Basic algorithms and recent advances on safe control synthesis with reachability and invariance

Necmiye Ozay, EECS
University of Michigan, Ann Arbor

44th International Summer School of Automatic Control
Grenoble, France

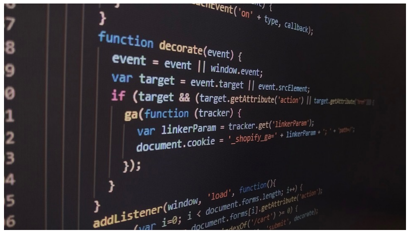
August 28 - September 01, 2023

Mostly based on joint work with **Liren Yang** (Huazhong University of Science and Technology)

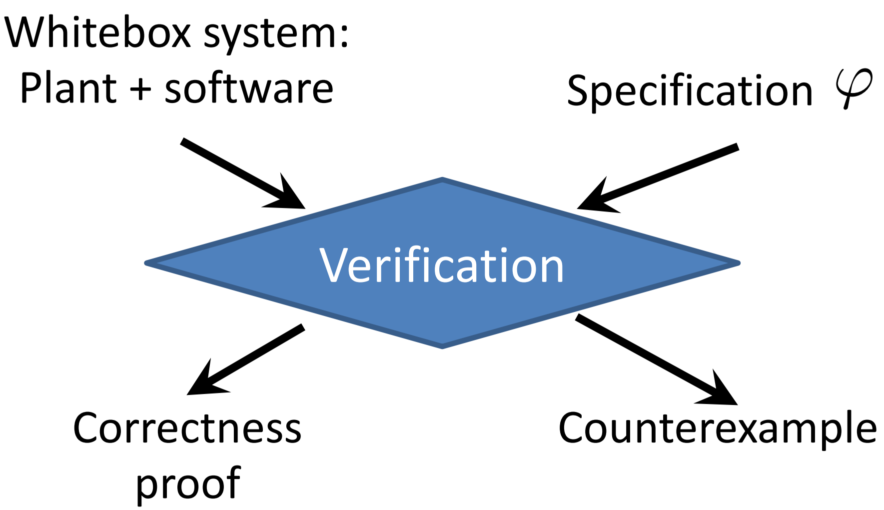
Research funded in part by

V&V landscape

Plant
(with a model)



Autonomy
software



PROS:

- Strong guarantees

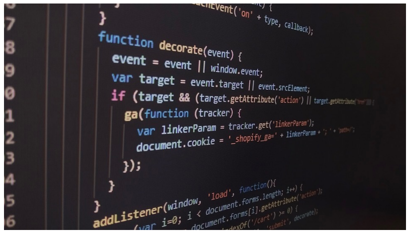
CONS:

- Hybrid system verification is computationally very hard
- Autonomy software ➔ up to millions of lines of code (loc):
  - hard to model
  - hard to scale

Mars curiosity rover: 5M loc
Boeing 787 flight software: 15M loc
F 35-fighter jet: 25M loc
Average modern high-end car: 100M loc

V&V landscape

+

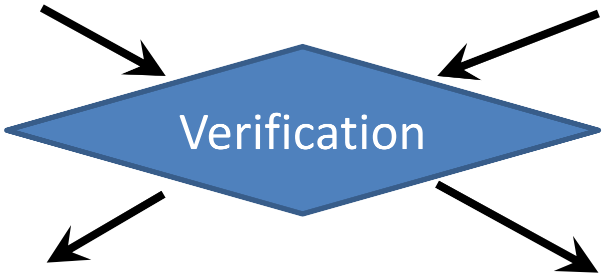


Plant
(with a model)

Autonomy
software

Whitebox system:
Plant + software

Specification φ



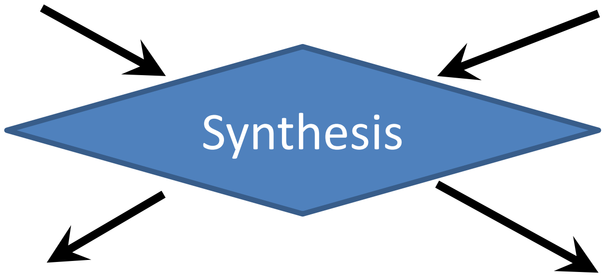
Correctness
proof

Counterexample

Alternative #1: Correct-by-construction (control) synthesis

Partial whitebox system:
Plant (∃? Software)

Specification



Software with
correctness guarantee
+ validity domain

Proof of
impossibility

PROS:

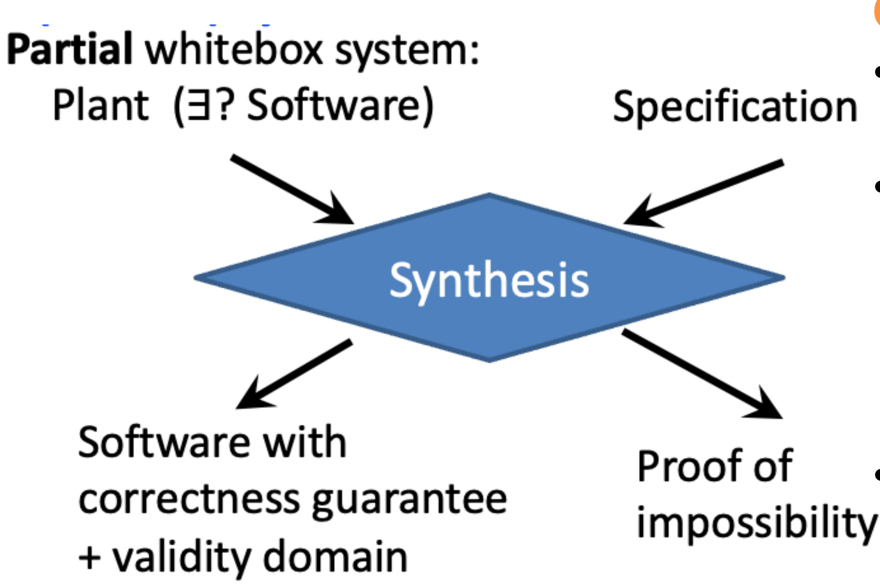
- Strong guarantees
- Avoids the complexity induced by software
- “Explains” fundamental limits (impossibility)

CONS:

- Correct-by-construction synthesis is computationally even harder
- Limited specs; spec correctness & completeness is more crucial
- Almost no synthesis approach for perception or learning components

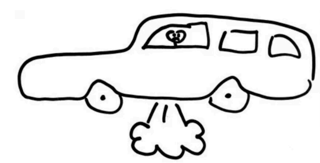
Deployment of synthesized controllers in Mcity

Alternative #1: Correct-by-construction (control) synthesis



What we learned from early deployments?

- Putting "correct" and automatically synthesized software on a car is feasible
- There were failures but having mathematical models and formal assumptions help detect failures
  - We realized that the model was missing actuator delays
- Conservativeness due to not looking ahead



- Motivated future work on safety with learned models, delays, and predictions

Alternative #1: Correct-by-construction (control) synthesis

$$\Sigma\text{: } x(t+1) = Ax(t) + Bu(t-\boldsymbol{\tau_d}) + Fd(t)$$

$$\Sigma_{\text{aug}}\text{: } \begin{cases} x(t+1) = Ax(t) + Bu_1(t) + Fd(t) \\ u_1(t+1) = u_2(t) \\ u_2(t+1) = u_3(t) \\ \quad\quad \dots \\ u_{\boldsymbol{\tau_d}}(t+1) = u(t) \end{cases}$$

$$S_{aug} = S \times U^{\tau_d}$$

- Goal: Find a set $C_{aug} \in S_{aug}$ (ideally the maximal such C_{aug}) such that if $x_{aug} \in C_{aug}$ then there exist $u \in U$ such that $x_{aug}^+ \in C_{aug}$ (C_{aug} is a controlled invariant set).

Alternative #1: Correct-by-construction (control) synthesis

$$\Sigma:\ x(t+1) = Ax(t) + Bu(t-\tau_d) + Fd(t)$$

$$\boxed{\begin{aligned} &T(x, u_1, \ldots, u_{\tau_d}) \\ &= A^{\tau_d}x + \Sigma_{i=1}^{\tau_d} A^{i-1}Bu_{\tau_d - i+1} \end{aligned}}$$

$$\Sigma_{\text{aug}}: \begin{cases} x(t+1) = Ax(t) + Bu_1(t) + Fd(t) \\ u_1(t+1) = u_2(t) \\ u_2(t+1) = u_3(t) \\ \quad\quad \ldots \\ u_{\tau_d}(t+1) = u(t) \end{cases}$$

$$S_{aug} = S \times U^{\tau_d}$$

Σ_{aug}, S_{aug} Dim= $n + m \cdot \tau_d$ → Direct method ✗ → C_{aug}

T

1. Dimension reduction

$$\boxed{\Sigma_{aux}:\ \hat{x}_{\tau_d}(t+1) = A\hat{x}_{\tau_d}(t) + Bu(t) + A^{\tau}Fd(t).}$$

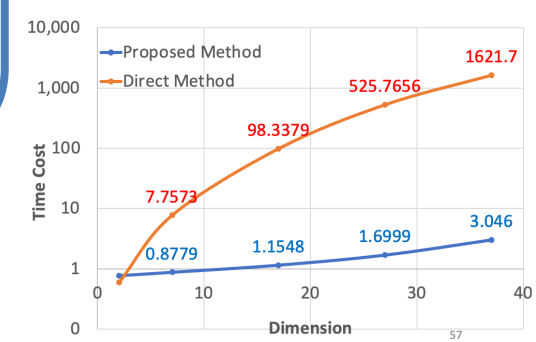
Σ_{aux}, S_{aux} Dim= n → C_{aux} Dim= n $\xrightarrow{T^{-1}}$ C_{lift} Dim= $n + m \cdot \tau_d$

$C_{aug} = C_{lift}$

2. Compute maximal invariant set of Σ_{aux}

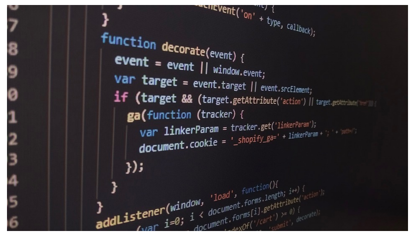
3. Lifting

Theorem: *Any controlled invariant set of the $n + m \cdot \tau_d$ dimensional system Σ_{aug} can be obtained from that of the n dimensional auxiliary system Σ_{aux}.*



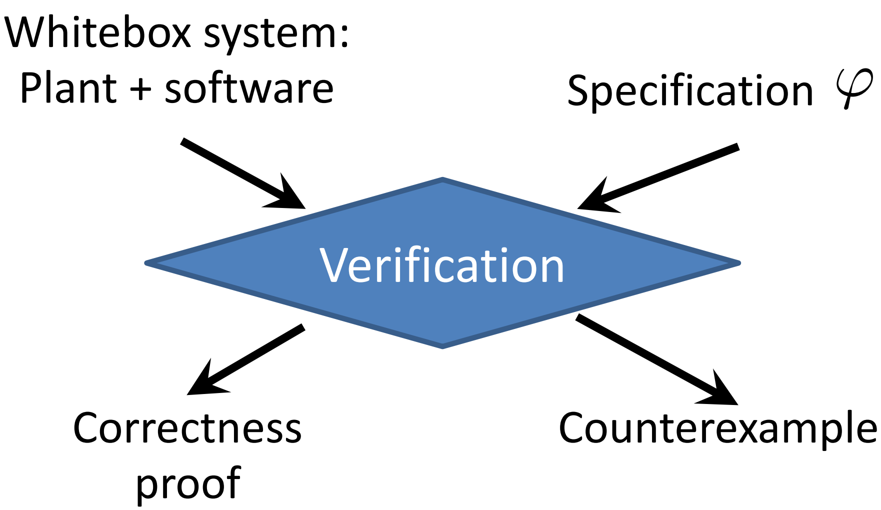
V&V landscape

+

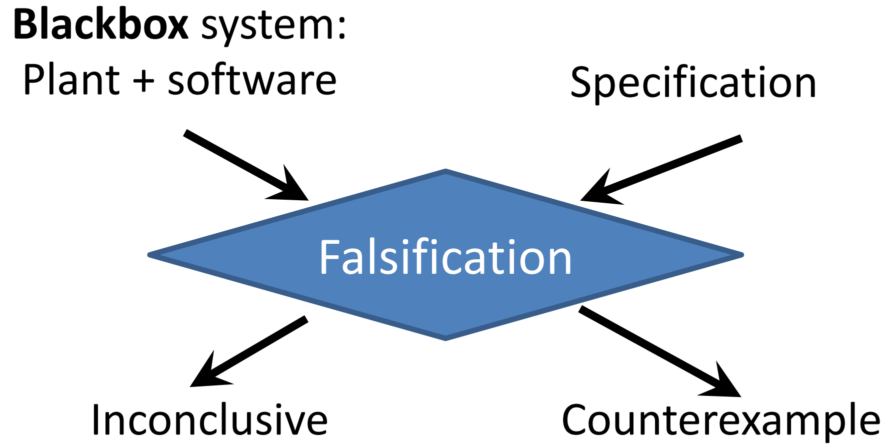


Plant
(with a model)

Autonomy
software



Alternative #2: Falsification



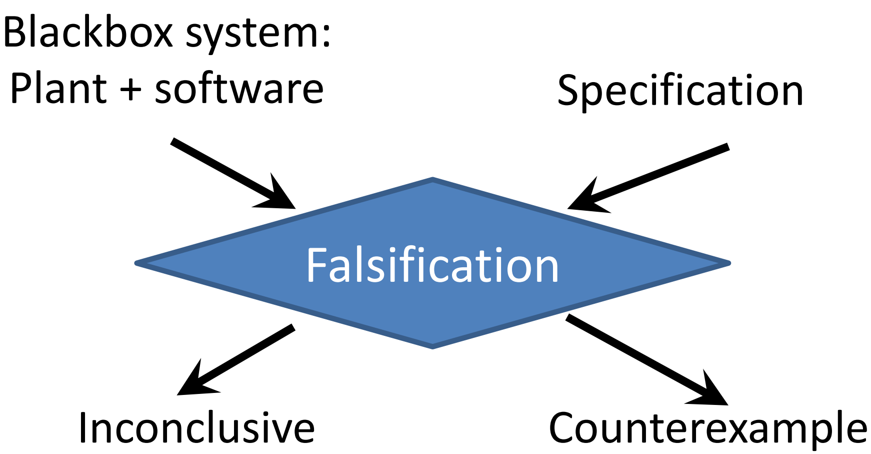
PROS:

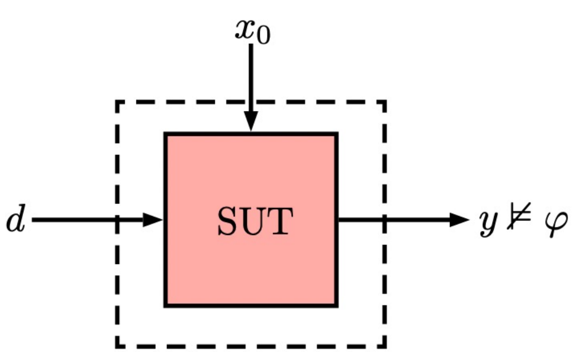
- Can handle arbitrarily complex models (plant + software) including learning-based components
- Industry-adopted tools (e.g., Breach, S-Taliro)

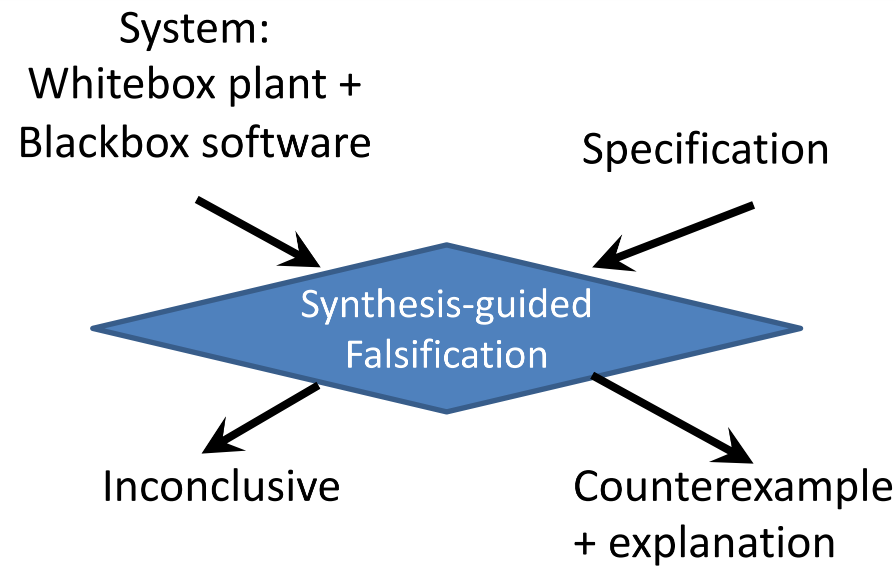
CONS:

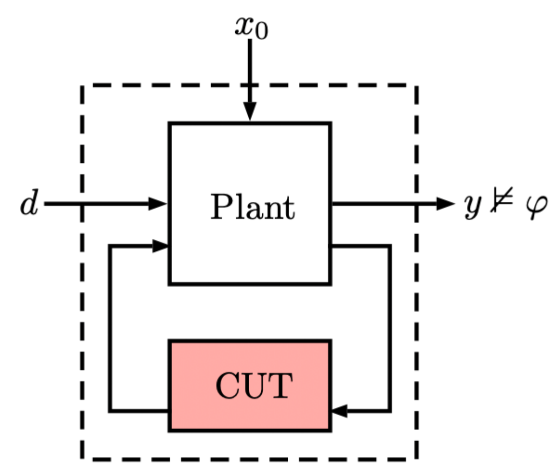
- Weaker conclusions
- No explanation of the counterexamples:
  - can give "trivial" counterexamples if assumptions are not modeled carefully
  - is it a hardware (plant) limitation or software bug?

