Lecture 6
Verification of Hybrid Systems

Ufuk Topcu
Nok Wongpiromsarn Richard M. Murray

AFRL, 25 April 2012

Outline:
• A hybrid system model
• Finite-state abstractions and use of model checking
• Deductive verification and optimization-based construction of certificates
• Approximate bisimulation functions
Model and tools so far (in the course) help reason about discrete evolution of systems:
• does there exist a control sequence for which \( \varphi \) holds, or
• do all control sequences lead to executions for which \( \varphi \) holds with

\[
\varphi = \Box (\text{gear} = 1 \rightarrow \Diamond \text{gear} = 3) \, ?
\]

Need to modify to capture continuous evolution:
• Can the car come to a stop from any (reasonable) speed within \( x \) meters?
• How to efficiently accelerate?
Why to use hybrid system models?

- Continuous systems with multiple modes
- Discrete logic controlling continuous systems
- Continuous systems with “hybrid” specifications

\[
\begin{align*}
\min & \int_{t_0}^{T} L(x, u) dt \\
\text{s.t.} & \quad \dot{x} = f(x, u) \\
& \quad g(x, u) \leq 0
\end{align*}
\]
A (simple) hybrid system model

Hybrid system: \( H = (\mathcal{X}, L, X_0, I, F, T) \) with

- \( \mathcal{X} \), continuous state space;
- \( L \), finite set of locations (modes);
- Overall state space \( X = \mathcal{X} \times L \);
- \( X_0 \subseteq X \), set of initial states;
- \( I : L \rightarrow 2^\mathcal{X} \), invariant that maps \( l \in L \) to the set of possible continuous states while in location \( l \);
- \( F : X \rightarrow 2^{\mathbb{R}^n} \), set of vector fields, i.e., \( \dot{x} \in F(l, x) \);
- \( T \subseteq X \times X \), relation capturing discrete transitions between locations.
Specifications

Given: \( H = (\mathcal{X}, L, X_0, I, F, T) \)

Solution at time \( t \) with the initial condition \( x_0 \in X_0: \phi(t; x_0) \)

- With the simple model \( H \), specifying the initial state also specifies the initial mode.

Sample temporal properties:

- **Stability:** Given equilibrium \( x_e \in \mathcal{X} \), for all \( x_0 \in X_0 \subseteq \mathcal{X} \),
  \[
  \phi(t; x_0) \in \mathcal{X}, \ \forall t \text{ and } \phi(t; x_0) \to x_e, \ t \to \infty
  \]

- **Safety:** Given \( \mathcal{X}_{\text{unsafe}} \subseteq \mathcal{X} \), safety property holds if there exists no \( t_{\text{unsafe}} \) and trajectory with initial condition \( x_0 \in X_0 \),
  \[
  \phi(t_{\text{unsafe}}; x_0) \in \mathcal{X}_{\text{unsafe}}
  \]
  \[
  \phi(t; x_0) \in \mathcal{X}, \ \forall t \in [0, t_{\text{unsafe}}]
  \]

- **Reachability:** Given \( \mathcal{X}_{\text{reach}} \subseteq \mathcal{X} \), reachability property holds if there exists finite \( t_{\text{reach}} \geq 0 \) and a trajectory with initial condition \( x_0 \in X_0 \),
  \[
  \phi(t_{\text{reach}}; x_0) \in \mathcal{X}_{\text{reach}} \text{ and } \phi(t; x_0) \in \mathcal{X}, \ \forall t \in [0, t_{\text{reach}}]
  \]

- **Eventuality:** reachable from every initial condition

- Combinations of the above, e.g., starting in \( X_A \), reach both \( X_B \) and \( X_C \), but \( X_B \) will not be reached before \( X_C \) is reached while staying safe.
Verification of hybrid systems: Overview

Why not directly use model checking?

- Model checking applied to finite transitions systems
- exhaustively search for counterexamples....
  - if found, property does not hold.
  - if there is no counterexample in all possible executions, the property is verified.

Exhaustive search is not possible over continuous state spaces.

