}$ to encode positive invariance: Flow on the boundary points "inwards".

$$(\forall\ \vec{x})\ \left[\sum_i c_i x_i = d\ \Rightarrow\ \frac{d(\sum_i c_i x_i)}{dt} < 0 \right]$$

3. Solve constraints to find invariants.

Encoding Positive Invariance

Given ODE,

$$\frac{d\vec{x}}{dt} = F(\vec{x}),\ \vec{x}(0) \in X_0 \text{ and}$$

ansatz $g(\vec{c}, \vec{x}) \leq 0$, **search for** a positive invariant function

- Initial conditions

$$\forall\ \vec{x}\ (\vec{x} \in X_0)\ \Rightarrow\ g(\vec{c}, \vec{x}) \leq 0\,.$$

- Positive Invariance

$$\forall\ \vec{x}\ g(\vec{c}, \vec{x}) = 0\ \Rightarrow\ (\nabla g) \cdot F < 0\,.$$

Recall Lagrangian Relaxation for LP

- Primal: $\max c^T x$ subject to $Ax \leq b, x \geq 0$;
- Dual: $\min b^T y$ subject to $A^T y \geq c, y \geq 0$;
- **Weak Duality Theorem:** For any feasible y of the Dual and feasible x of the primal, $b^T y \geq c^T x$.
- Langrange relaxation: $\max c^T x + \lambda^T (b - Ax)$, λ's are called the Lagrange multipliers
- $\forall \hat{\lambda}, c^T x^* \leq c^T x^* + \hat{\lambda}^T (b - Ax^*) \leq c^T \bar{x} + \hat{\lambda}^T (b - A\bar{x})$
- For any fixed set of $\hat{\lambda}$ values, the optimal for the Lagrangian Relaxation will be no smaller than the optimal result to Primal.

Lagrangian Relaxation

Idea: Use Lagrange multipliers to dualize the implication:

$$(\forall\ \vec{x})\ g_1(\vec{x}) \leq 0\ \wedge\ \ldots\ \wedge\ g_m(\vec{x}) \leq 0\ \Rightarrow\ g(\vec{x}) \leq 0\,.$$

$$(\exists\ \lambda_1, \ldots, \lambda_m \geq 0)\ \lambda_1 g_1 + \cdots + \lambda_m g_m \equiv g$$

It is always sound to relax this way.

Lagrangian Relaxation

Idea: Use Lagrange multipliers to dualize the implication:

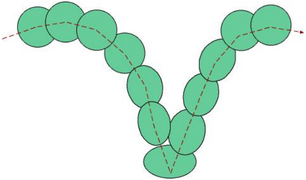
$$(\forall\ \vec{x})\ g_1(\vec{x}) \leq 0\ \wedge\ \ldots\ \wedge\ g_m(\vec{x}) \leq 0\ \Rightarrow\ g(\vec{x}) \leq 0\,.$$

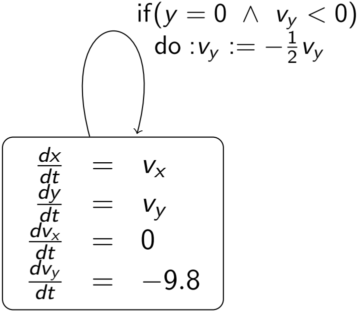
$$(\exists\ \lambda_1, \ldots, \lambda_m \geq 0)\ \lambda_1 g_1 + \cdots + \lambda_m g_m \equiv g$$

It is always sound to relax this way. *Complete* in the following cases:

1. $g_1, \ldots, g_m, g$ are affine functions (**Farkas' Lemma**).
2. $m = 1, g_1, g$ are positive-semidefinite quadratic forms (**S-Procedure**).
3. $g_1, \ldots, g_m, g$ are convex and satisfy some constraint qualifications.
4. ...

Example # 1: Bouncing Ball





Initial Conditions:

$$x(0) \in [0, 1], y(0) \in [2, 3], v_x(0) \in [1, 2], v_y(0) \in [-2, 2]$$

Ansatz

$$c_0 + c_1 x + c_2 y + c_3 v_x + c_4 v_y \leq 0$$

Encoding Initialization

Implication

$$(\forall\, x, y, v_x, v_y)\ \underbrace{\begin{bmatrix} x \in [0,1] \\ y \in [2,3], \\ v_x \in [1,2] \\ v_y \in [-2,2] \end{bmatrix}}_{\text{Initial Condition}} \Rightarrow \underbrace{c_0 + c_1 x + c_2 y + c_3 v_x + c_4 v_y \le 0}_{\text{Ansatz}}$$

Dualized Condition:

$$\left(\begin{array}{lcl} c_0 & \geq & 0\lambda_1 + 1\lambda_2 - 2\lambda_3 + 3\lambda_4 - \lambda_5 + 2\lambda_6 + 2\lambda_7 + 2\lambda_8 \\ c_1 & = & \lambda_2 - \lambda_1 \\ c_2 & = & \lambda_4 - \lambda_3 \\ c_3 & = & \lambda_6 - \lambda_5 \\ c_4 & = & \lambda_8 - \lambda_7 \\ \lambda_1, \ldots, \lambda_8 & \geq & 0 \end{array}\right)$$

Encoding Positive Invariance

Implication

$$(c_0+c_1x+c_2y+c_3v_x+c_4v_y=0) \Rightarrow \frac{d(c_0+c_1x+c_2y+c_3v_x+c_4v_y)}{dt} < 0$$

$$(c_0+c_1x+c_2y+c_3v_x+c_4v_y=0) \Rightarrow c_1v_x+c_2v_y-9.8c_4 < 0$$

Dualized Implication:

$$\begin{pmatrix} \mu c_0 & < & -9.8c_4 \\ \mu c_1 & = & 0 \\ \mu c_2 & = & 0 \\ \mu c_3 & = & c_1 \\ \mu c_4 & = & c_2 \\ \mu & \geq & 0 \end{pmatrix}$$

Combined Constraints

$$\left(\begin{array}{rcl}
c_0 & \geq & 0\lambda_1 + 1\lambda_2 - 2\lambda_3 + 3\lambda_4 - \lambda_5 + 2\lambda_6 + 2\lambda_7 + 2\lambda_8 \\
c_1 & = & \lambda_2 - \lambda_1 \\
c_2 & = & \lambda_4 - \lambda_3 \\
c_3 & = & \lambda_6 - \lambda_5 \\
c_4 & = & \lambda_8 - \lambda_7 \\
\lambda_1, \ldots, \lambda_8 & \geq & 0 \\
\mu c_0 & < & -9.8 c_4 \\
\mu c_1 & = & 0 \\
\mu c_2 & = & 0 \\
\mu c_3 & = & c_1 \\
\mu c_4 & = & c_2 \\
\mu & \geq & 0
\end{array}\right)$$

Solving Constraints

Solutions:

$$
\begin{array}{lcl}
c_0 = -2, c_1 = 0, c_2 = 0, c_3 = 0, c_4 = 1 & \rightarrow & v_y - 2 \leq 0 \\
c_0 = 1, c_1 = 0, c_2 = 0, c_3 = -1, c_4 = 0 & \rightarrow & 1 - v_x \leq 0
\end{array}
$$

Current Work

Nonlinear Systems:

1. Systems with polynomial dynamics.
2. Useful computational techniques from (semi-)algebraic geometry:

   Gröbner bases, Syzygies, Sum-of-Squared Programming
3. Ideas for linearization using F-related vector fields. [S.'11]

Relational Abstraction: Invariants that relate current state to some "future" state.

Use invariants to discretize a hybrid system. [S.+Tiwari'11]

Thank you! Questions?