Alternative #2.5: Synthesis-guided falsification



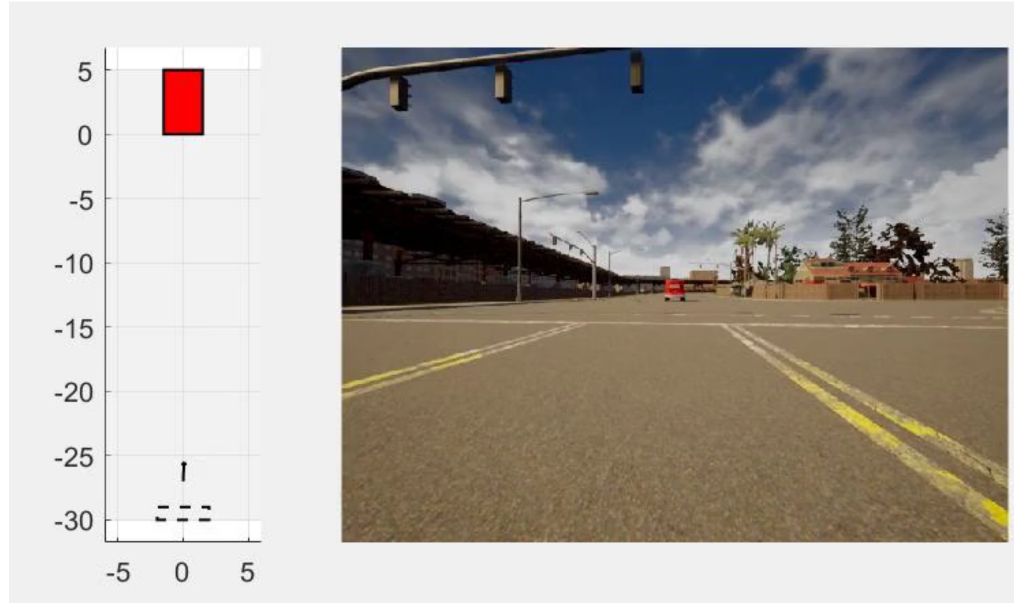


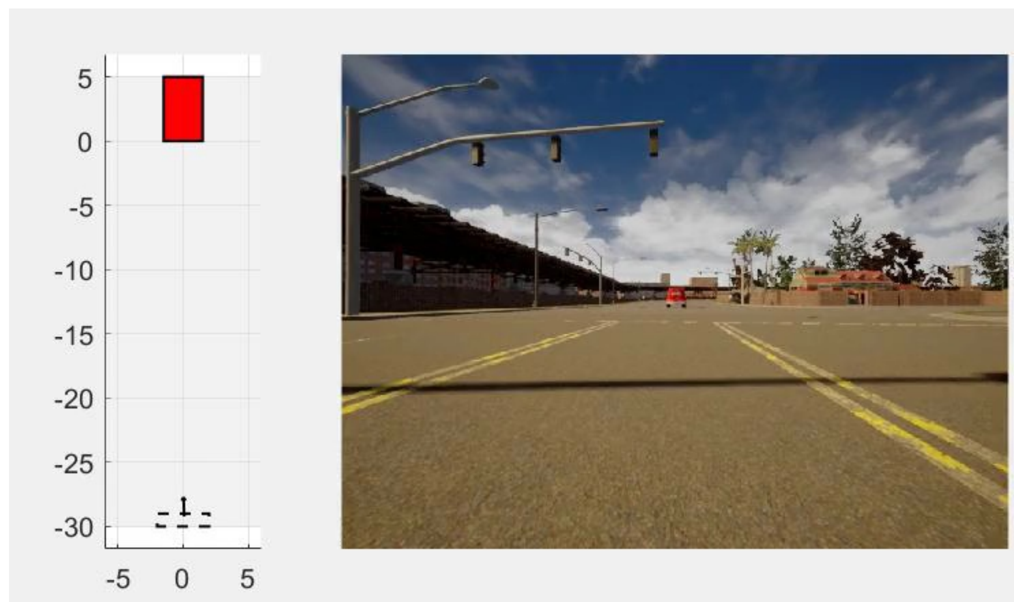




* SUT: system under test; CUT: controller (autonomy software) under test

a typical result with our falsification algorithm





Underlying tool in verification:

Forward reachable sets (FRS)

Underlying tool in synthesis and synthesis-guided falsification:

Backward reachable sets (BRS)

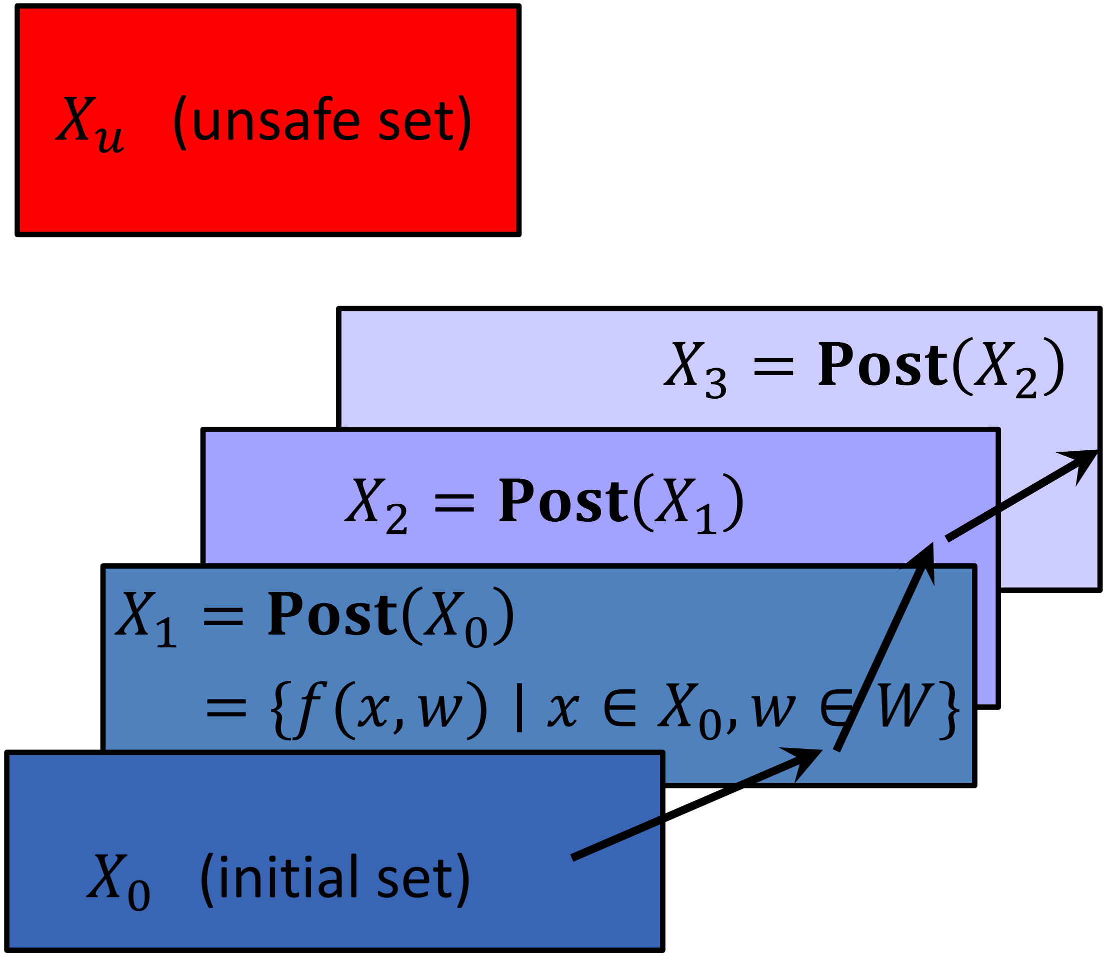
Forward reachable sets

Closed-loop system:

$$x_{t+1} = f(x_t, w_t)$$
$$w_t \in W$$

Typical **verification** problem:

Given a set of initial states X_0, an unsafe set X_u, and a time horizon T, prove or disprove that for all $x_0 \in X_0$, for all $t \in [0, T]$ and for all $w_{0:T-1} \in W^T$, we have $x_t \notin X_u$.



Reachability [Girard 2005, Kurzhansky & Varaiya 2011]
Also used in constructing symbolic models (abstractions)

Forward reachable sets

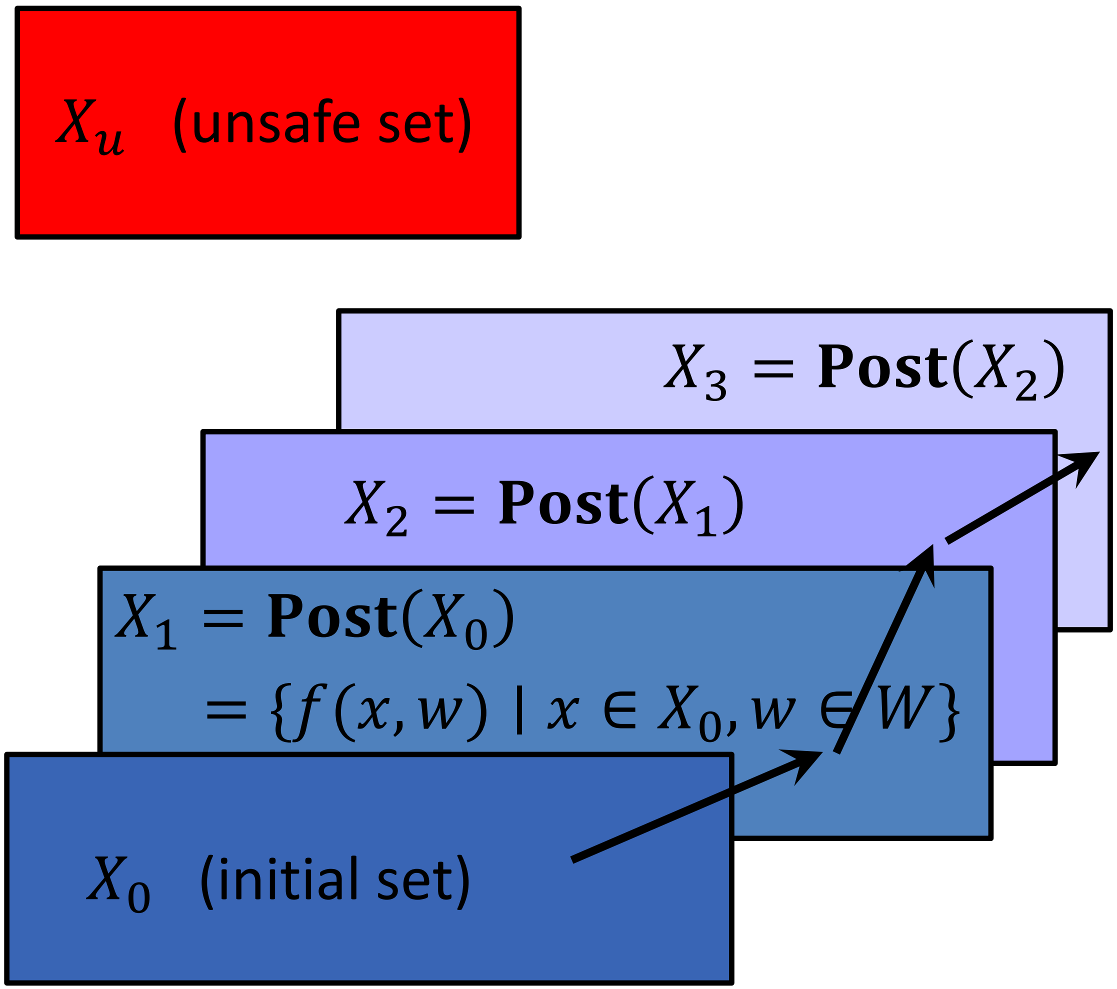
Closed-loop system:

$$x_{t+1} = f(x_t, w_t)$$
$$w_t \in W$$

Typical **verification** problem:

Given a set of initial states X_0, an unsafe set X_u, and a time horizon T, prove or disprove that for all $x_0 \in X_0$, for all $t \in [0, T]$ and for all $w_{0:T-1} \in W^T$, we have $x_t \notin X_u$.

Outer-approximations can be used to prove safety



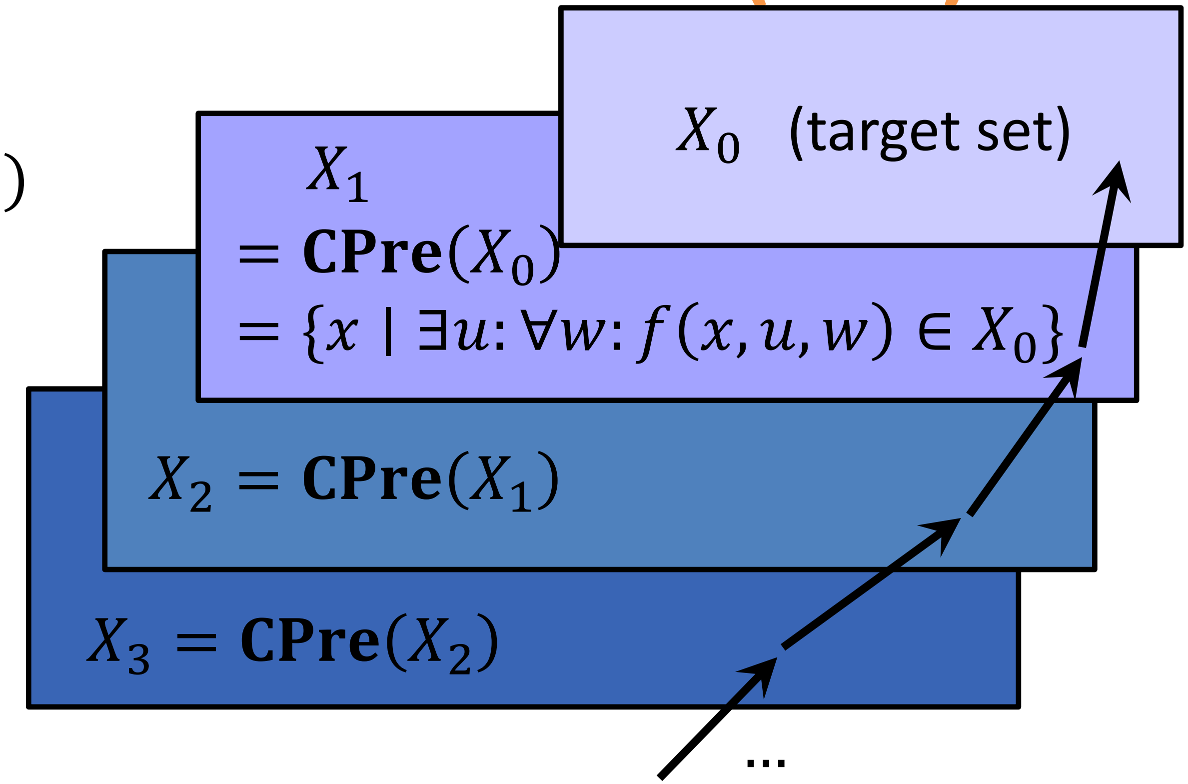
Backward Reachable Sets (BRS)

Control system:

$$x_{t+1} = f(x_t, u_t, w_t)$$

$$u_t \in U$$

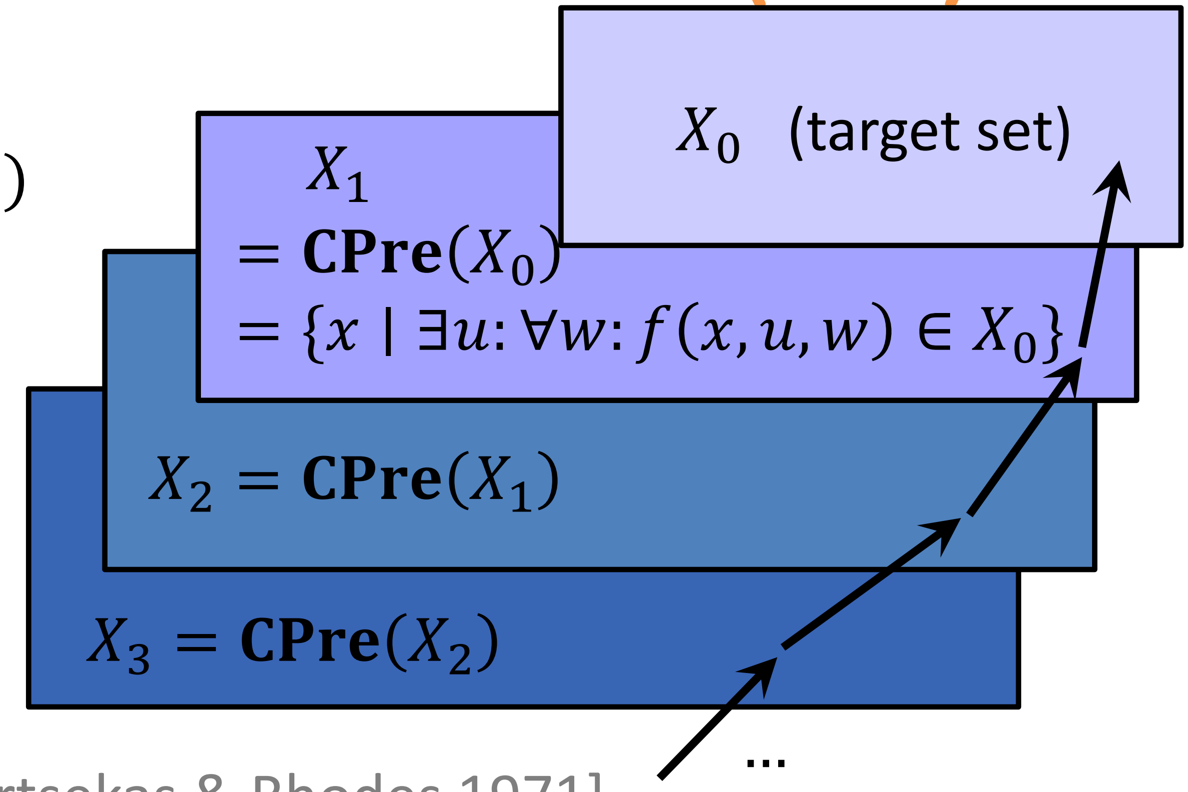
$$w_t \in W$$



Backward Reachable Sets (BRS)