Approaches for hybrid system verification:

1. Construct finite-state approximations and apply model checking
   - preserve the meaning of the properties, i.e., proposition preserving partitions
   - use “over”- or “under”-approximations

2. Deductive verification
   - Construct Lyapunov-type certificates
   - Account for the discrete jumps in the construction of the certificate

3. Explicitly construct the set of reachable states
   - Limited classes of temporal properties (e.g., reachability and safety)
   - Not covered in this lecture
Finite-state, under- and over-approximations

Hybrid system: \( H = (\mathcal{X}, L, X_0, I, F, T) \)

Finite-transition system: \( TS = (Q, \to, Q_0) \)

Define the map \( T : Q \to 2^{\mathcal{X}} \)

For discrete state \( q \), \( T^{-1}(q) \) is the corresponding cell in \( \mathcal{X} \).

**Under-approximation:** \( TS \) is an under-approximation of \( H \) if the following two statements hold.

- Given \( q, q' \in Q \) with \( q \neq q' \), if \( q \to q' \), then for all \( x_0 \in T^{-1}(q) \), there exists finite \( \tau > 0 \) such that
  \[
  \phi(\tau; x_0) \in T^{-1}(q'), \quad \phi(t; x_0) \in T^{-1}(q) \cup T^{-1}(q'), \quad \forall t \in [0, \tau]
  \]
- If \( q \to q' \), then \( T^{-1}(q) \) is positively-invariant.

*In other words:*
- Every discrete trajectory in an under-approximation \( TS \) can be implemented by \( H \).
- \( TS \) “simulates” \( H \).

**Over-approximation:** \( TS \) is an over-approximation of \( H \), if for each discrete transition in \( TS \), there is a “possibility” to be implemented by \( H \).
- Possibility induced by the coarseness of the partition.
Use of under-approximations

Let the following be given.

- A hybrid system $H$,
- a finite-state, under-approximation $TS1$ for $H$,

**Verification**

- Let an LTL specification $\varphi$ be given.
- Question: $H \models \varphi$?
- Model check “$TS1 \models \varphi$?”

| $\text{Words}(\neg \varphi) \cap \text{Trace}(TS1)$ is nonempty | $H$ cannot satisfy the specification. |
| $\downarrow$ | $TS1 \not\models \varphi$ |
| $\text{Words}(\neg \varphi) \cap \text{Trace}(H)$ is nonempty | $H \not\models \varphi$ |

| $\text{Words}(\neg \varphi) \cap \text{Trace}(TS1)$ is empty | Inconclusive |

**Logic synthesis:**

- If $\text{Words}(\varphi) \cap \text{Trace}(TS1)$ is nonempty, there exists a trajectory of $TS1$ which satisfies $\varphi$ and can be implemented by $H$.
- Otherwise, inconclusive.
Use of over-approximations

Hybrid system $H$ and a finite-state, over-approximation $TS2$ for $H$.

**Verification**

- $\text{Words}(\varphi) \cap \text{Trace}(TS2)$ is nonempty
  - Inconclusive

- $\text{Words}(\neg\varphi) \cap \text{Trace}(TS2)$ is empty
  - $\Downarrow$
  - $\text{Words}(\neg\varphi) \cap \text{Trace}(H)$ is empty

$H$ satisfies the specification.

$TS2 \models \varphi$

$\Downarrow$

$H \models \varphi$

**Logic synthesis:**

- If $\text{Words}(\varphi) \cap \text{Trace}(TS2)$ is empty, no valid trajectories for $TS2$ or $H$.
- Otherwise, inconclusive.

**Remarks:**

- Under- and over-approximations give partial results.
- Potential remedies:
  - Finer approximations
  - Bisimulations
Example: verification

System models:
Continuous vector field:

Discrete over-approximation:
(small dots: self transitions)

Specifications:
\[
\phi_1 = (x_a < \theta_a^1 \land x_b > \theta_b^2 \rightarrow \square (x_a < \theta_a^1 \land x_b > \theta_b^2)) \land (x_b < \theta_b^1 \land x_a > \theta_a^2 \rightarrow \square (x_b < \theta_b^1 \land x_a > \theta_a^2))
\]
\[
\phi_2 = \Diamond (x_a < \theta_a^2 \lor x_b < \theta_b^2)
\]

Both hold for the over-approximation; hence, they hold for the actual system.

