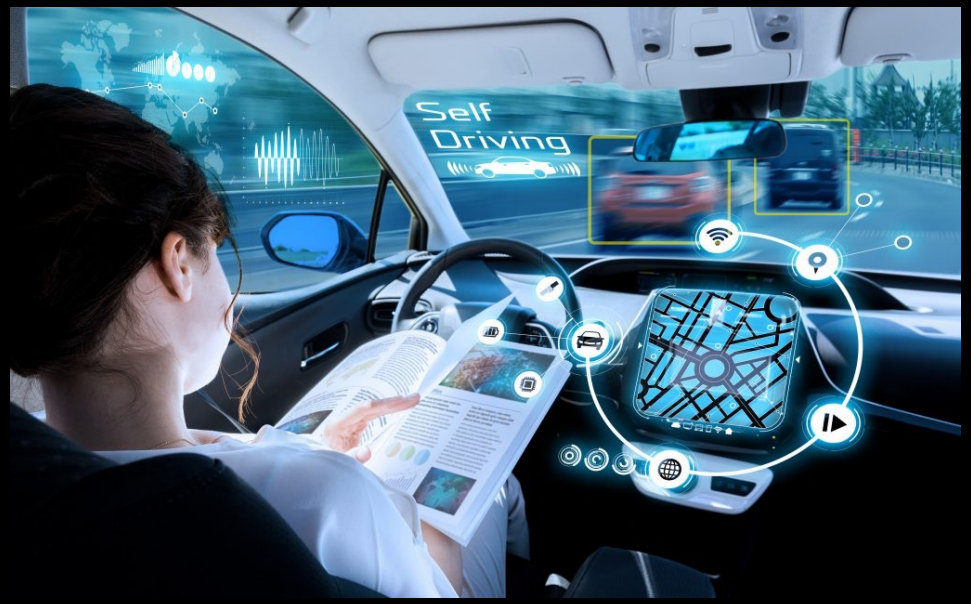
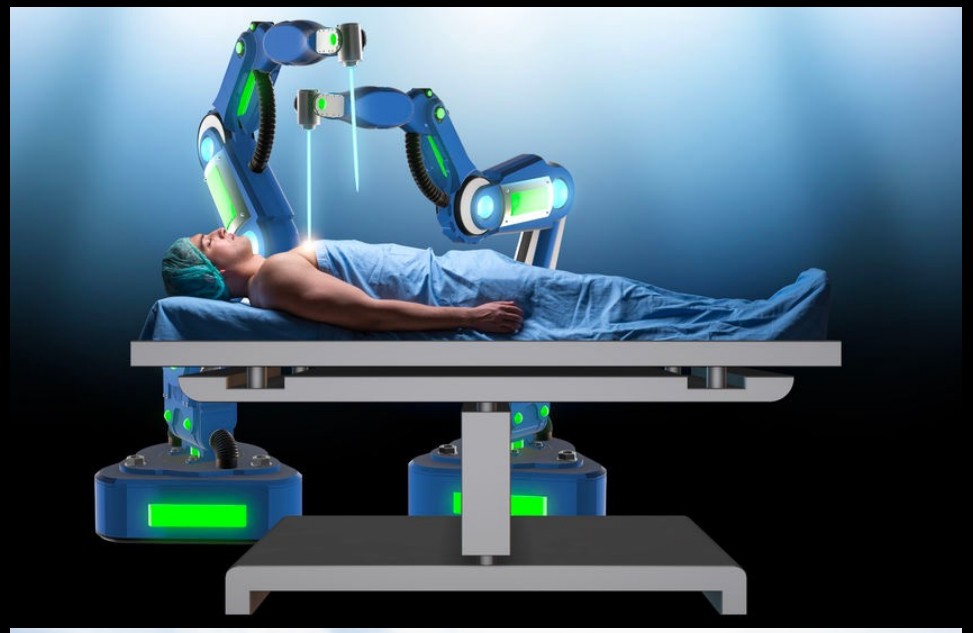
Formal methods for Cyber Physical Systems

Necmiye Ozay
Electrical Engineering and Computer Science, University of Michigan

2022 NSF CPS PI Meeting

November 9, 2022





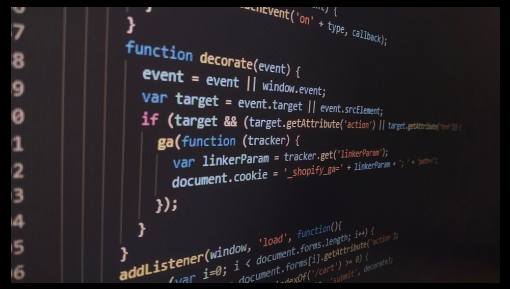
End-to-end guarantees on safety and task completion?

Formal methods: mathematically rigorous techniques for specification, verification, and design of software and hardware systems.