Control system:

$$x_{t+1} = f(x_t, u_t, w_t)$$
$$u_t \in U$$
$$w_t \in W$$



- Specification:
  - Reachability [Bertsekas & Rhodes 1971]
  - Safety [Bertsekas 1972]
  - Temporal logic spec [Chen et al. 2018]

Building block: BRS computation

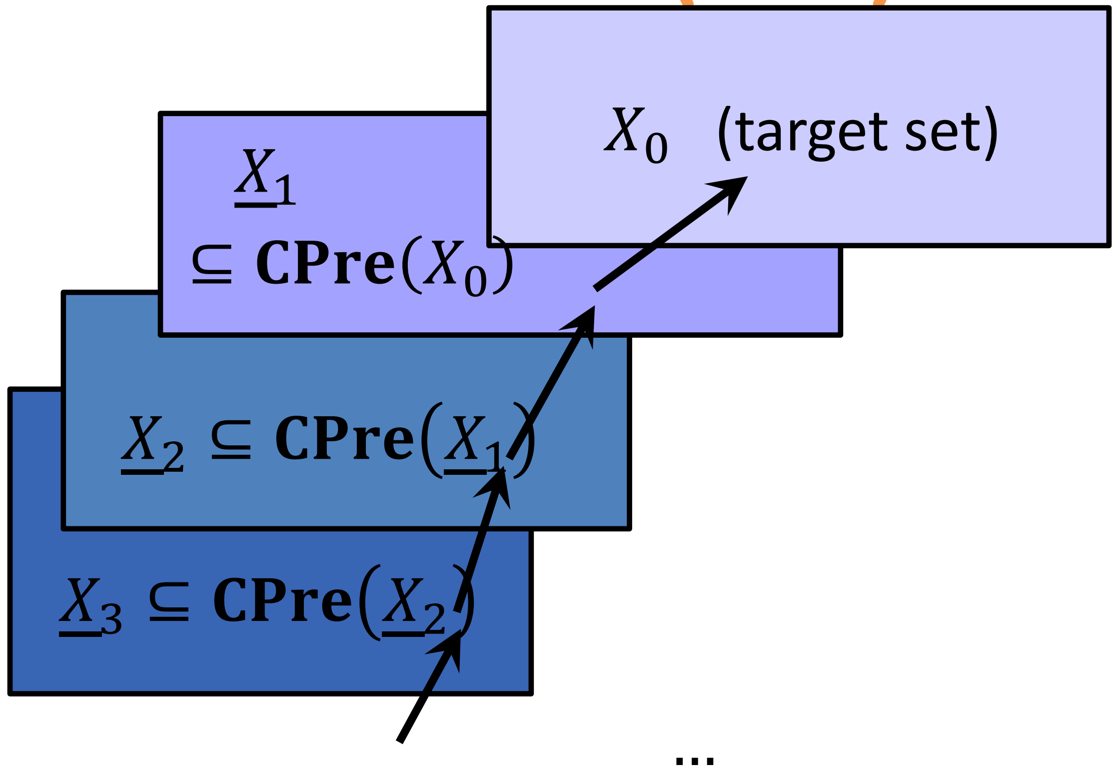
Backward Reachable Sets (BRS)

Control system:

$$x_{t+1} = f(x_t, u_t, w_t)$$

$$u_t \in U$$

$$w_t \in W$$



- Specification:
  - Reachability [Bertsekas & Rhodes 1971]
  - Safety [Bertsekas 1972]
  - Temporal logic spec [Chen et al. 2018]

Inner-approximations can be used for correct-by-construction control

For linear systems

Closed-loop system:

$$x_{t+1} = Ax_t + Bw_t$$
$$w_t \in W$$

$$\mathbf{Post}(X) = \{\, Ax + Bw \mid x \in X, w \in W \,\}$$
$$= AX \oplus BW$$

Control system:

$$x_{t+1} = Ax_t + Bu_t + Ew_t$$
$$u_t \in U$$
$$w_t \in W$$

$$\mathbf{CPre}(X) = \{x \mid \exists u : \forall w : Ax + Bu + Ew \in X\}$$
$$= \mathbf{Proj}_x \{(x, u) \mid Ax + Bu \oplus EW \subseteq X\}$$

If A is invertible:

$$= A^{-1}(X \ominus EW \oplus -BU)$$

What is needed for reachability?

- Set representations (and their complexity)
- Operations on the sets:
  - Affine transformation
  - Projection
  - Intersection
  - Minkowski sum
  - Emptiness check
  - Membership check

Some convex set representations

hyperplane: $P_{hp} = \{x \mid a^T x = b\} \subseteq \mathbb{R}^n$, where $a \in \mathbb{R}^n$, $a \neq 0$, and $b \in \mathbb{R}$

halfspace: $P_{hs} = \{x \mid a^T x \leq b\} \subseteq \mathbb{R}^n$, where $a \in \mathbb{R}^n$, $a \neq 0$, and $b \in \mathbb{R}$

polyhedron: $P = \{x \mid a_j^T x \leq b_j,\ j = 1, \ldots, m,\ c_i^T x = d_i,\ i = 1, \ldots, p\}$, alternatively, in matrix form $P = \{x \mid Ax \leq b,\ Cx = d\}$, or, equivalently $P = \{x \mid \bar{A}x \leq \bar{b}\}$,

Some convex set representations

hyperplane: $P_{hp} = \{x \mid a^T x = b\} \subseteq \mathbb{R}^n$, where $a \in \mathbb{R}^n$, $a \neq 0$, and $b \in \mathbb{R}$

halfspace: $P_{hs} = \{x \mid a^T x \leq b\} \subseteq \mathbb{R}^n$, where $a \in \mathbb{R}^n$, $a \neq 0$, and $b \in \mathbb{R}$

polyhedron: $P = \{x \mid a_j^T x \leq b_j,\ j = 1, \ldots, m,\ c_i^T x = d_i,\ i = 1, \ldots, p\}$, alternatively, in matrix form $P = \{x \mid Ax \leq b,\ Cx = d\}$, or, equivalently $P = \{x \mid \bar{A}x \leq \bar{b}\}$,

zonotope: $Z = \{x \in \mathbb{R}^n \mid x = c + \sum_{i=1}^p \alpha_i g_i,\ -1 \leq \alpha_i \leq 1\}$, where $c, g_1, \ldots, g_p \in \mathbb{R}^n$. The point c is called the *center* of Z; $g_1, \ldots, g_p$ are called the generators of Z. We denote a zonotope as $Z = (c, < g_1, \ldots, g_p >)$.

hyperbox: $H = \{x \in \mathbb{R}^n \mid x_i \in [l_i, u_i],\ i = 1, \ldots, n\}$, where $l_1, \ldots, l_n$ and $u_1, \ldots, u_n$ are real numbers corresponding to lower and upper limits for each coordinate. Hyperboxes are usually denoted as cross-products of intervals, i.e. $H = [l_1, u_1] \times \ldots \times [l_n, u_n]$.

Operations on convex sets

	Intersection $C = C^{(1)} \cap C^{(2)}$	Minkowski sum $C = C^{(1)} \oplus C^{(2)}$	Linear (affine) transformation
Hyperbox $[l_1,u_1] \times \ldots \times [l_n,u_n]$			Not a box!
Zonotope $(c, <g_1, \ldots, g_p>)$	Not a zonotope!	$(c^{(1)} + c^{(2)}, < g_1^{(1)}, .., g_{p_1}^{(1)}, g_1^{(2)}, .., g_{p_2}^{(2)} >)$	$(Lc, <Lg_1, \ldots, Lg_p>)$
Polytope V-rep $\mathrm{Conv}(\{v_1,\ldots,v_k\})$			
Polytope H-rep $Ax \leq b$			

Operations on convex sets

	Intersection $C = C^{(1)} \cap C^{(2)}$	Minkowski sum $C = C^{(1)} \oplus C^{(2)}$	Linear (affine) transformation
Hyperbox $[l_1,u_1] \times \ldots \times [l_n,u_n]$	simple min-max: $l_i = \max(l_i^{(1)}, l_i^{(2)})$ $u_i = \min(u_i^{(1)}, u_i^{(2)})$	simple algebra: $l_i = l_i^{(1)} + l_i^{(2)}$ $u_i = u_i^{(1)} + u_i^{(2)}$	Not a box!
Zonotope $(c, <g_1, \ldots, g_p>)$	Not a zonotope!	$(c^{(1)} + c^{(2)}, < g_1^{(1)}, .., g_{p_1}^{(1)}, g_1^{(2)}, .., g_{p_2}^{(2)} >)$	$(Lc, <Lg_1, \ldots, Lg_p>)$
Polytope V-rep $\mathrm{Conv}(\{v_1,\ldots,v_k\})$			
Polytope H-rep $Ax \leq b$			

Operations on convex sets

	Intersection $C = C^{(1)} \cap C^{(2)}$	Minkowski sum $C = C^{(1)} \oplus C^{(2)}$	Linear (affine) transformation
Hyperbox $[l_1,u_1] \times \ldots \times [l_n,u_n]$	simple min-max: $l_i = \max(l_i^{(1)}, l_i^{(2)})$ $u_i = \min(u_i^{(1)}, u_i^{(2)})$	simple algebra: $l_i = l_i^{(1)} + l_i^{(2)}$ $u_i = u_i^{(1)} + u_i^{(2)}$	Not a box!
Zonotope $(c, <g_1, \ldots, g_p>)$	Not a zonotope!	$(c^{(1)} + c^{(2)}, < g_1^{(1)}, .., g_{p_1}^{(1)}, g_1^{(2)}, .., g_{p_2}^{(2)} >)$	$(Lc, <Lg_1, \ldots, Lg_p>)$
Polytope V-rep $\mathrm{Conv}(\{v_1,\ldots,v_k\})$		$\mathrm{Conv}(\{v_i^{(1)} + v_j^{(2)}\}_{i,j})$ ★	$\mathrm{Conv}(\{Lv_1,\ldots,Lv_k\})$
Polytope H-rep $Ax \leq b$	concatenation: ★ $\begin{bmatrix} A^{(1)} \\ A^{(2)} \end{bmatrix} x \leq \begin{bmatrix} b^{(1)} \\ b^{(2)} \end{bmatrix}$	Outer-approx. when $C^{(2)}$ is ∞-norm ball: $Ax \leq b + \begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix} \lvert\lvert A \rvert\rvert_\infty \epsilon$	

Radius of the ball

★ Representations might be redundant, reductions are possible.

In theory do not scale well with dimension n, in practice it is OK.

Operations on convex sets

	Membership check Is point x in C?	Emptiness check Is C the empty set?
Hyperbox $[l_1,u_1] \times \ldots \times [l_n,u_n]$	simple comparisons: $l_i \le x_i \le u_i$ for all i?	Non-empty iff $l_i \le u_i$ for all i
Zonotope $(c, \langle g_1, \ldots, g_p \rangle)$	linear program (LP) poly-time	Can't represent empty sets
Polytope V-rep $\mathrm{Conv}(\{v_1,\ldots,v_k\})$	linear program (LP) poly-time	Can't represent empty sets (trivial empty vertex set)
Polytope H-rep $Ax \le b$	simple algebra: $Ax \le b$?	linear program (LP) poly-time

Additional notes

- Other important operations:
  - Containment check (see Sadraddini & Tedrake'19), Minkowski difference, complexity reduction
- Approximate (inner or outer) set computations when exact operations are hard
- Many other set representations:
  - Constrained zonotopes, hybrid zonotopes, polynomial zonotopes, AH-polytopes, star sets, ellipsoids, …
- Software packages for manipulating sets:
  - Matlab MPT3, Python polytope, Julia JuMP
- Reachability software:
  - FRS: CORA, JuliaReach, SpaceX, dReach, etc. (see https://ieeecss.org/tc/hybrid-systems/tools)
  - BRS: HJB, MPT3

Backward Reachable Set Computation

- Methods/tools (far from being complete):
  - HJB [Mitchell et al. 2007], interval analysis [Li & Liu 2017], polynomial optimization [Lasserre 2015], linear optimization [Blanchini & Miani 2008], etc.

- Challenge: **scalability** (even for linear systems):

$$
\begin{aligned}
x_{t+1} &= Ax_t + Bu_t + w_t \\
u_t \in U &= \{u \mid H_u u \le h_u\} \\
w_t \in W &= \{w \mid H_w w \le h_w\} \\
X_0 &= \{x \mid H_x x \le h_x\}
\end{aligned}
$$

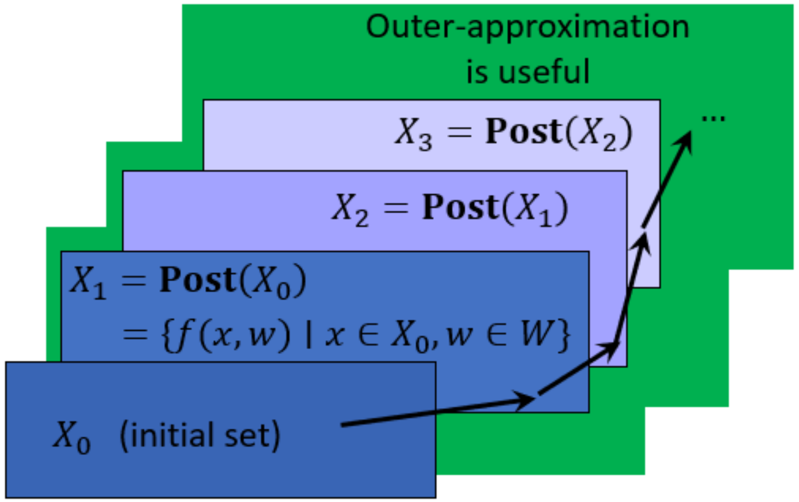
Polytopes in half-space representations (H-Reps)

Projection is difficult for H-Reps

$$
X_{k+1} = \boxed{\mathbf{Proj}_x} \{(x,u) \mid Ax + Bu + EW \subseteq X_k\}
$$

Can we use zonotopes to represent X_k?

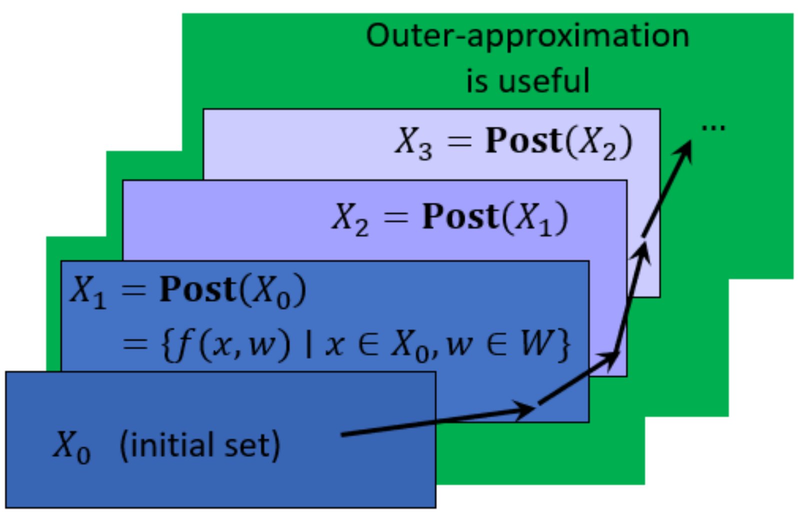
- Zonotope: $\{ G\theta + c \mid \theta \in [-1,1]^N \} = \boxed{\langle G, c \rangle}$ Generator representation (G-Rep)
- Advantages:
  - Affine transformation (projection), Minkowski sum are easy
  - Order reduction (for outer-approximations)



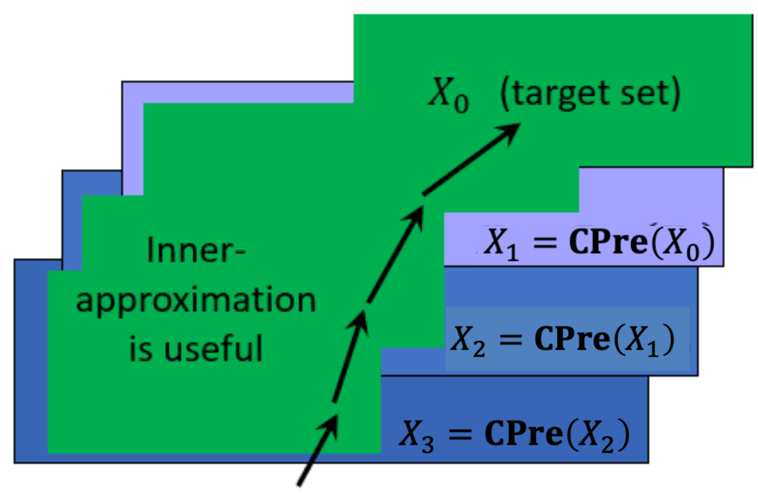
$$X_{k+1} = AX_k \oplus BW$$

Can we use zonotopes to represent X_k?