Verification of hybrid systems: Overview

Why not directly use model checking?
• Model checking applied to finite transitions systems
• exhaustively search for counterexamples....
  • if found, property does not hold.
  • if there is no counterexample in all possible executions, the property is verified.

Exhaustive search is not possible over continuous state spaces.

Approaches for hybrid system verification:

1. Construct finite-state approximations and apply model checking
   • preserve the meaning of the properties, i.e., proposition preserving partitions
   • use “over”- or “under”-approximations

2. Deductive verification
   • Construct Lyapunov-type certificates
   • Account for the discrete jumps in the construction of the certificate

3. Explicitly construct the set of reachable states
   • Limited classes of temporal properties (e.g., reachability and safety)
   • Not covered in this lecture
What does deductive verification mean?

Example with continuous, nonlinear dynamics:

\[ \dot{x}(t) = f(x(t)) \]

where \( x(t) \in \mathbb{R}^n, \ f(0) = 0, \ x = 0 \) is an asymptotically stable equilibrium.

Region-of-attraction: \( \mathcal{R} := \left\{ x : \lim_{t \to \infty} \phi(t; x) = 0 \right\} \)

**Question 1** (a system analysis question):
Given \( S \subset \mathbb{R}^n \), is \( S \) invariant and \( S \subseteq \mathcal{R} \)?

the question we **want** to answer

the question we **attempt** to answer

**Question 2** (an algebraic question):

Does there exist a continuously differentiable function \( V : \mathbb{R}^n \to \mathbb{R} \) such that

- \( V \) is positive definite,
- \( V(0) = 0 \),
- \( \Omega := \{ x : V(x) \leq 1 \} \subset \{ x : \nabla V \cdot f(x) < 0 \} \cup \{0\} \)
- \( S \subseteq \Omega \)?

Yes to Question 2 \( \rightarrow \) Yes to Question 1.
Barrier Certificates - Safety

Safety property holds if there exists no $T \geq 0$ and trajectory such that:

$$x = \phi(0; x) \in \mathcal{X}_{\text{initial}}$$
$$\phi(T; x) \in \mathcal{X}_{\text{unsafe}}$$
$$\phi(t; x) \in \mathcal{X} \forall t \in [0, T].$$

**Continuous dynamics:**

$$\dot{x}(t) = f(x(t))$$

Suppose there exists a differentiable function $B$ such that

$$B(x) \leq 0, \forall x \in \mathcal{X}_{\text{initial}}$$
$$B(x) > 0, \forall x \in \mathcal{X}_{\text{unsafe}}$$
$$\frac{\partial B}{\partial x} f(x) \leq 0, \forall x \in \mathcal{X}.$$  

Then, the safety property holds.

**Hybrid dynamics:**

$$H = (\mathcal{X}, L, X_0, I, F, T)$$

Suppose there exist differentiable functions $B_l$ (for each mode) such that

$$B_l(x) \leq 0, \forall x \in I(l) \cap \mathcal{X}_{\text{initial}}$$
$$B_l(x) > 0, \forall x \in I(l) \cap \mathcal{X}_{\text{unsafe}}$$
$$\frac{\partial B_l}{\partial x} F(x) \leq 0, \forall x \in I(l)$$
$$B_{l'}(x') - B_l(x) \leq 0, \text{ for each jump} \quad (l, x) \rightarrow (l', x')$$

Then, the safety property holds.
Barrier Certificates - Eventuality

\[ \dot{x}(t) = f(x(t)) \]

Eventuality property holds if for all \( x_0 \in X_{\text{initial}} \),

\[ \phi(T; x_0) \in X_{\text{target}} \]

\[ \phi(t; x_0) \in X, \ \forall t \in [0, T] \]

for some non-negative \( T \).

\( X, X_{\text{target}}, X_{\text{initial}} \) are bounded

don’t leave \( X \) before reaching \( X_{\text{target}} \)
leave \( X \backslash X_{\text{target}} \) in finite time

Suppose that \( f \) is continuously differentiable and there exists a continuously differentiable function \( B \) such that

\[ B(x) \leq 0, \ \forall x \in X_{\text{initial}} \]

\[ B(x) > 0, \ \forall x \in \partial X \backslash \partial X_{\text{target}} \]

\[ \frac{\partial B}{\partial x}(x) \cdot f(x) < 0, \ \forall x \in X \backslash X_{\text{target}} \]

Then, the eventuality property holds.

