Synthesis-guided Adversarial Example Generation for Gray-box Autonomous Systems with Sensing Imperfections

Liren Yang
4th March 2021
Autonomous Control Systems

![Control System Diagram](image)

- **Plant**: The system being controlled
- **Sensor**: Measures the system's state
- **Controller**: Processes the input to control the system
- **Observation**: System state
- **Control**: Input to the system
- **Disturbance**: External influence on the system
- **Noise**: Internal disturbance
- **State**: Current state of the system

Control $u$, Observation $y$, Noise $v$, Disturbance $w$, State $x$
Autonomous Control Systems

- Complex components
  - Controller: millions of lines of code (learning, optimization...)

![Diagram of control system components]

- Plant
- Sensor
- Controller
- Control $u$
- Observation $y$
- Disturbance $w$
- State $x$
- Noise $\nu$
Autonomous Control Systems

- Complex components
  - Controller: millions of lines of code (learning, optimization...)
  - Sensor: imperfect & hard to formalize (e.g., camera)
Autonomous Control Systems

- Complex components
  - Controller: millions of lines of code (learning, optimization...)
  - Sensor: imperfect & hard to formalize (e.g., camera)
- Uncertain environment
Autonomous Control Systems

- Complex components
  - Controller: millions of lines of code (learning, optimization...)
  - Sensor: imperfect & hard to formalize (e.g., camera)
- Uncertain environment
- Safety-critical requirement

We want state $x \models$ Specification (e.g., safety)

plant $x$ control $u$

sensor $y$ observation $\nu$
disturbance $w$
Autonomous Control Systems

- Complex components
  - Controller: millions of lines of code (learning, optimization...)
  - Sensor: imperfect & hard to formalize (e.g., camera)
- Uncertain environment
- Safety-critical requirement

**Question:** can we prove an autonomous system to be safe under all circumstances?

We want

state $x \models$ Specification (e.g., safety)
Verification

• Pro: strong conclusion
• Con: computationally challenging

White-box

Plant

Controller  Sensor

Spec $\varphi$

Certificate

Counter-example

$x_0, w_{0:t}, v_{0:t}$ s.t. $x_{0:t}$ violates the spec $\varphi$
Alternative 1: Synthesis

• Pro: avoid the complexity of controller, “explain” fundamental limits (impossibility)
• Con: even more difficult, difficulty $\uparrow$ with sensing error difficulty $\uparrow\uparrow$ with vision
Alternative 2: Falsification

- Pro: computationally cheaper
- Con: weaker conclusion, lack of interpretability

\[ x_0, w_{0:t}, v_{0:t} \text{ s.t. } x_{0:t} \text{ violates the spec } \varphi \]
Alternative 2: Falsification

- Pro: computationally cheaper
- Con: weaker conclusion, lack of interpretability

**Question:** When a plant model is at hand, can we combine synthesis and falsification to find more “meaningful” counter-examples?
A Typical Adversarial Example

- Plant: integrator-like dynamics
- Sensor: a camera, simulated with Carla¹
- End-to-end controller: a feedforward neural network
- Spec: avoid the obstacle (red) in the front, without decelerating

1. A. Dosovitskiy et al. “CARLA: An open urban driving simulator”. 2017
Our Gray-box Setting:

- **White-box components:**
  - Plant model \( f \)
  - Sensor model \( g \), to be relaxed later

- **Black-box component**
  - Controller \( \pi \): static, i.e., no memory (e.g., rule-based, MPC, feedforward neural network)
  - Can be “queried”: given \( y \), we get \( u = \pi(y) \)

\[
x_{t+1} = f(x_t, u_t, w_t)
\]

\[
y_t = g(x_t, v_t)
\]
Problem Statement

Given:
- A gray-box system
- An unsafe set $X_{\text{unsafe}}$, and an initial set $X_{\text{init}}$

Find one adversarial example:
- A trajectory $x_0, x_1, \ldots, x_T$
- External inputs $w_0, w_1, \ldots, w_{T-1}, v_0, v_1, \ldots, v_{T-1}$
- $x_0 \in X_{\text{init}}$ and $x_T \in X_{\text{unsafe}}$
Problem Statement