- Zonotope: $\{\, G\theta + c \mid \theta \in [-1,1]^N \,\} = \boxed{\langle G, c \rangle}$ Generator representation (G-Rep)
- Advantages:
  - Affine transformation (projection), Minkowski sum are easy
  - Order reduction (for outer-approximations)
- For backward reachability, there lack efficient algos for:
  - Minkowski difference
  - Order reduction (for inner-approximations)



$$X_{k+1} = AX_k \oplus BW$$



$$X_{k+1} = A^{-1}(X_k \ominus EW \oplus -BU)$$

Main Results

- Efficient inner/outer-approximations of the Minkowski difference when the minuend is a zonotope – LCSS’22

- A zonotope order reduction technique (for inner-approximation) – LCSS’22

- A scalable BRS under-approximation algorithm – LCSS’22

- Extensions to constrained zonotopes and nonlinear systems (with some results on complexity and approximability) – EMSOFT’22

Under-Approximating $X_k \ominus EW$

- $X_k = \langle G, c \rangle, \;\; W = \mathrm{cvxh}(V_W)$

- Step I: over-approximate EW by $\langle G\ \mathrm{diag}(\alpha), c' \rangle$

  α, c' can be found by solving a linear program

  #variables $= \mathcal{O}(MN + n)$

  #constraints $= \mathcal{O}(MN + Mn)$

n: dimension of X_k
M: #vertices in V_W
N: #columns in G

- Step II: $X_k \ominus EW \subseteq \langle G, c \rangle \ominus \langle G\ \mathrm{diag}(\alpha), c' \rangle$

  $= \langle G\ \mathrm{diag}(1 - \alpha), c - c' \rangle$

Step II is just G-Rep manipulation

Under-Approximating $X_k \ominus EW$

- $X_k = \langle G, c \rangle, \quad W = \mathrm{cvxh}(V_W)$

- Step I: over-approximate EW by $\langle G\ \mathrm{diag}(\alpha), c' \rangle$
  α, c' can be found by solving a linear program

$$
\begin{aligned}
\min_{\theta,\,\alpha,\,c} \quad & \textstyle\sum_{i=1}^{N} b_i \alpha_i \\
\text{s.t.} \quad & \forall w_j \in V : c + \textstyle\sum_{i=1}^{N} \theta_{ij} g_i = E w_j \\
& |\theta_{ij}| \le \alpha_i \le 1, \;\; i = 0, 1, \ldots N
\end{aligned}
$$

n: dimension of X_k
M: #vertices in V_W
N: #columns in G

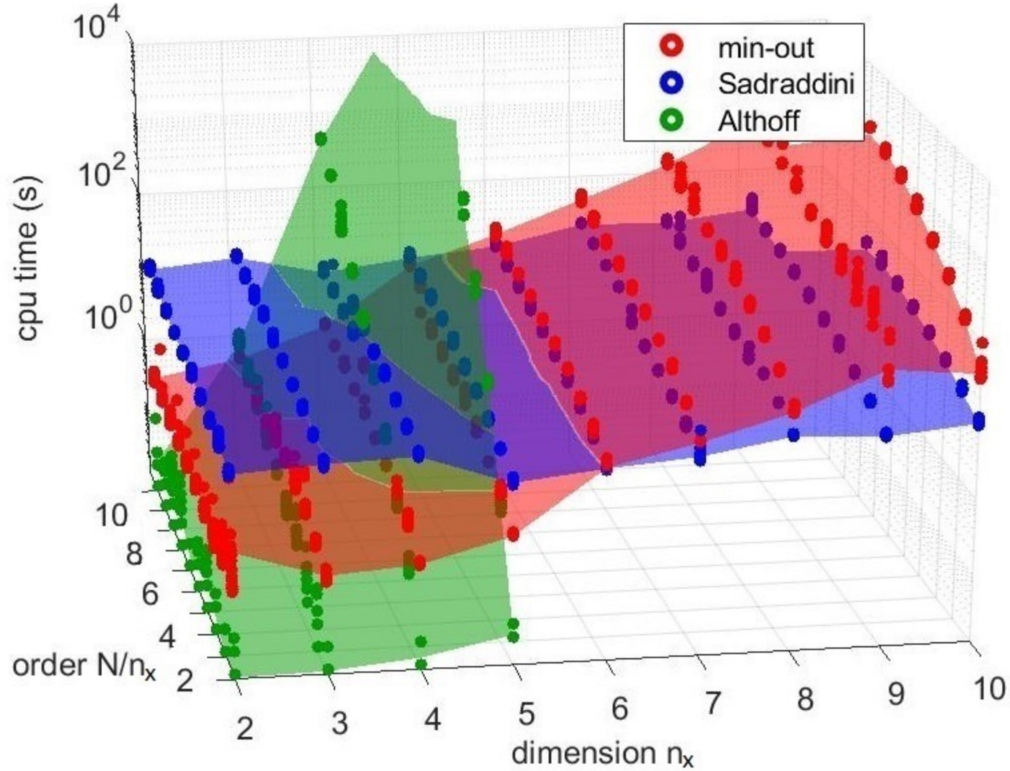
- Step II: $X_k \ominus EW \subseteq \langle G, c \rangle \ominus \langle G\ \mathrm{diag}(\alpha), c' \rangle$

$$= \langle G\ \mathrm{diag}(1-\alpha), c - c' \rangle$$

Step II is just G-Rep manipulation

Evaluation with Random Instances

- Comparison
  - H-Rep manipulation [Althoff 2015]
  - Zonotope containment [Sadraddini & Tedrake 2019], [Raghuraman & Koeln 2022] if W has a G-Rep



	mean	std.	min	max
$(V/V)^{1/n}$	1.0017	0.0577	0.9900	1.3856
$(V/V)^{1/n}$	0.9678	0.1891	0.8372	1.7498

Order Reduction (for Inner-Approximation)

- Idea:

$$\langle [g_1, g_2, \dots, g_N], c \rangle \xrightarrow[]{\text{replace } [g_i, g_j] \text{ by } g_i + g_j \text{ or } g_i - g_j} \text{\#generators reduces by one}$$

Order Reduction

- Idea:

$$\langle [g_1, g_2, \dots, g_N], c \rangle \xrightarrow[]{\text{replace } [g_i, g_j] \text{ by } g_i + g_j \text{ or } g_i - g_j} \begin{array}{c}\text{\#generators} \\ \text{reduces by one}\end{array}$$

- Two questions:
  - Which $[g_i, g_j]$ to "combine"?
  - Replace $[g_i, g_j]$ with $g_i + g_j$ or $g_i - g_j$?

Order Reduction

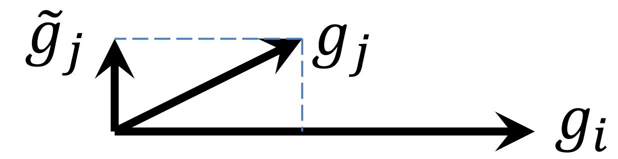
- Idea:

$$\langle [g_1, g_2, \dots, g_N], c \rangle \xrightarrow{\text{replace } [g_i, g_j] \text{ by } g_i + g_j \text{ or } g_i - g_j}$$

#generators reduces by one

- Two questions:
  - Which $[g_i, g_j]$ to "combine"?

Pick **small** or **closely-aligned** generators



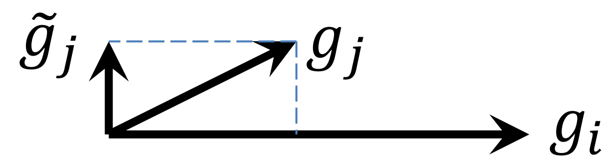
  - Replace $[g_i, g_j]$ with $g_i + g_j$ or $g_i - g_j$?

Order Reduction

- Idea:

$$\langle [g_1, g_2, \dots, g_N], c\rangle \xrightarrow{\substack{\text{replace } [g_i, g_j] \text{ by} \\ g_i + g_j \text{ or } g_i - g_j}} \begin{array}{c}\#\text{generators} \\ \text{reduces by one}\end{array}$$

- Two questions:
  - Which $[g_i, g_j]$ to "combine"?
    - $(i, j) = \operatorname{argmin} \|g_i\|_2 \|\tilde{g}_j\|_2$
    - $(i, j) = \operatorname{argmin} \|g_i\|_2 \|g_j\|_2 \frac{\sigma_{\min}}{\sigma_{\max}}$



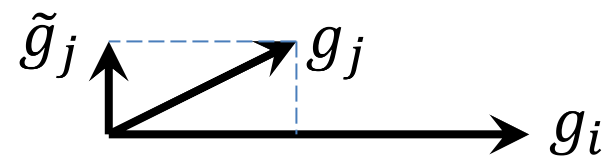
  - Replace $[g_i, g_j]$ with $g_i + g_j$ or $g_i - g_j$?

Order Reduction

- Idea:

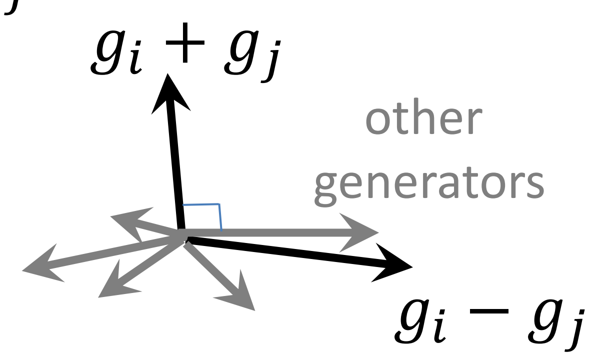
$$\langle [g_1, g_2, \dots, g_N], c \rangle \xrightarrow{\text{replace } [g_i, g_j] \text{ by } g_i + g_j \text{ or } g_i - g_j} \begin{array}{c}\#\text{generators} \\ \text{reduces by one}\end{array}$$

- Two questions:
  - Which $[g_i, g_j]$ to "combine"?
    - $(i, j) = \operatorname{argmin} \|g_i\|_2 \|\tilde{g}_j\|_2$
    - $(i, j) = \operatorname{argmin} \|g_i\|_2 \|g_j\|_2 \frac{\sigma_{\min}}{\sigma_{\max}}$



  - Replace $[g_i, g_j]$ with $g_i + g_j$ or $g_i - g_j$?

Use the one that is **larger and "more perpendicular"** to the other generators

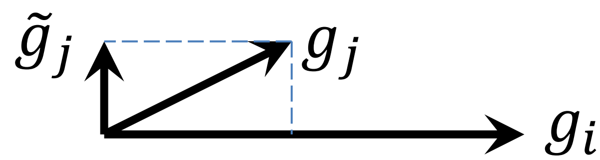


Order Reduction

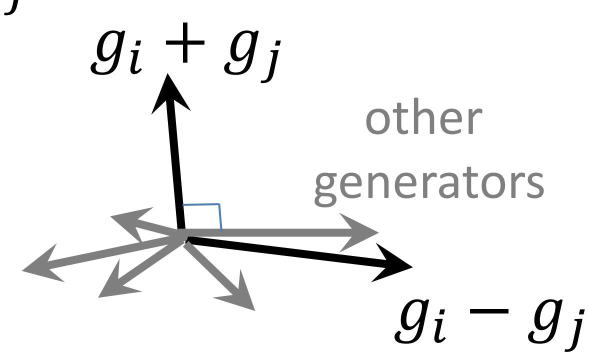
- Idea:

$$\langle [g_1, g_2, \dots, g_N], c \rangle \xrightarrow{\substack{\text{replace } [g_i, g_j] \text{ by} \\ g_i + g_j \text{ or } g_i - g_j}} \begin{matrix}\#\text{generators} \\ \text{reduces by one}\end{matrix}$$

- Two questions:
  - Which $[g_i, g_j]$ to "combine"?
    - $(i, j) = \operatorname{argmin} \|g_i\|_2 \|\tilde{g}_j\|_2$
    - $(i, j) = \operatorname{argmin} \|g_i\|_2 \|g_j\|_2 \dfrac{\sigma_{\min}}{\sigma_{\max}}$



  - Replace $[g_i, g_j]$ with $g_i + g_j$ or $g_i - g_j$?
    - Replace with $g_i + g_j$ if $\|G^\dagger (g_i + g_j)\|_2 \geq \|G^\dagger (g_i - g_j)\|_2$ and $g_i - g_j$ otherwise

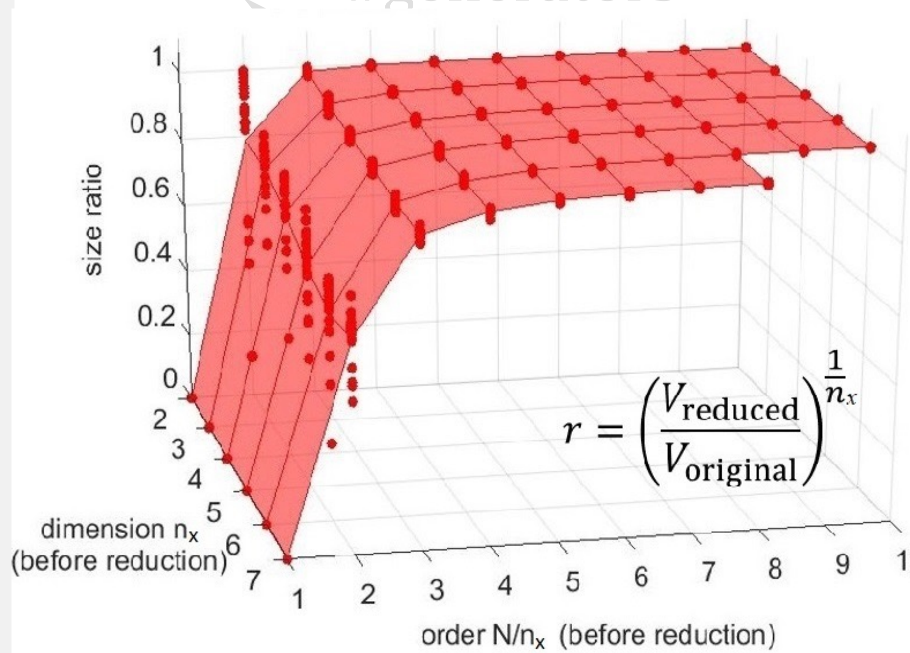


Order Reduction

- Idea:

replace $[g_i, g_j]$ by $g_i + g_j$ or $g_i - g_j$

#generators



- Replace with $g_i + g_j$ if $\|G^{\dagger}(g_i + g_j)\|_2 \geq \|G^{\dagger}(g_i - g_j)\|_2$ and $g_i - g_j$ otherwise

Example: Aircraft Position Control

- Longitudinal: x in 6D, u in 2D,
- Lateral: x in 6D, u in 2D

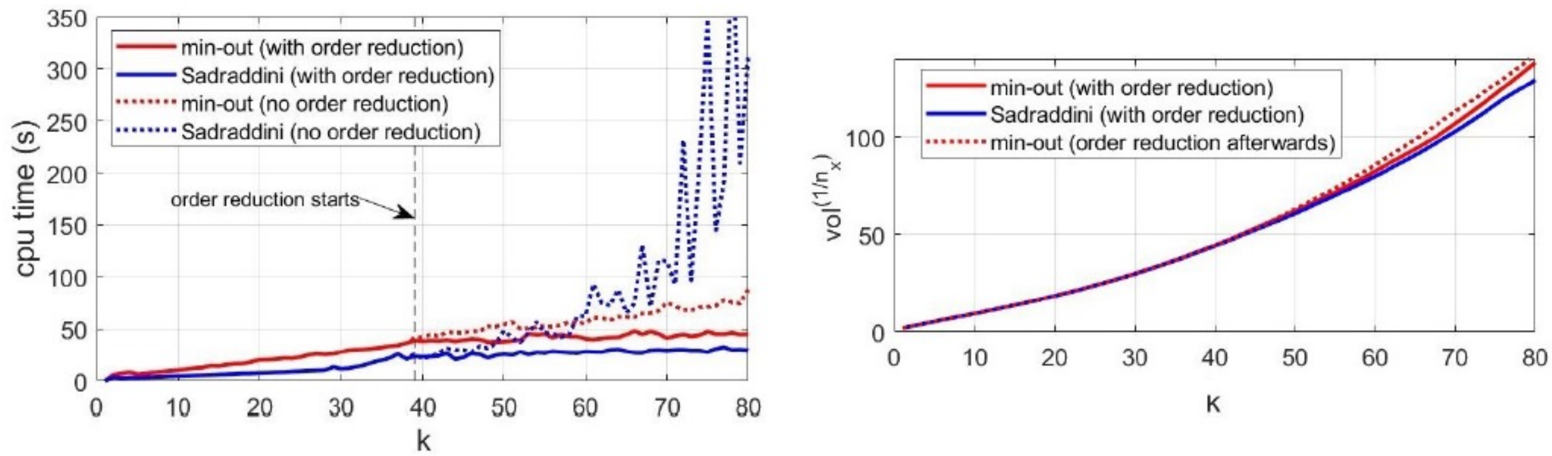
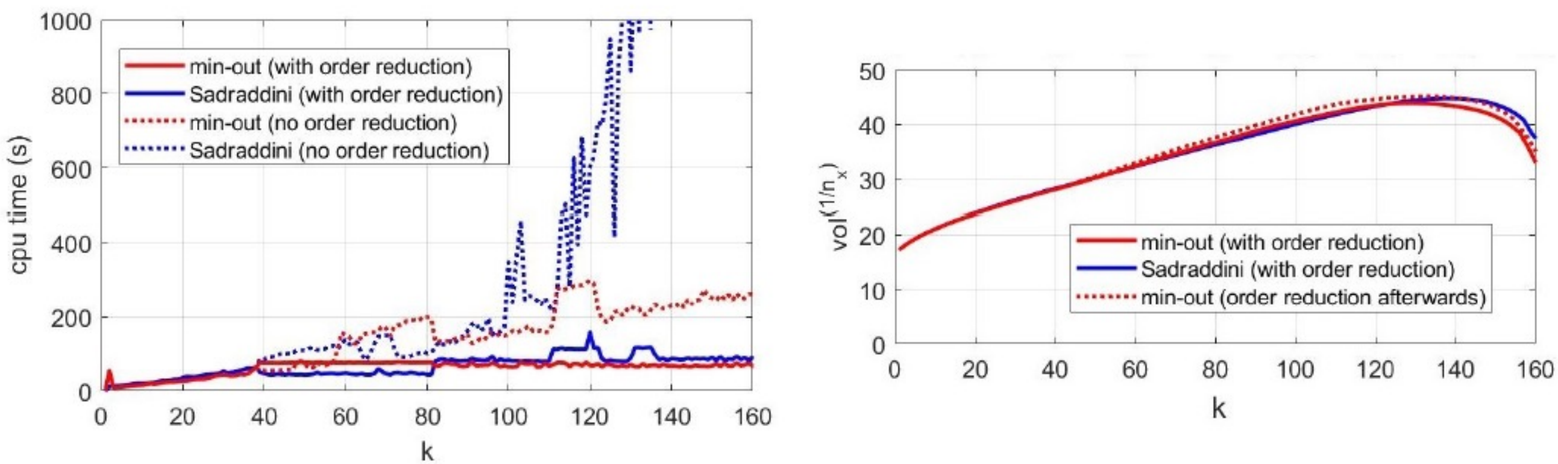


Fig. 3: Backward reachable set computation for lateral dynamics. Left: computation time. Right: set volume.



Limitations & extensions

- Zonotopes are not as rich as polytopes in terms of expressiveness
  - How about constrained zonotopes?
- So far, applicable to linear systems
  - How about nonlinear systems?