- Straightforward extensions for hybrid dynamics as in safety verification are possible.
If system starts in $\mathcal{X}_A$, then both $\mathcal{X}_B$ and $\mathcal{X}_C$ are reached in finite time, but $\mathcal{X}_C$ will not be reached before system reaches $\mathcal{X}_B$. 

\[
\begin{cases}
B_1(x) \leq 0 & \forall x \in \mathcal{X}_A, \\
B_1(x) > 0 & \forall x \in \partial \mathcal{X} \cup \mathcal{X}_C, \\
\frac{\partial B_1}{\partial x}(x) f(x, d) \leq -\epsilon & \forall (x, d) \in (\mathcal{X} \setminus \mathcal{X}_B) \times D,
\end{cases}
\]

\[
\begin{cases}
B_2(x) \leq 0 & \forall x \in \mathcal{X}_A, \\
B_2(x) > 0 & \forall x \in \partial \mathcal{X}, \\
\frac{\partial B_2}{\partial x}(x) f(x, d) \leq -\epsilon & \forall x \in (\mathcal{X} \setminus \mathcal{X}_C) \times D.
\end{cases}
\]
How to construct the certificates?

- System properties $\rightarrow$ Algebraic conditions
  - Lyapunov, dissipation inequalities.

- Algebraic conditions $\rightarrow$ Numerical optimization problems
  - Restrict the attention to polynomial vector fields, polynomial certificates,...
  - S-procedure like conditions (for set containment constraints)
  - Sum-of-squares (SOS) relaxations for polynomial nonnegativity
  - Pass to semidefinite programming (SDP) that are equivalent of SOS conditions

- Solve the resulting (linear or “bilinear”) SDPs

- Construct polynomial certificates

Problem-dependent!
Some preliminaries

• Semidefinite programming problems
• Positive semidefinite polynomials and sum-of-squares (SOS) programming
• Set containment conditions and S-procedure
Linear and bilinear matrix inequalities

Convex, efficient, general-purpose solvers exist

Non-convex, no efficient, general-purpose solvers

Given matrices \( \{F_i\}_{i=0}^{N} \subset S^{n \times n} \), Linear Matrix Inequality (LMI) is a constraint on \( \lambda \in \mathbb{R}^N \) of the form:

\[
F_0 + \sum_{k=1}^{N} \lambda_k F_k \succeq 0
\]

Given matrices \( \{F_i\}_{i=0}^{N}, \{G_j\}_{j=1}^{M}, \text{ and } \{H_{k,j}\}_{k=1}^{N_{k=1}}_{j=1}^{M} \subset S^{n \times n} \), a Bilinear Matrix Inequality (BMI) is a constraint on \( \lambda \in \mathbb{R}^N \) and \( \gamma \in \mathbb{R}^M \) of the form:

\[
F_0 + \sum_{k=1}^{N} \lambda_k F_k + \sum_{j=1}^{M} \gamma_j G_j + \sum_{k=1}^{N} \sum_{j=1}^{M} \lambda_k \gamma_j H_{k,j} \succeq 0
\]

Semidefinite program (SDP)

very roughly speaking, optimization of affine objective subject to LMI or/and BMI constraints
Polynomials and Multipoly Toolbox

Given $\alpha \in \mathbb{N}^n$, a monomial in $n$ variables is a function $m_\alpha : \mathbb{R}^n \rightarrow \mathbb{R}$ defined as $m_\alpha(x) := x_1^{\alpha_1} x_2^{\alpha_2} \cdots x_n^{\alpha_n}$.

The degree of a monomial is defined as $\deg m_\alpha := \sum_{i=1}^n \alpha_i$.

Polynomial: Finite linear combination of monomials.

$$p := \sum_{\alpha \in \mathcal{A}} c_\alpha m_\alpha = \sum_{\alpha \in \mathcal{A}} c_\alpha x^\alpha$$

where $\mathcal{A} \subset \mathbb{N}^n$ is a finite set and $c_\alpha \in \mathbb{R}$ $\forall \alpha \in \mathcal{A}$.