Given:
- A gray-box system
- An unsafe set $X_{unsafe}$, and an initial set $X_{init}$

Find:
- A trajectory $x_0, x_1, \ldots, x_T$
- External inputs $w_0, w_1, \ldots, w_{T-1}$, $v_0, v_1, \ldots, v_{T-1}$
- $x_0 \in X_{init}$ and $x_T \in X_{unsafe}$

Questions:
- **Interpretability**: Can we say anything like:
  - the bug is specific to the given controller,
  - the imperfect sensing is the root reason of this bug
- **Smart Query**: How do we find safety violations specific to a controller, without exhaustively querying the controller?

Key Idea: Use synthesis to guide the search
Key Idea

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

\[
X_0 = X_{\text{unsafe}} \\
X_{k+1} = \text{Pre}(X_k) := X_k \cup \{x | \exists w, v: f(x, \pi(x, v), w) \in X_k\}
\]

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

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

$\hat{X}_0 = X_{\text{unsafe}}$

$\hat{X}_{k+1} = \text{CPre}(\hat{X}_k) := \hat{X}_k \cup \{x | \forall u: \exists w: f(x, u, w) \in \hat{X}_k\}$

Open-loop dynamics: simple, independent of $\pi$
- The adversarial examples are trivial (generic)
- Noise $\nu$ is not essential for violation
Key Idea

Synthesis guided Falsification: compute $\bigcup X_k$, where

1. $X_0 \subseteq$ 2-player backward reachable set
2. $Y_{k+1} = \text{CPre}_y(X_k) := \{y | \forall u: \exists x, v: y = g(x, v), f(x, u, w) \in X_k\}$
3. $y_{k+1} \in Y_{k+1}$, $u_{k+1} = \pi(y_{k+1})$
4. $X_{k+1} = \text{Pre}(X_k | y_{k+1}) := \{x | \exists w, v: y_{k+1} = g(x, v), f(x, u_{k+1}, w) \in X_k\}$
**Key Idea**

Synthesis guided Falsification: compute $\bigcup X_k$, where

1. $X_0 \subseteq \text{2-player backward reachable set}$
2. $Y_{k+1} = \text{CPre}_y(X_k)$: where it is trickier to satisfy the spec
3. query at the tricky place
4. $X_{k+1} = \text{Pre}(X_k | y_{k+1})$ 1-player backward reachability based on the query result

---

**Diagram:**

- **1-player backward reachable set**
- **2-player backward reachable set (perfect info)**
- $X_{\text{init}}$
- $\bigcup X_k$
- $X_{\text{unsafe}}$

---

Liren Yang  
Synthesis-guided Adversarial Example Generation
A Toy Example

- We find controller-specific violations using the same algorithm.

- Violations occur at “decision boundaries”.
A Toy Example

- We find controller-specific violations using the same algorithm.

- Violations occur at “decision boundaries”.

Machine learning:

- “panda” 57.7% confidence
- “gibbon” 99.3% confidence

Adversarial examples occur at decision boundaries in classification.

Diagram showing feedforward neural network.
A Toy Example

- We find controller-specific violations using the same algorithm.

- Violations occur at “decision boundaries”.

Decision boundaries are everywhere.

Which car goes first?

Left or right?
Vision-in-the-loop Systems

$\Sigma_1$: I know how to falsify this

$\Sigma_2$: What I want to falsify

$\Sigma_3$: What I will falsify instead

**Idea**: instead of perturbing the images ($y$, high dim), perturb the state ($x$, low dim) where the image is taken.
More Examples

More complex specification including a deadline

\[ \square(x \in X_{\text{safe}}) \land \Diamond_{[0,T]}(x \in X_{\text{target}}) \]

Buck converter with rule-based switching controller \( \rightarrow \) forced to overvoltage

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

- A new framework: synthesis-guided falsification:
  - Leads to explainable counterexamples
  - Works with black-box 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

Thank you!

Q & A