Problem Formulation

- **System:**

$$\boldsymbol{x}_{t+1} = \boldsymbol{f}(\boldsymbol{x}_t, \boldsymbol{u}_t) + \boldsymbol{w}_t$$
$$\boldsymbol{u}_t \in \mathcal{U}, \quad \boldsymbol{w}_t \in \mathcal{W}$$

Assumptions on sets

Target set, control set

$\mathcal{X}_0 = \langle \boldsymbol{G}_0, \boldsymbol{c}_0, \boldsymbol{A}_0, \boldsymbol{b}_0 \rangle$

$\mathcal{U} = \langle \boldsymbol{G}_u, \boldsymbol{c}_u, \boldsymbol{A}_u, \boldsymbol{b}_u \rangle$

are constrained zonotopes

Disturbance set

$\mathcal{W} = \langle \boldsymbol{G}_w, \boldsymbol{c}_w \rangle$ **is a zonotope**

Safe set

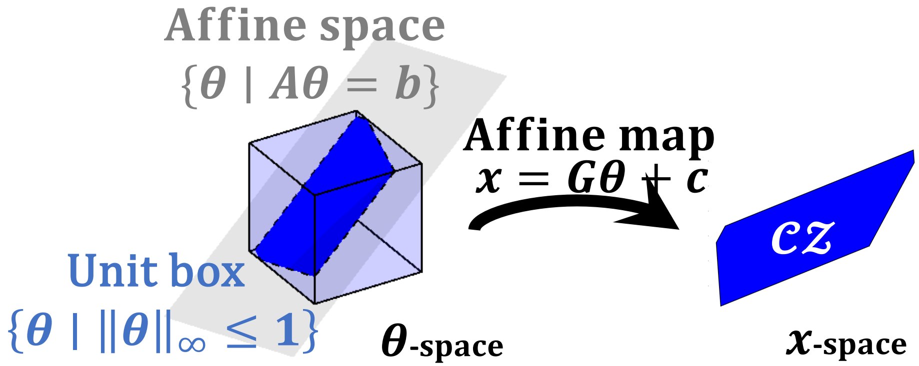
$\mathcal{X}_{\text{safe}} = \{ \boldsymbol{x} \mid \boldsymbol{H}\boldsymbol{x} \le \boldsymbol{a} \}$ **is a polyhedron**

Constrained zonotope

If no A, b the set is just a zonotope

$\mathcal{CZ} = \{ \boldsymbol{G}\boldsymbol{\theta} + \boldsymbol{c} \mid \|\boldsymbol{\theta}\|_\infty \le 1, \cancel{A\theta} = b \}$

CG-Rep of $\mathcal{CZ}$: $\langle \boldsymbol{G}, \boldsymbol{c}, A, b \rangle$



BRS Computation Using Constrained Zonotopes

- **Linear system:** $$\boldsymbol{x}_{t+1} = \boldsymbol{A}\boldsymbol{x}_t + \boldsymbol{B}\boldsymbol{u}_t + \boldsymbol{w}_t$$ $$\boldsymbol{u}_t \in \mathcal{U}, \quad \boldsymbol{w}_t \in \mathcal{W}$$
- **BRS computation:** $$\mathcal{X}_{k+1} = Pre(\mathcal{X}_k) = \boldsymbol{A}^{-1}(\mathcal{X}_k \ominus \mathcal{W} \oplus -\boldsymbol{B}\mathcal{U}) \cap \mathcal{X}_{\text{safe}}$$

For nonlinear systems, replace $\mathcal{W}$ by $\mathcal{W} \oplus \mathcal{L}$, where $\mathcal{L}$ contains all linearization error [Althoff 2008]

Assumptions on sets

Target set, BRS, control set

$\mathcal{X}_k = \langle \boldsymbol{G}_k, \boldsymbol{c}_k, \boldsymbol{A}_k, \boldsymbol{b}_k \rangle$

$\mathcal{U} = \langle \boldsymbol{G}_u, \boldsymbol{c}_u, \boldsymbol{A}_u, \boldsymbol{b}_u \rangle$

are constrained zonotopes

Disturbance set

$\mathcal{W} = \langle \boldsymbol{G}_w, \boldsymbol{c}_w \rangle$ **is a zonotope**

Safe set

$\mathcal{X}_{\text{safe}} = \{\boldsymbol{x} \mid \boldsymbol{H}\boldsymbol{x} \leq \boldsymbol{a}\}$ **is a polyhedron**

Set operations involved

Affine map

$A\mathcal{X} = \{Ax \mid x \in \mathcal{X}\}$

$\mathcal{Y}\}$

Easy to compute for **constrained zonotopes,** via **CG-Rep manipulation**

Minkowski difference

$\mathcal{X} \ominus \mathcal{Y} = \{x \mid \{x\} \oplus \mathcal{Y} \subseteq \mathcal{X}\}$?

Main Result: Under-Approximate $\mathcal{CZ} \ominus \mathcal{Z}$ Efficiently

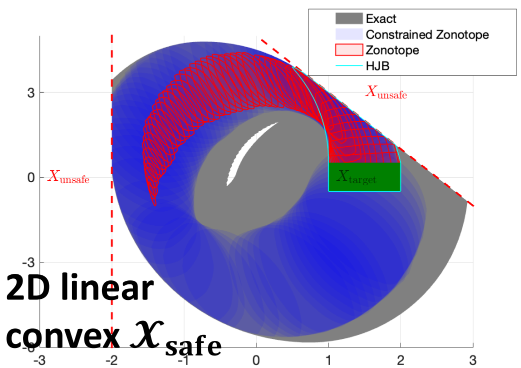
Theorem 1.
Given $\mathcal{CZ} = \langle \boldsymbol{G}, \boldsymbol{c}, \boldsymbol{A}, \boldsymbol{b} \rangle$ and $\mathcal{Z} = \langle \boldsymbol{G}', \boldsymbol{c}' \rangle$, no algorithm finds a polynomial-size CG-Rep of $\mathcal{CZ} \ominus \mathcal{Z}$ in polynomial-time (unless P=NP).

Theorem 2.
We can find a set $\mathcal{CZ}_{\mathrm{d}} \subseteq \mathcal{CZ} \ominus \mathcal{Z}$ by solving a linear program, whose # variables and # constraints are polynomial in the size of $\mathcal{CZ}$'s and $\mathcal{Z}$'s representations.

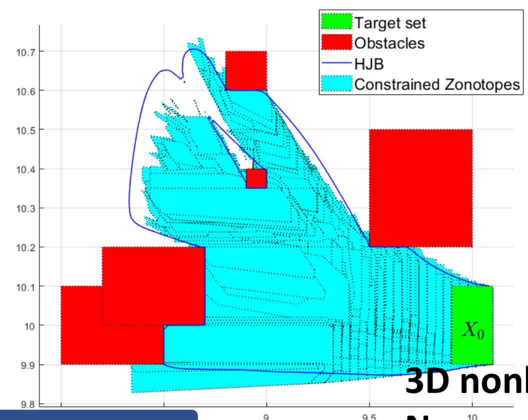
Theorem 3.
CG-Rep not unique
Every constrained zonotope $\mathcal{CZ}$ has a "rich" enough CG-Rep s.t. our under-approximation is exact, i.e., $\mathcal{CZ}_{\mathrm{d}} = \mathcal{CZ} \ominus \mathcal{Z}$.

There is a trade-off between accuracy & efficiency, which can be "tuned" via CG-Rep selection

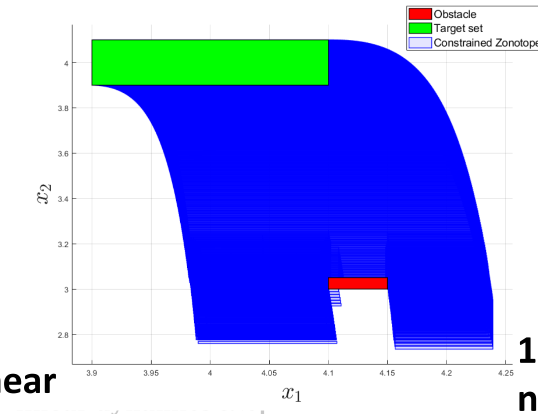
BRS Under-Approximation Algorithms



2D linear convex $\mathcal{X}_{\text{safe}}$



3D nonlinear Nonconvex $\mathcal{X}_{\text{safe}}$



10D nonlinear Nonconvex $\mathcal{X}_{\text{safe}}$

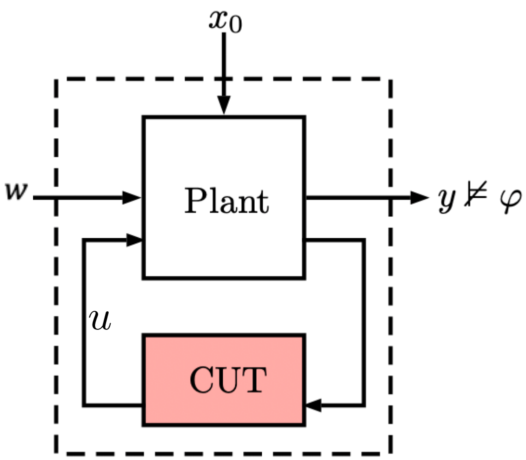
- **Less conservative then zonotope-based approach** [Yang & Ozay 2022]
- **Deal with constraints (convex or nonconvex)**
- **More scalable than HJB** [Bansal et al. 2017]

	n	k	Splitting	Scaling	HJB
Example 2	2	100	N/A	56.1s	45.2s
Example 3	10	10	226.7s	N/A	Memory error
Example 4: Convex constraints	3	—	478.9s ($k = 25$)	2821.3s ($k = 400$)	Memory error
Example 4: Nonconvex constraints	3	20	1564.1s	N/A	4521.6s
Example 5	10	340	951.6s	N/A	Memory error

HJB: hard to scale

How to use backward reachable sets (BRS) in falsification?

Synthesis-guided falsification



w disturbance (external input)
x_0: initial condition
u: control input
y: output
φ: specification

Falsification problem: Given a plant model, a (blackbox) controller and a specification, can we find an initial condition and an external input sequence (disturbance sequence) so that the specification is violated?

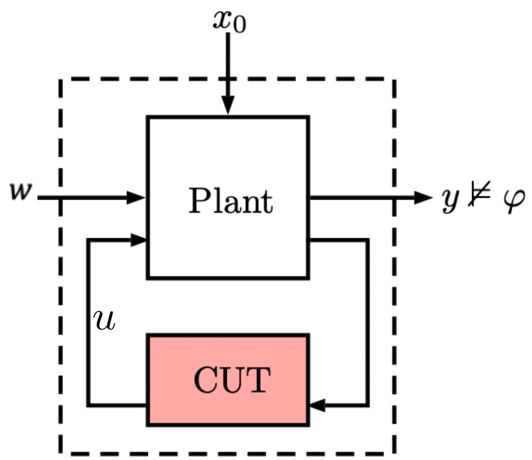
Assumptions:

- "simple enough" plant model
$$x_{t+1} = f(x_t, u_t, w_t)$$
- "simple enough" specification: we will focus on safety (invariance) and reachability specifications

"simple enough": almost anything for which you can compute the validity domain (i.e., winning set) of the synthesis problem

Chou et al. EMSOFT 18

Synthesis-guided falsification

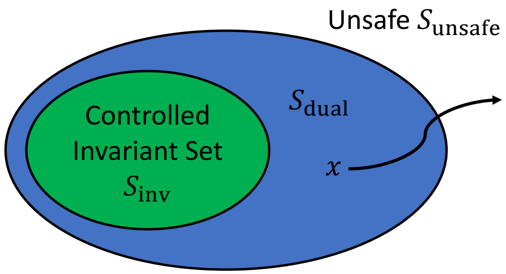


w disturbance (external input)
x_0: initial condition
u: control input
y: output
φ: specification

Plant model:

$$x_{t+1} = f(x_t, u_t, w_t)$$

Ideal result of the synthesis problems:



Approach:

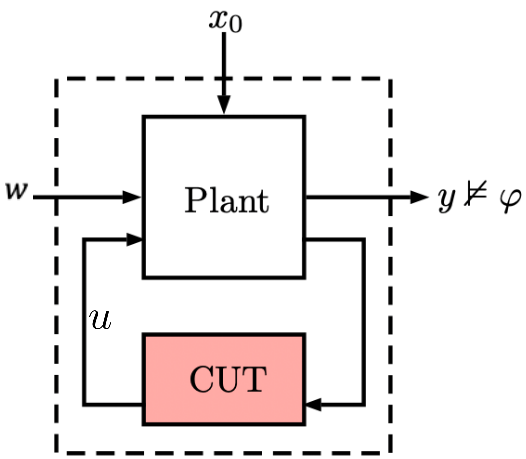
- Ignore the controller, focus on safety-critical part of the spec.
- Given the plant model and safety (invariance) part X_{safe} of the spec, consider the safety and dual reachability synthesis problems:

Invariance in X_{safe}

Reachability to

$X_{unsafe} = \neg X_{safe}$

Synthesis-guided falsification

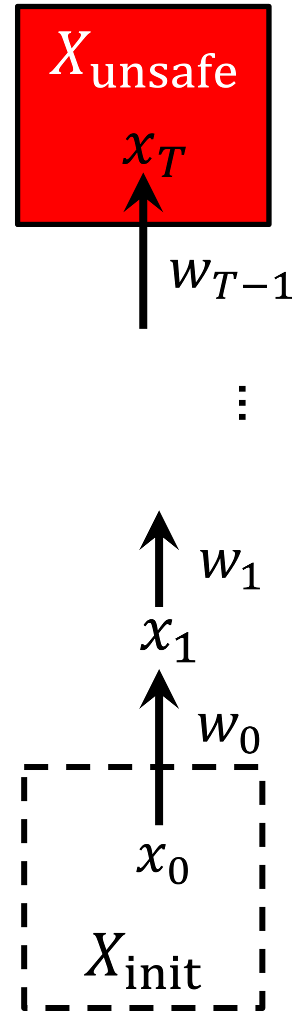


w disturbance (external input)
x_0: initial condition
u: control input
y: output
φ: specification

Invariance in X_{safe}

Reachability to

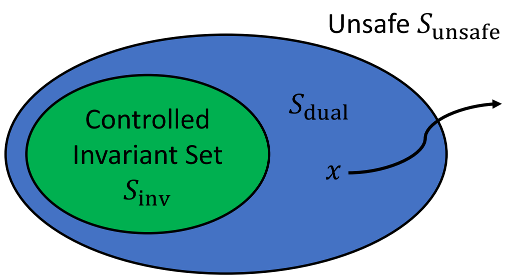
$X_{unsafe} = \neg X_{safe}$



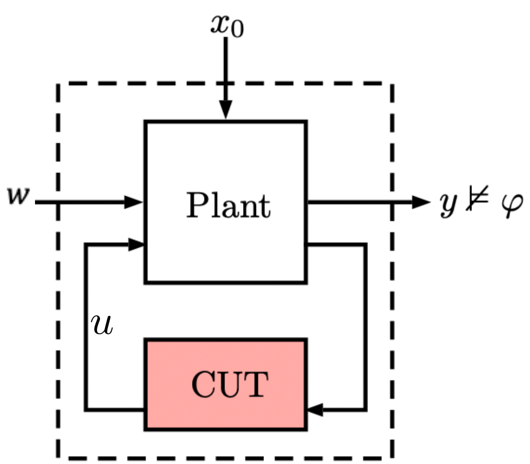
Plant model:

$$x_{t+1} = f(x_t, u_t, w_t)$$

Ideal result of the synthesis problems:



Synthesis-guided falsification

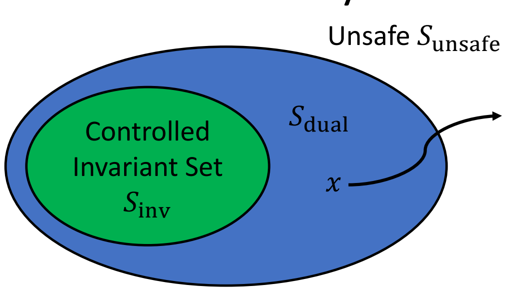


w disturbance (external input)
x_0: initial condition
u: control input
y: output
φ: specification

Plant model:

$$x_{t+1} = f(x_t, u_t, w_t)$$

Ideal result of the synthesis problems:



Approach:

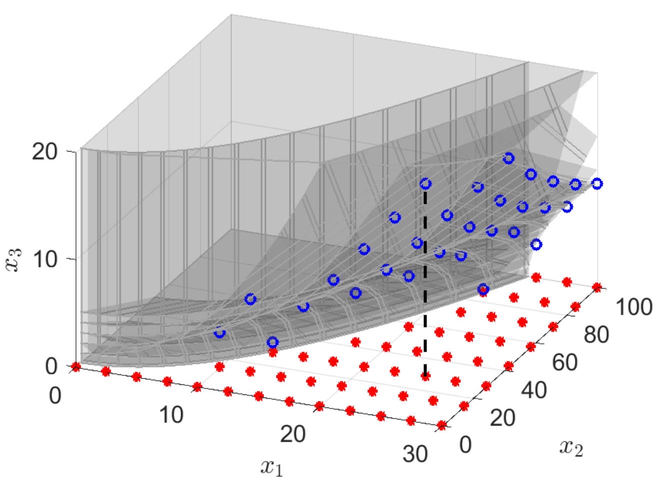
- Ignore the controller, focus on safety-critical part of the spec.
- Given the plant model and safety (invariance) part X_{safe} of the spec, consider the safety and dual reachability synthesis problems.