Multipoly is a Matlab toolbox for the creation and manipulation of polynomials of one or more variables.

Example:

```
p-var x1 x2
p = 2*x1^4 + 2*x1^3*x2 - x1^2*x2^2 + 5*x2^4
q = x1^2
p*q =
    2*x1^6 + 2*x1^5*x2 - x1^4*x2^2 + 5*x1^2*x2^4
jacobian(p, [x1;x2]) =
    [ 8*x1^3 + 6*x1^2*x2 - 2*x1*x2^2 ,
        2*x1^3 - 2*x1^2*x2 + 20*x2^3 ]
```
Positive semidefinite polynomials

\( \mathbb{R}[x_1, \ldots, x_n] \) or \( \mathbb{R}[x] \) denotes the set of polynomials (with real coefficients) in the variables \( \{x_1, \ldots, x_n\} \).

\( p \in \mathbb{R}[x] \) is positive semi-definite (PSD) if \( p(x) \geq 0 \) \( \forall x \). The set of PSD polynomials in \( n \) variables \( \{x_1, \ldots, x_n\} \) will be denoted \( \mathcal{P}[x_1, \ldots, x_n] \) or \( \mathcal{P}[x] \).

Testing if \( p \in \mathcal{P}[x] \) is NP-hard when the polynomial degree is at least four.

How about a quadratic polynomial?

**Sum-of-Squares Polynomials**

A polynomial \( p \) is a **sum of squares (SOS)** if there exist polynomials \( \{ f_i \}_{i=1}^N \) such that

\[
p = \sum_{i=1}^N f_i^2.
\]

The set of SOS polynomials in \( n \) variables \( \{ x_1, \ldots, x_n \} \) will be denoted \( \Sigma [x_1, \ldots, x_n] \) or \( \Sigma [x] \).

If \( p \) is a SOS then \( p \) is PSD.

For every polynomial \( p \) of degree \( 2d \), there exists a symmetric matrix \( Q \) such that

\[
p(x) = z(x)^T Q z(x)
\]

with

\[
z(x) := [1, x_1, \ldots, x_n, x_1^2, x_1x_2, \ldots, x_n^2, \ldots, x_n^d]^T
\]

\( p \) is SOS if and only if there exists \( Q \succeq 0 \) s.t.

\[
p(x) = z(x)^T Q z(x)
\]

Given \( p \), can be verified as SDP.
SOS example

All possible Gram matrix representations of

\[ p(x) = 2x_1^4 + 2x_1^3x_2 - x_1^2x_2^2 + 5x_2^4 \]

are given by \( z^T (Q + \lambda N) z \) where:

\[
\begin{align*}
    z &= \begin{bmatrix} x_1^2 \\ x_1x_2 \\ x_2^2 \end{bmatrix}, &
    Q &= \begin{bmatrix} 2 & 1 & -0.5 \\ 1 & 0 & 0 \\ -0.5 & 0 & 5 \end{bmatrix}, &
    N &= \begin{bmatrix} 0 & 0 & -0.5 \\ 0 & 1 & 0 \\ -0.5 & 0 & 0 \end{bmatrix}
\end{align*}
\]

\[ p(x) = z(x)^T Q z(x) \]

\[ 0 = z(x)^T N z(x) \]

\[ (x_1x_2) \cdot (x_1x_2) = x_1^2 \cdot x_2^2 \]

\[ (x_1x_2) \cdot (x_1x_2) = x_1^2 \cdot x_2^2 \]

\[ p \text{ is SOS iff } Q + \lambda N \succeq 0 \]

for some \( \lambda \in \mathbb{R} \).
SOS programming

SOS Programming: Given $c \in \mathbb{R}^m$ and polynomials $\{f_{j,k}\}_{j=1}^{N_s} \quad m$, solve:

$$\min_{\alpha \in \mathbb{R}^m} c^T \alpha$$

subject to:

$$f_{1,0}(x) + f_{1,1}(x)\alpha_1 + \cdots + f_{1,m}(x)\alpha_m \in \Sigma [x]$$

$$\vdots$$

$$f_{N_s,0}(x) + f_{N_s,1}(x)\alpha_1 + \cdots + f_{N_s,m}(x)\alpha_m \in \Sigma [x]$$

There is freely available software (e.g. SOSTOOLS, YALMIP, SOSOPT) that:

1. Converts the SOS program to an SDP
2. Solves the SDP with available SDP codes (e.g. Sedumi)
3. Converts the SDP results back into polynomial solutions
Set containment conditions

Given polynomials $g_1$ and $g_2$, define sets $S_1$ and $S_2$:

$$S_1 := \{ x \in \mathbb{R}^n : g_1(x) \leq 0 \}$$
$$S_2 := \{ x \in \mathbb{R}^n : g_2(x) \leq 0 \}$$

Is $S_2 \subseteq S_1$?

**Polynomial S-procedure**

$$\exists \lambda \in \Sigma[x] \text{ s.t. } -g_1(x) + \lambda(x)g_2(x) \in \Sigma[x]$$

$$\Downarrow$$

$$\exists \lambda \text{ positive semidefinite polynomial s.t. } -g_1(x) + \lambda(x)g_2(x) \geq 0 \ \forall x$$

$$\Downarrow$$

$$\{ x : g_2(x) \leq 0 \} \subseteq \{ x : g_1(x) \leq 0 \}$$

**Example:**

$B(x) \leq 0, \ \forall x \in \mathcal{X}_{\text{initial}}$

Suppose $\mathcal{X}_{\text{initial}} = \{ x : g(x) \leq 0 \}$ for some $g$

**Sufficient condition:** There exists positive semidefinite function $s$ such that

$$-B(x) + s(x)g(x) = -B(x) - s(x)(-g(x)) \geq 0, \ \forall x \in \mathbb{R}^n$$
Global stability theorem

**Theorem:** Let \( l_1, l_2 \in \mathbb{R}[x] \) satisfy \( l_i(0) = 0 \) and \( l_i(x) > 0 \ \forall x \neq 0 \) for \( i = 1, 2 \). If there exists \( V \in \mathbb{R}[x] \) such that:

- \( V(0) = 0 \)
- \( V - l_1 \in \Sigma[x] \)
- \( -\nabla V \cdot f - l_2 \in \Sigma[x] \)

Then \( R_0 = \mathbb{R}^n \).

(Refer to Section 5.3 for theorems on Lyapunov's direct method.)
Global stability examples with sosopt

% Code from Parrilo1_GlobalStabilityWithVec.m

% Create vector field for dynamics
pvar x1 x2;
x = [x1;x2];
x1dot = -x1 - 2*x2^2;
x2dot = -x2 - x1*x2 - 2*x2^3;
xdot = [x1dot; x2dot];

% Use sosopt to find a Lyapunov function
% that proves x = 0 is GAS

% Define decision variable for quadratic
% Lyapunov function
zV = monomials(x,2);
V = polydecvar(’c’,zV,‘vec’);

% Constraint 1: V(x) - L1 \in SOS
L1 = 1e-6 * ( x1^2 + x2^2 );
sosconstr{1} = V - L1;

% Constraint 2: -Vdot - L2 \in SOS
L2 = 1e-6 * ( x1^2 + x2^2 );
Vdot = jacobian(V,x)*xdot;
sosconstr{2} = -Vdot - L2;

% Solve with feasibility problem
[info,dopt,sosol] = sosopt(sosconstr,x);
Vsol = subs(V,dopt)
Vsol =
0.30089*x1^2 + 1.8228e-017*x1*x2 + 0.6018*x2^2
Approximate bisimulation relations & bisimulation functions

Two systems with \( x_i \in \mathbb{R}^{n_i}, x_i(0) \in I_i \subseteq \mathbb{R}^{n_i}, u_i(t) \in U_i \subseteq \mathbb{R}^{m_i}, y_i \in \mathbb{R}^p \)

\[
\Phi_1 : \begin{cases} 
\dot{x}_1(t) = f_1(x_1(t), u_1(t)) \\
y_1(t) = g_1(x_1(t))
\end{cases} \quad \Phi_2 : \begin{cases} 
\dot{x}_2(t) = f_2(x_2(t), u_2(t)) \\
y_2(t) = g_2(x_2(t))
\end{cases}
\]