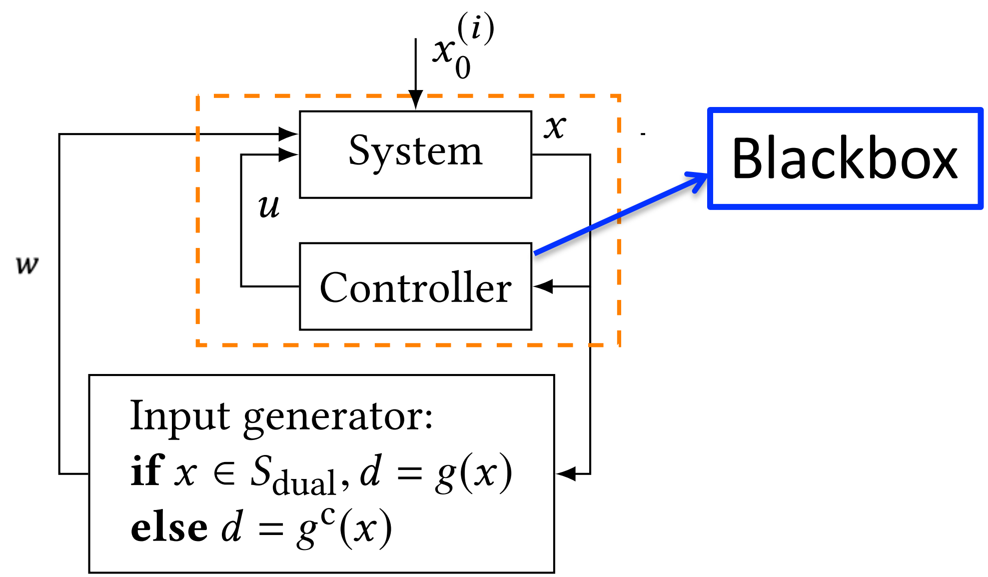
Comments:

- Synthesized X_{inv} is the validity domain W of the best safety controller (i.e., **the maximal** invariant set)
- Synthesized X_{dual} is the validity domain of "best" disturbance policy
- OK to use approximate computations/models
- Can use any synthesis approach: *iterative polytopic computations*, Hamilton Jacobi Bellman, control barrier functions, abstractions

Synthesis-guided falsification

- Hypotheses:
  - **Hard initial conditions**: boundary of the control invariant set (small # of safe control inputs)
  - **Hard external inputs**: solutions of a dual reachability problem (when out of S_{dual}, pick best effort input to get close to S_{dual})



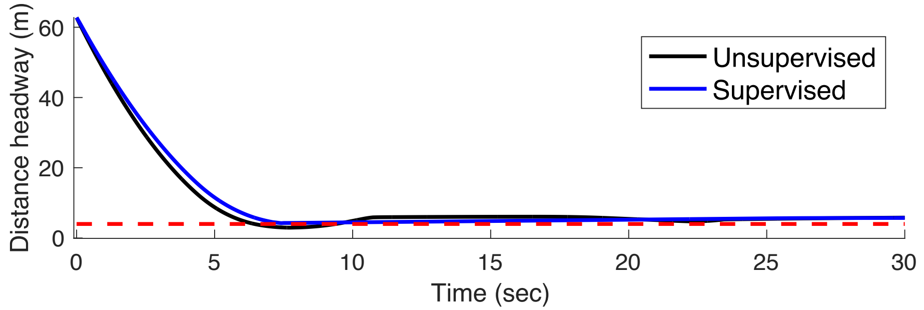


Synthesis-guided falsification

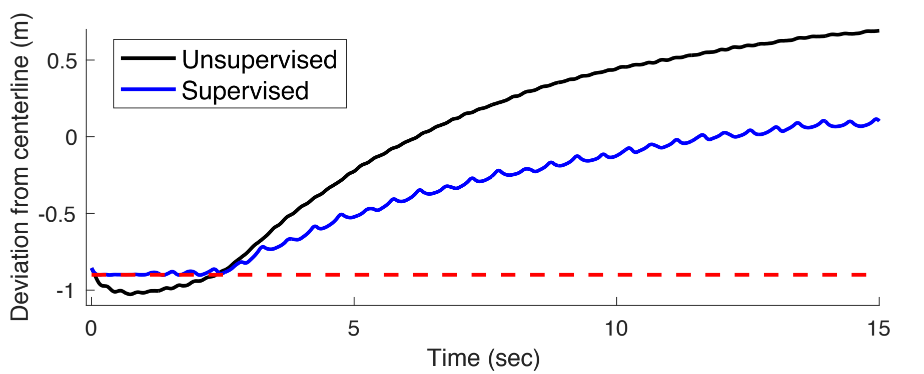
- Some example “bugs” found in the open-source autonomous driving software CommaAI (using its python source code directly!)
- Similar results with Stanford DARPA Grand Challenge code (C++ code)
- Can use the synthesis artifacts for “sandboxing” (supervising) complex controllers
- Our software is available online: integration of several driving software with car dynamics models

Can handle any (learning-based) state-feedback controller (e.g., NNs, RL, etc.)
→ so far no perception modules

Adaptive cruise control



Lane keeping

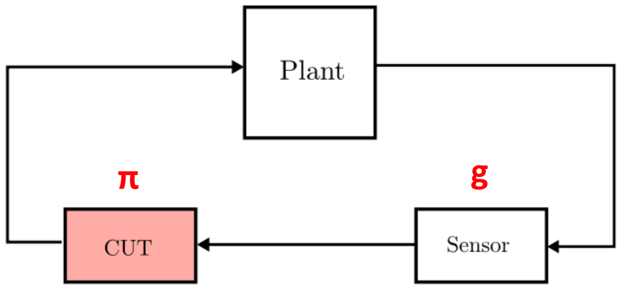


Synthesis-guided falsification with partial information

known & simple
(can be uncertain)

π

g



unknown &
complicated

known & simple
(can be noisy)

an abstraction of a
complex perception
module

Plant model including sensor:

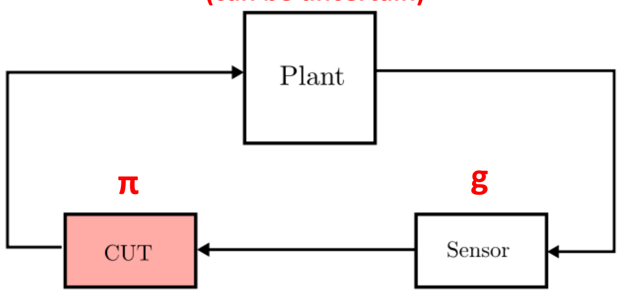
$$x_{t+1} = f(x_t, u_t, w_t)$$

$$y_t = g(x_t, v_t)$$

- Given:
  - A gray-box system
  - An unsafe set X_{unsafe}, and an initial set X_{init}
- Find one adversarial example:
  - A trajectory $x_0, x_1, \dots, x_T$
  - External inputs $w_0, w_1, \dots, w_{T-1}, v_0, v_1, \dots, v_{T-1}$
  - $x_0 \in X_{\text{init}}$ and $x_T \in X_{\text{unsafe}}$

Synthesis-guided falsification with partial information

known & simple
(can be uncertain)



unknown & complicated

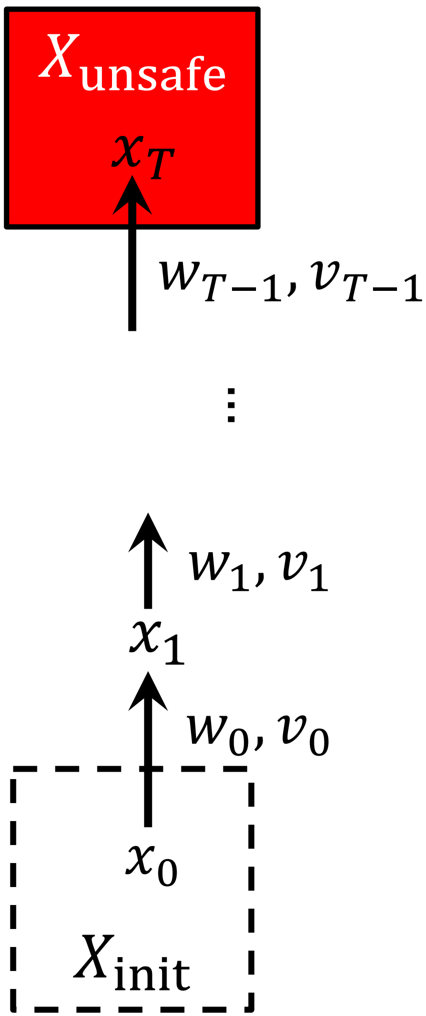
known & simple
(can be noisy)

an abstraction of a complex perception module

Plant model including sensor:

$$x_{t+1} = f(x_t, u_t, w_t)$$

$$y_t = g(x_t, v_t)$$

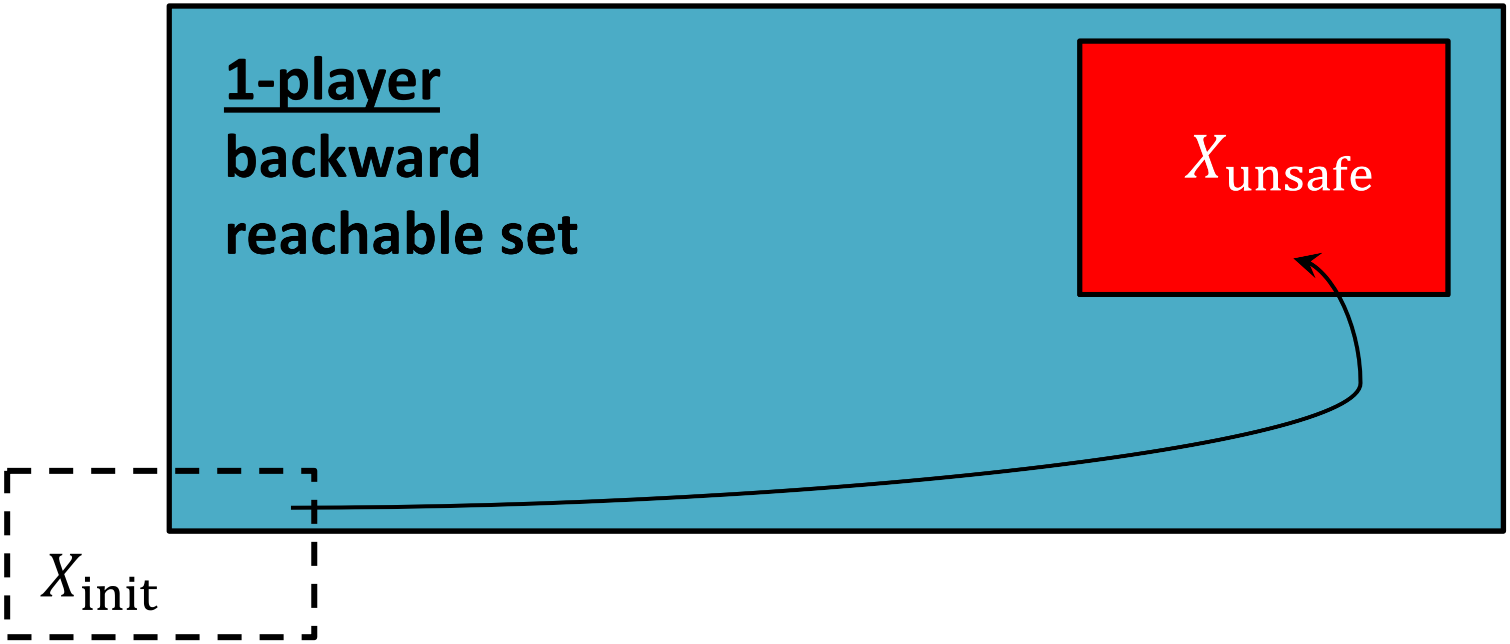


Key Idea

Verification: compute 1-player backward reachable set $= \bigcup X_k$

$$X_0 = X_{\text{unsafe}}$$
$$X_{k+1} = \mathbf{Pre}(X_k) := X_k \cup \{x \mid \exists w, v: \underbrace{f(x, \pi(x, v), w)} \in X_k\}$$

Closed-loop dynamics: complex due to π, even if f itself is simple (precisely why verification is hard)



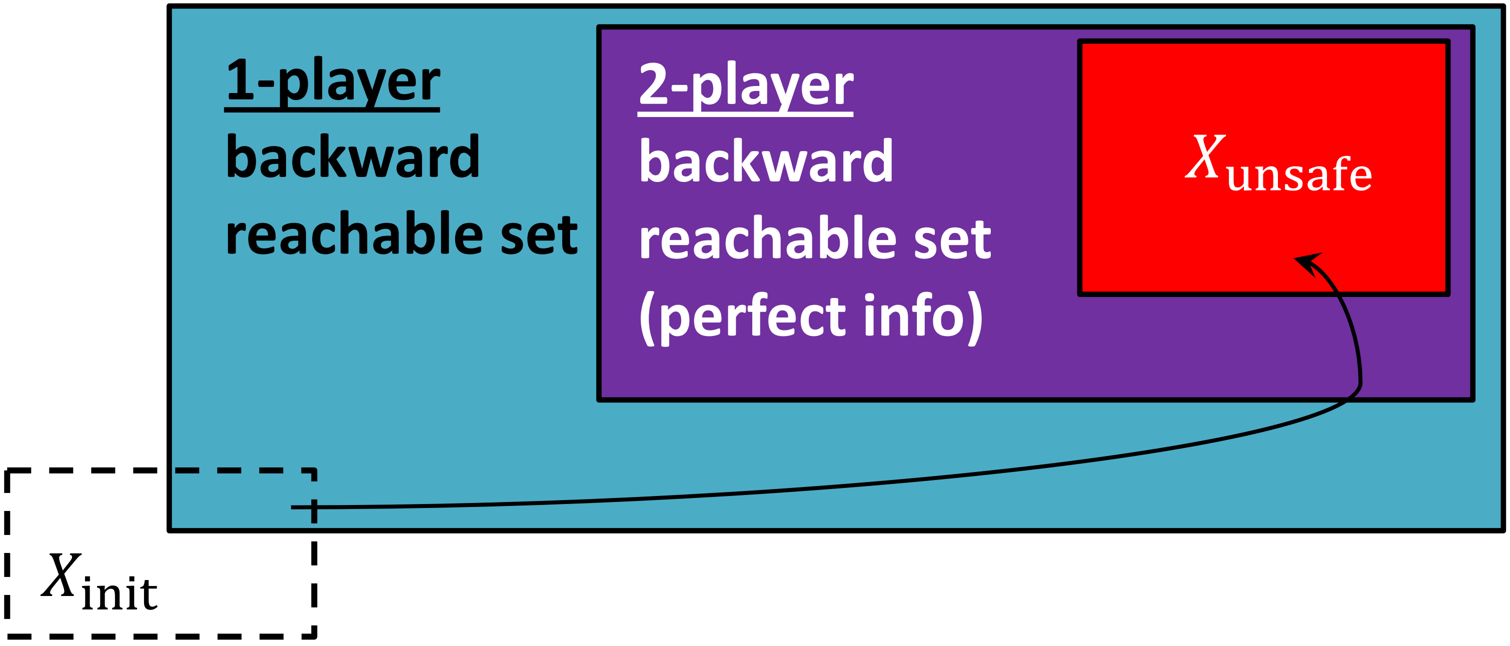
Key Idea

Synthesis: compute 2-player backward reachable set $= \bigcup \hat{X}_k$

$$\hat{X}_0 = X_{\text{unsafe}}$$
$$\hat{X}_{k+1} = \mathbf{CPre}(\hat{X}_k) := \hat{X}_k \cup \{x \mid \forall u : \underbrace{\exists w : f(x,u,w) \in \hat{X}_k}\}$$

<u>Open-loop</u> dynamics: simple, independent of π

- The adversarial examples are trivial (generic)
- Noise v is not essential for violation

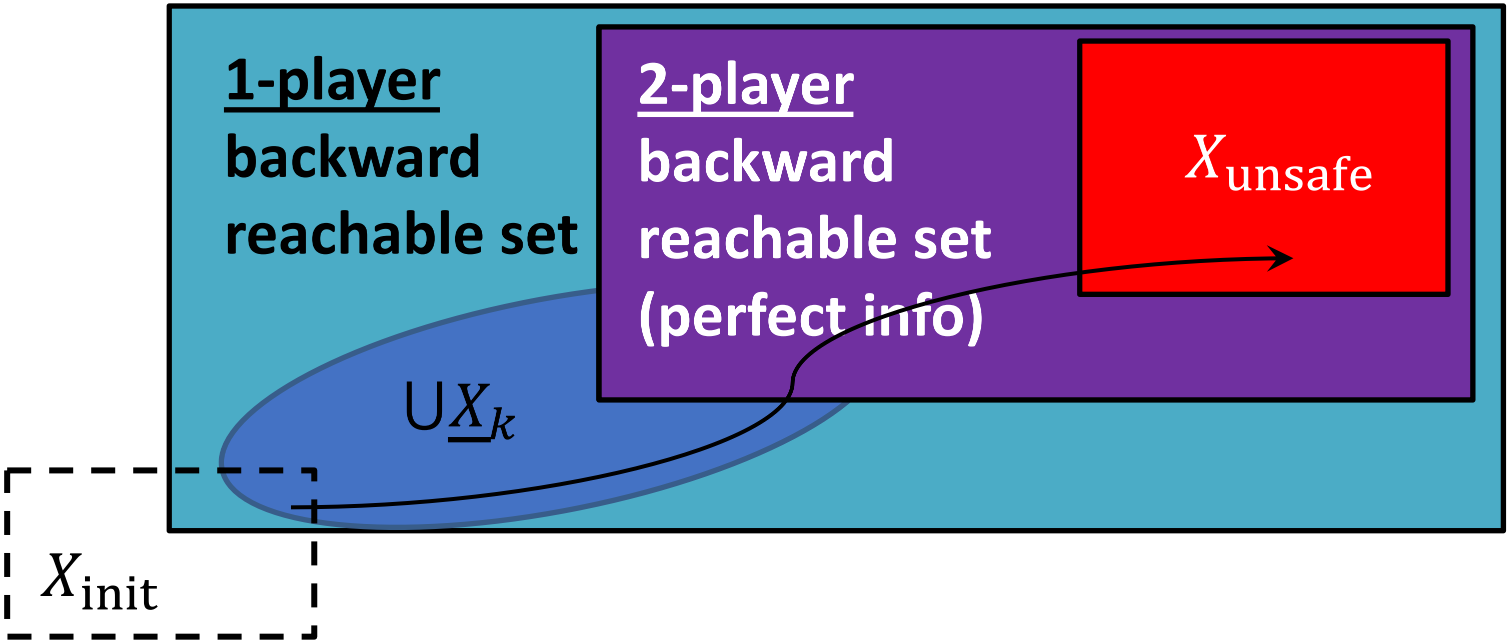


Key Idea

Synthesis guided Falsification: compute $\cup \underline{X}_k$, where

1. $\underline{X}_0 \subseteq$ 2−player backward reachable set
2. $Y_{k+1} = \mathbf{CPre}_y(\underline{X}_k) := \{y \mid \forall u: \exists x, v: y = g(x,v), f(x,u,w) \in \underline{X}_k\}$
3. $y_{k+1} \in Y_{k+1}$, $u_{k+1} = \pi(y_{k+1})$,
4. $\underline{X}_{k+1} = \mathbf{Pre}(\underline{X}_k \mid y_{k+1}) := \{x \mid \exists w, v: y_{k+1} = g(x,v), f(x, u_{k+1}, w) \in \underline{X}_k\}$

independent of π!