A relation \( R_\delta \in \mathbb{R}^{m_1} \times \mathbb{R}^{n_2} \) is a \( \delta \)-approximate bisimulation relation between \( \Phi_1 \) and \( \Phi_2 \) if for all \( (x_1, x_2) \in R_\delta \):

\[
\begin{align*}
&\cdot \|g_1(x_1) - g_2(x_2)\| \leq \delta; \\
&\cdot \forall T > 0 \text{ and } \forall u_1(\cdot), \exists u_2(\cdot) \text{ s.t. } (\phi_1(t; x_1) - \phi_2(t; x_2)) \in R_\delta \forall t \in [0, T]; \\
&\cdot \forall T > 0 \text{ and } \forall u_2(\cdot), \exists u_1(\cdot) \text{ s.t. } (\phi_1(t; x_1) - \phi_2(t; x_2)) \in R_\delta \forall t \in [0, T].
\end{align*}
\]

If start in relation, stay in relation. Observations are “close.”

A function \( V : \mathbb{R}^{n_1} \times \mathbb{R}^{n_2} \rightarrow \mathbb{R}^+ \cup \{+\infty\} \) is a bisimulation function between \( \Phi_1 \) and \( \Phi_2 \) if for all \( \delta \geq 0 \):

\[
R_\delta = \{(x_1, x_2) \in \mathbb{R}^{n_1} \times \mathbb{R}^{n_2} : V(x_1, x_2) \leq \delta\}
\]

is a closed set and a \( \delta \)-approximate bisimulation relation between \( \Phi_1 \) and \( \Phi_2 \).
Approximate bisimulation relations & bisimulation functions

Two systems with \( x_i \in \mathbb{R}^{n_i}, \ x_i(0) \in I_i \subseteq \mathbb{R}^{n_i}, \ u_i(t) \in U_i \subseteq R^{m_i}, \ y_i \in \mathbb{R}^p \)

\[
\Phi_1 : \begin{cases} 
    \dot{x}_1(t) = f_1(x_1(t), u_1(t)) \\
    y_1(t) = g_1(x_1(t))
\end{cases} \quad \Phi_2 : \begin{cases} 
    \dot{x}_2(t) = f_2(x_2(t), u_2(t)) \\
    y_2(t) = g_2(x_2(t))
\end{cases}
\]

Let \( W : \mathbb{R}^{n_1} \times \mathbb{R}^{n_2} \rightarrow \mathbb{R}^+ \) be a continuously differentiable function. If for all \((x_1, x_2) \in \mathbb{R}^{n_1} \times \mathbb{R}^{n_2}, \)

\[
W(x_1, x_2) \geq \|g_1(x_1) - g_2(x_2)\|^2
\]

\[
\frac{\partial W}{\partial x_1} f_1(x_1, u_1) - \frac{\partial W}{\partial x_2} f_2(x_2, u_2) \leq 0, \ \forall (x_1, x_2) \in \mathbb{R}^{n_1} \times \mathbb{R}^{n_2}, \ u_1 \in \mathbb{R}^{m_1}, \ u_2 \in \mathbb{R}^{m_2}
\]

then \( V := |\sqrt{W}| \) is a bisimulation function between \( \Phi_1 \) and \( \Phi_2. \)

\[
\text{guarantees that no matter what} \ u_1 \ \text{and} \ u_2 \ \text{do, the time derivative of} \ W \ \text{stays non-positive}
\]
Approximate bisimulations + safety

\[ \Phi_1 \text{ s.t. } x_1 \in \mathbb{R}^{10} \]
\[ \Phi_2 \text{ s.t. } x_2 \in \mathbb{R}^4 \]
\[ \Phi_3 \text{ s.t. } x_3 \in \mathbb{R}^6 \]

\( \Phi_1 \) is 1.90-approximate bisimilar to \( \Phi_2 \)
\( \Phi_1 \) is 0.76-approximate bisimilar to \( \Phi_3 \)