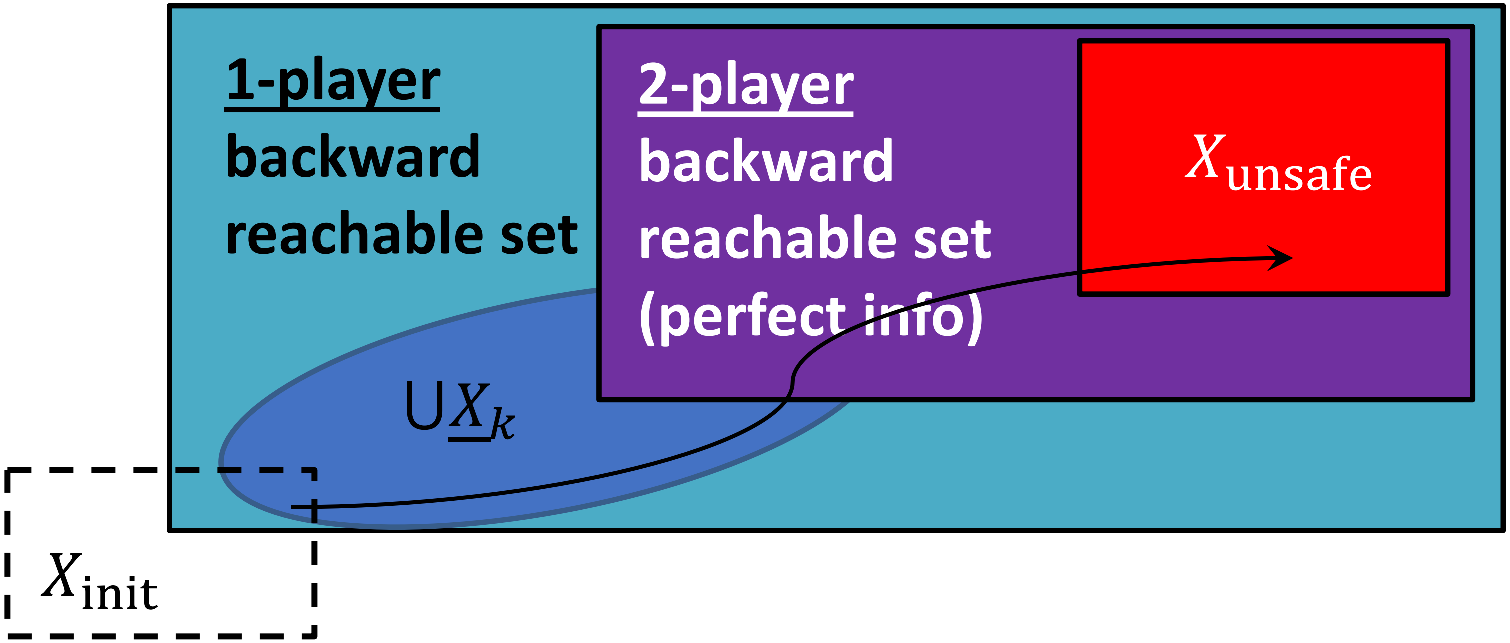


Key Idea

Synthesis guided Falsification: compute $\cup \underline{X}_k$, where

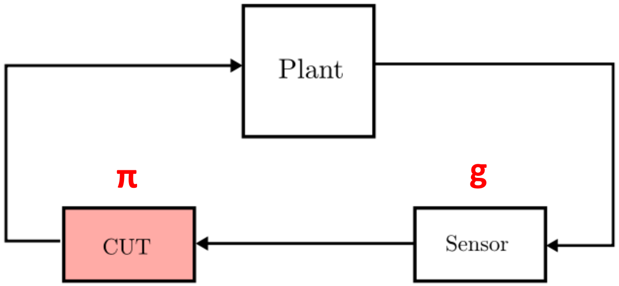
1. $\underline{X}_0 \subseteq$ 2−player backward reachable set
2. $Y_{k+1} = \mathbf{CPre}_{\nu}(X_k)$: From where it is challenging to satisfy the spec
3. query at the challenging region of the state space
4. $\underline{X}_{k+1} = \mathbf{Pre}(\underline{X}_k | y_{k+1})$ 1-player backward reachability based on the query result

independent of π!



A toy example

known & simple
(can be uncertain)



π

g

unknown & complicated

known & simple
(can be noisy)

an abstraction of a complex perception module

Plant model including sensor:

$$x_{t+1} = f(x_t, u_t, w_t)$$

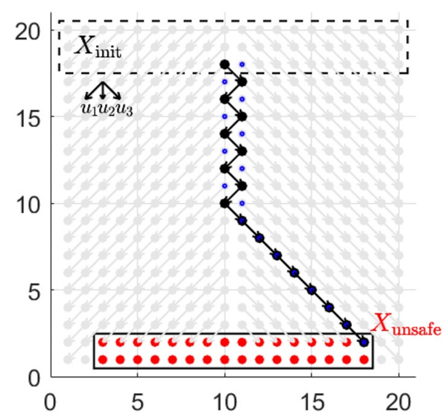
$$y_t = g(x_t, v_t)$$

Obstacle avoidance:

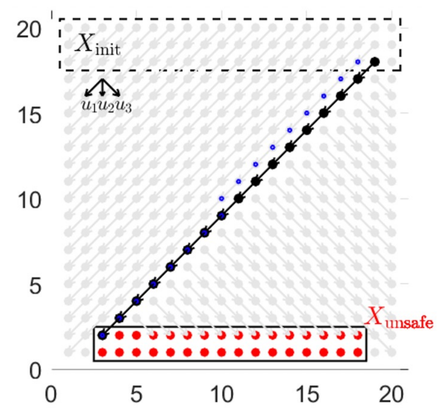
Moving down stream
Needs to avoid obstacle
Can go left or right (controller decisions)

Obstacle

Two different controllers for this task (unknown to the algorithm):



Controller 1



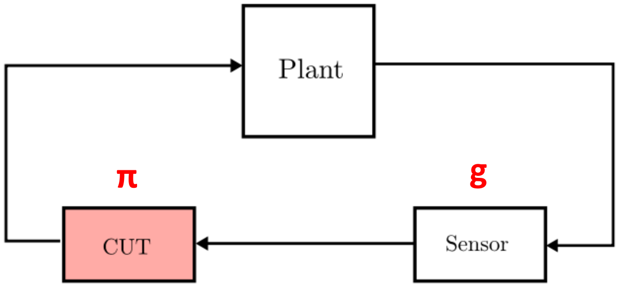
Controller 2

A toy example

known & simple
(can be uncertain)

π

g



unknown &
complicated

known & simple
(can be noisy)

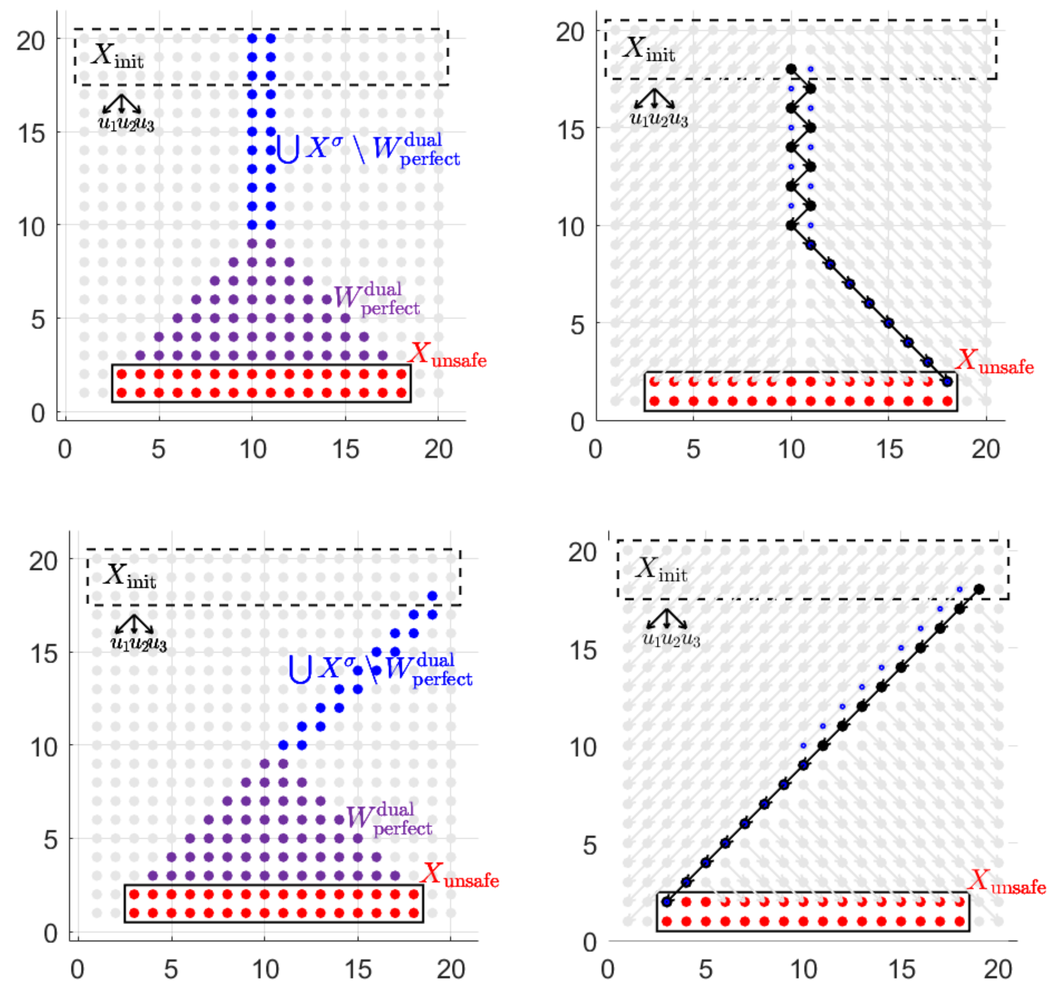
an abstraction of a
complex perception
module

Plant model including sensor:

$$x_{t+1} = f(x_t, u_t, w_t)$$

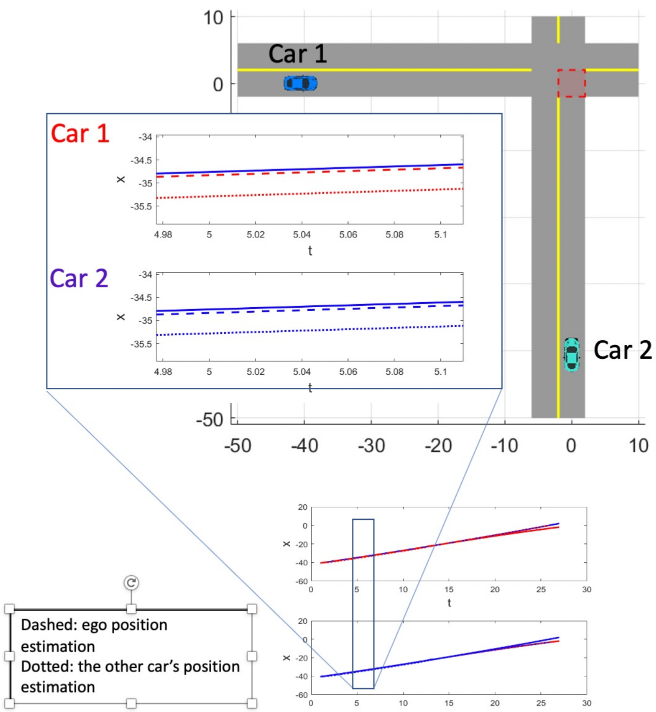
$$y_t = g(x_t, v_t)$$

Interesting insight: falsifying "sensor noise" trajectories at the discontinuities of the controller

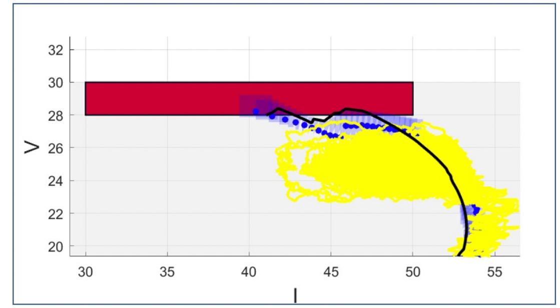


More examples with complex models/ controllers

Two cars at an intersection, 8-D dynamics, complex hybrid MPC controllers, each car has partial information of the other



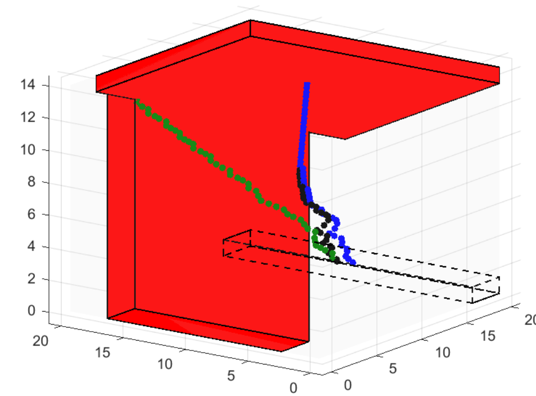
Buck converter with rule-based switching controller → forced to overvoltage



- Black: falsifying trajectory
- Yellow: simulations with random noise & disturbance (no violation)

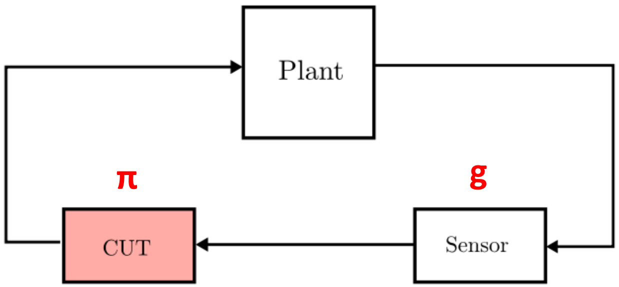
More complex specification including a deadline

$$\square(x \in X_{\text{safe}}) \wedge \lozenge_{[0,T]}\square(x \in X_{\text{target}})$$

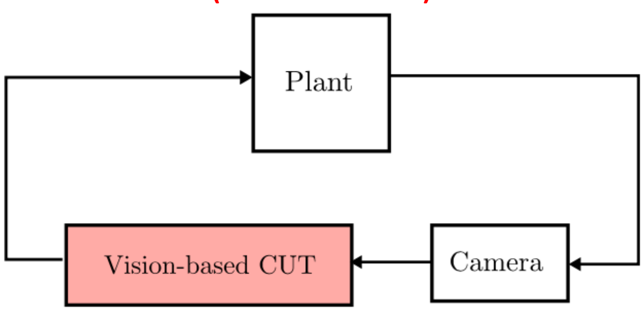


Falsification with perception in-the-loop

So far, we have

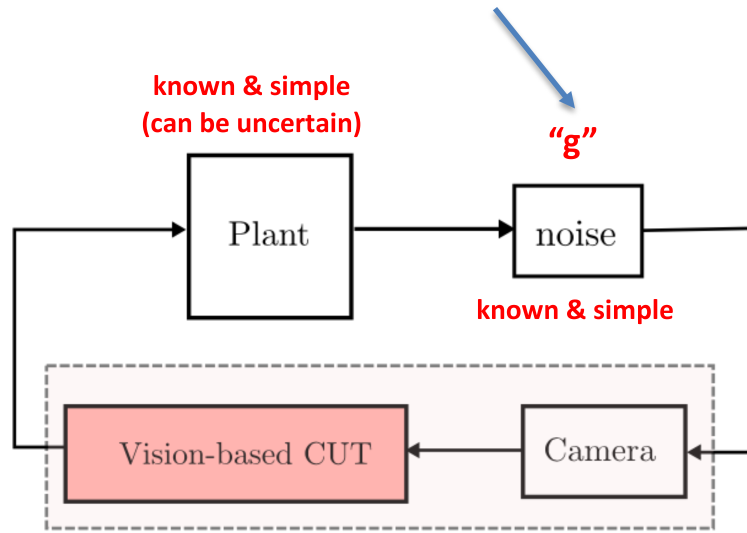


Yet, we want



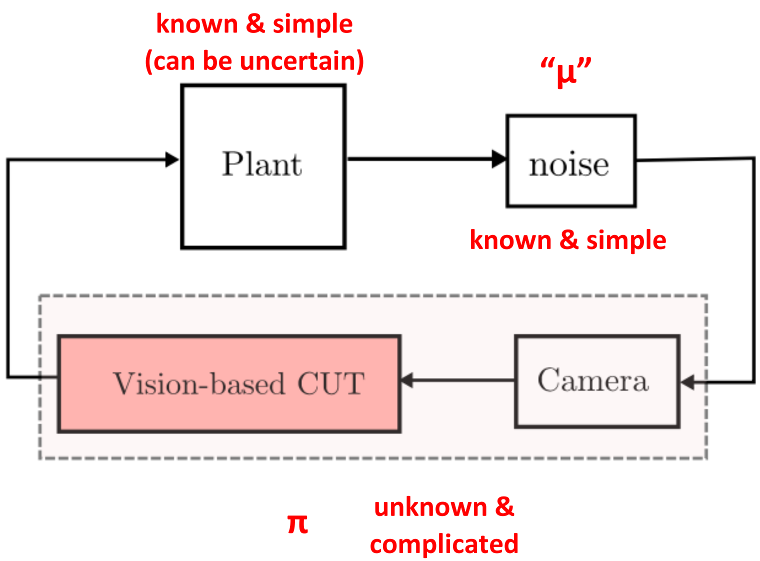
$\approx$

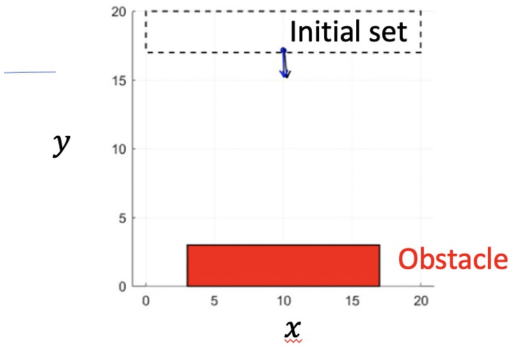
Searching for falsifying noise (adversary) in the semantic (state) space instead of image space



Vision-based CUT can be a modular design or an end-to-end controller..

"Adversarial inputs" for dynamic decision making with vision in-the-loop





Learn an end-to-end vision-based neural network controller from demonstrations generated by a state-feedback MPC

- Input: low resolution image
- Output: control input u (i.e., acceleration along x-direction)
- Camera model implemented in Matlab (or CARLA)

- a vehicle moving on the 2d plane
- dynamics along x-direction: double integrator

$$x(t+1) = x(t) + \tau * v_x(t)$$
$$v_x(t+1) = v_x(t) + \tau * u\,(t)$$

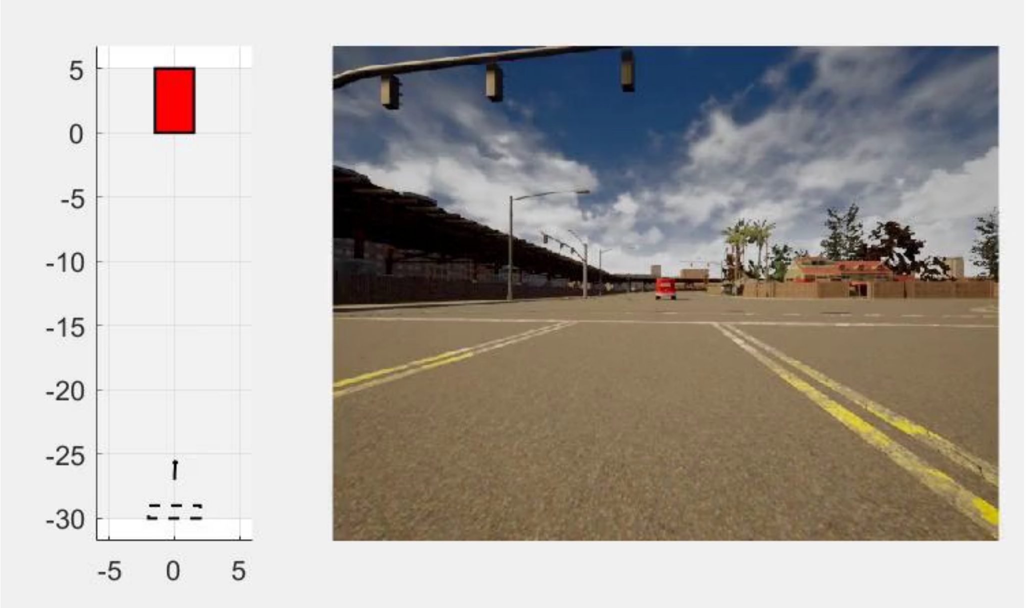
acceleration c is the control input

- dynamics along y-direction:

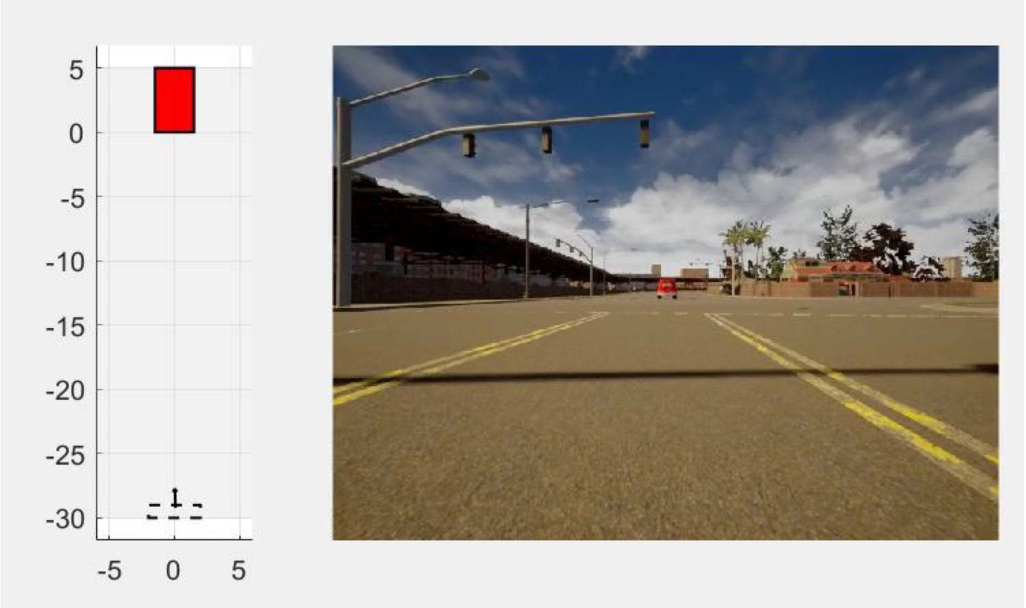
constant velocity + bounded disturbance w

$$y(t+1) = y(t) + \tau * \left(v_y(t) + w(t)\right)$$

Randomly perturbed images



Adversarially perturbed images with our falsification algorithm (similar perturbation magnitude)



Koopman-Inspired Safety Control for Unknown Nonlinear Systems

$z = \psi(x)$

Nonlinear system
$x^+ = f(x, u)$

$x = Cz$

Koopman over-approximation
$z^+ \in Az + Bu + W$

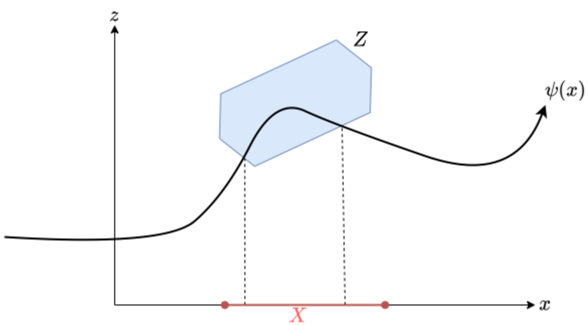
Hard

Easy

Inner approximates

Backward Reachable Set
X_t

Backward Reachable Set
Z_t



Two key ingredients:

- **Koopman over-approximation (KoA):** a simulation-like relation between the original system and Koopman-inspired abstraction

- **Implicit inner-approximation** Z of target set X where $\{x \mid \psi(x) \in Z\} \subseteq X$.

Balim, Aspeel, Liu, Ozay, L-CSS'23

Koopman-Inspired Safety Control for Unknown Nonlinear Systems

Nonlinear system
$x^+ = f(x,u)$

$z = \psi(x)$

$x = Cz$

Koopman over-approximation
$z^+ \in Az + Bu + W$

Hard

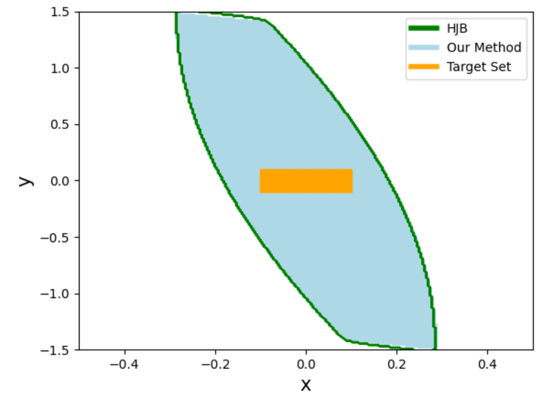
Easy

Inner approximates

Backward Reachable Set
X_t

Backward Reachable Set
Z_t

Forced Duffing Oscillator



Some properties:

- Any lifting function ψ including x in its coordinates can be used;
- The Koopman over-approximation is learned from data;
- If we can estimate local Lipschitz constants, can improve computation further by updating the linear representation locally

Balim, Aspeel, Liu, Ozay, L-CSS’23

Koopman-Inspired Safety Control for Unknown Nonlinear Systems

Nonlinear system
$x^+ = f(x,u)$

$z = \psi(x)$ ⇄ $x = Cz$

Koopman over-approximation
$z^+ \in A_i z + B_i u + W_i$ if $x \in D_i$

Hard ↓

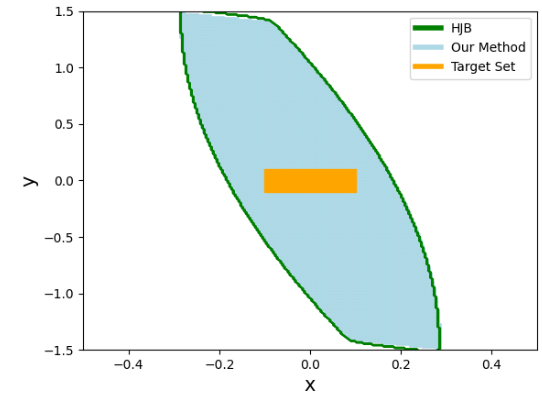
Easy ↓

Backward Reachable Set
X_t

Inner approximates ←

Backward Reachable Set
Z_t

Forced Duffing Oscillator

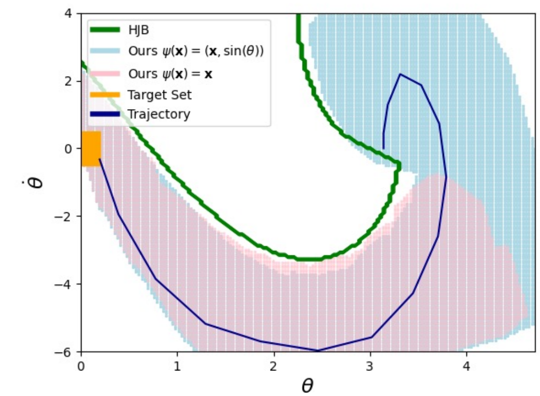


Single linearization is not enough:

- Different over-approximations are learned over local subdomains (leading to a PWA system) for better accuracy:
  - Experiments show that to obtain BRSs with similar sizes, the Koopman over-approximation requires less pieces than direct linearization (hybridization).

  Why do we need hybridization in the lifted space? See Liu, Ozay, Sontag IFAC WC'23 paper on non-existence of linear immersions for systems with multiple omega limit sets

Inverted Pendulum



Balim, Aspeel, Liu, Ozay, L-CSS'23

Koopman-Inspired Safety Control for Unknown Nonlinear Systems

$z = \psi(x)$

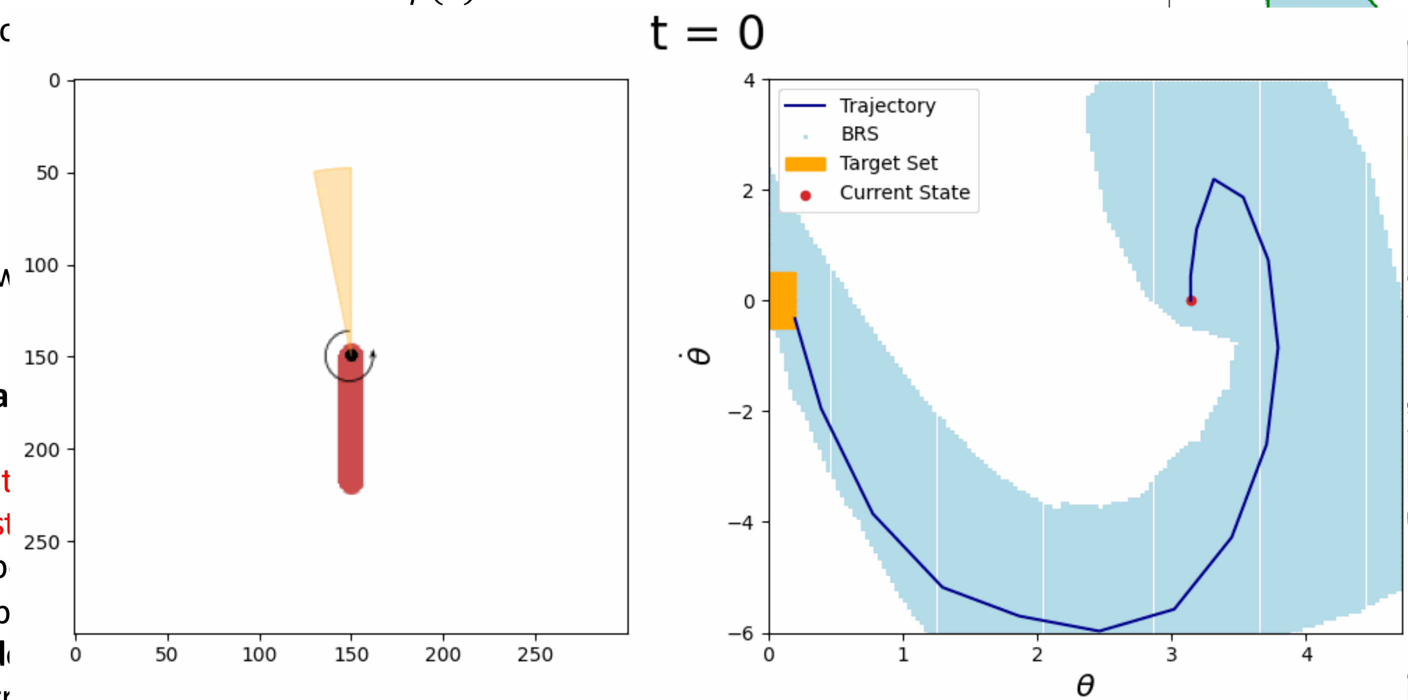
Nc

Backw

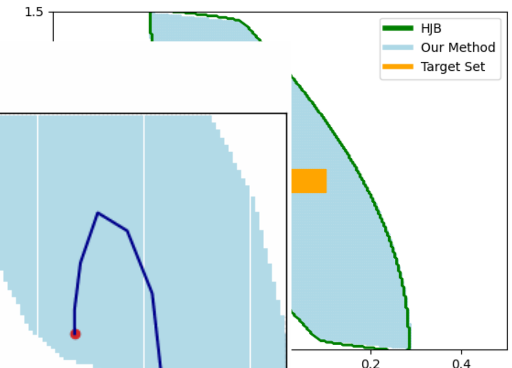
Single linea

- Different
  PWA syst
  - Exp
    app

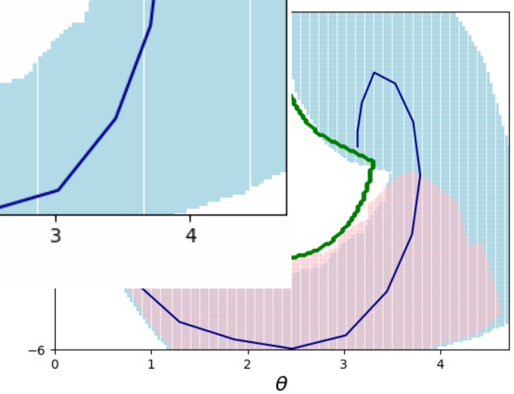
Why d
tomorr
with multiple omega limit sets



Forced Duffing Oscillator



ndulum



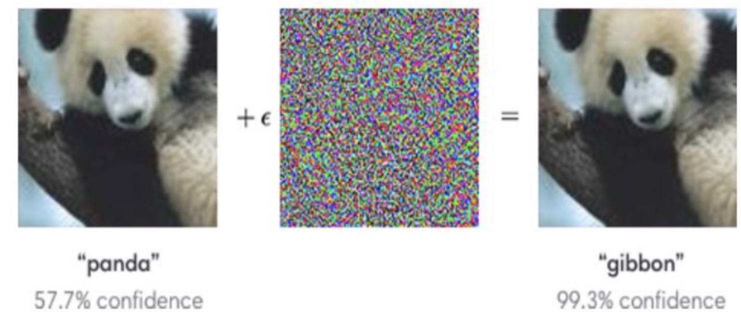
Balim, Aspeel, Liu, Ozay, L-CSS'23

Summary and conclusions

Key takeaways:

- Zonotopes and constrained zonotopes for backward reachability → applications in synthesis and falsification
- A new framework: synthesis-guided falsification:
  - Leads to explainable counterexamples
  - Works with blackbox controllers (code)
  - Extends to vision/perception-based control or end-to-end learning controllers
- An interesting connection between adversarial examples in machine learning and those in decision-making
- We can also do backward reachability for nonlinear systems using liftings

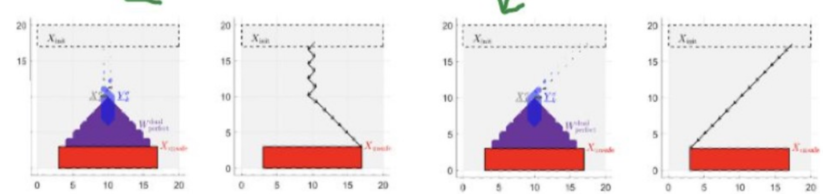
Machine learning:



Adversarial examples occur at decision boundaries in classification

Decision making (obstacle avoidance):

Controller 1: Controller 2:



Adversarial examples occur at decision boundaries

i.e., discontinuities of the controller

Linear programming (LP)

- An optimization problem

$$\begin{array}{lll} \text{minimize} & f_0(x) & \\ \text{subject to} & f_i(x) \leq 0 & \text{for all } i = 1, \ldots, m \\ & h_i(x) = 0 & \text{for all } i = 1, \ldots, p \end{array}$$

is an LP problem if f_i for $i=0,\ldots,m$ and h_i for $i=1,\ldots,p$ are affine functions. An LP is typically written in the following form:

$$\begin{array}{ll} \text{minimize}_x & c_0^T x \\ \text{subject to} & Ax \leq b \\ & Cx = d \end{array}$$

In other words, an LP problem is an optimization problem whose objective function is linear and feasible set is defined by a polyhedron.