Formal methods landscape

Plant
(with a model)

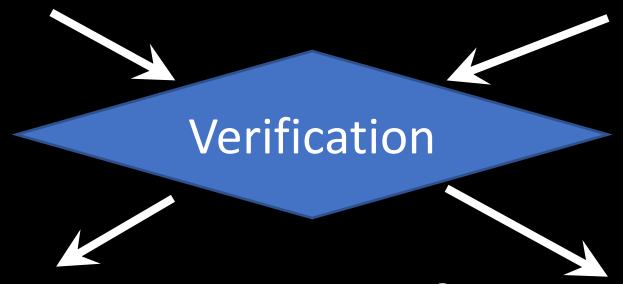
+



decision-making
(autonomy) software

Whitebox system:
Plant + software

Specification



Correctness
proof

Counterexample

PROS:

- Strong guarantees

CONS:

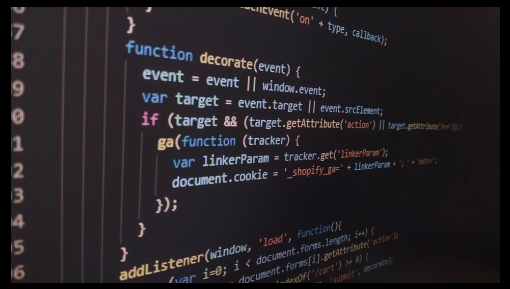
- Hybrid system verification is computationally very hard
- Autonomy software ➔ up to millions of lines of code (loc):
  - hard to model
  - hard to scale

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

Formal methods landscape

Plant
(with a model)

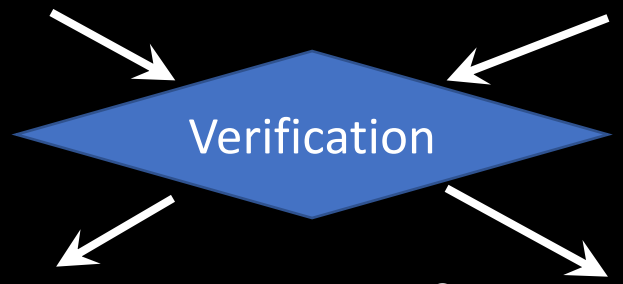
+



decision-making
(autonomy) software

Whitebox system:
Plant + software

Specification



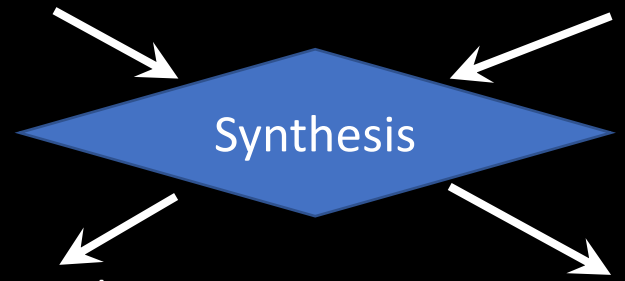
Correctness
proof

Counterexample

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

Partial whitebox system:
Plant (∃? Software)

Specification



Software with
correctness guarantee +
validity domain

Proof of
impossibility

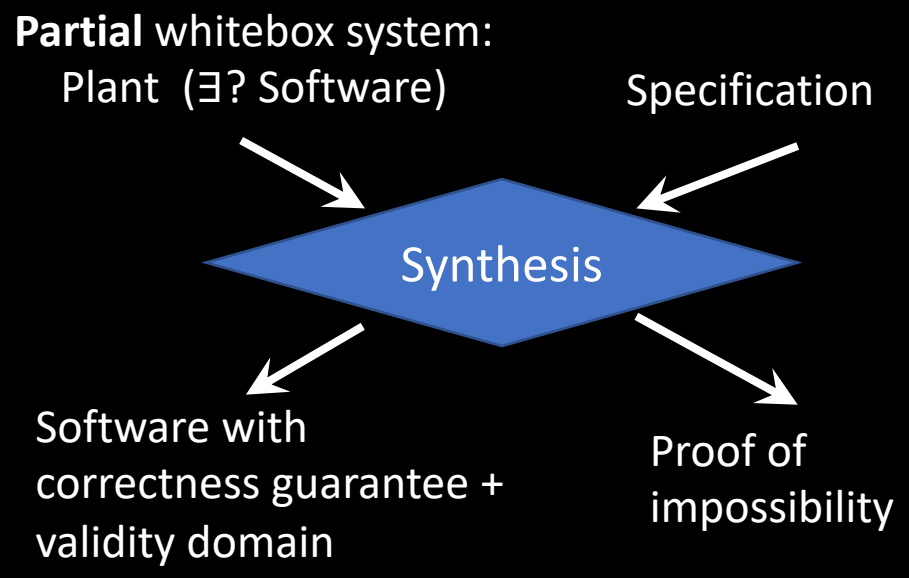
PROS:

- 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

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



We have worked on many applications in the automotive and aerospace domain.
Two highlights (from ’14-’15):

- Adaptive cruise control
- Lane keeping

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

Partial whitebox system:
Plant (∃? Software)

Specification

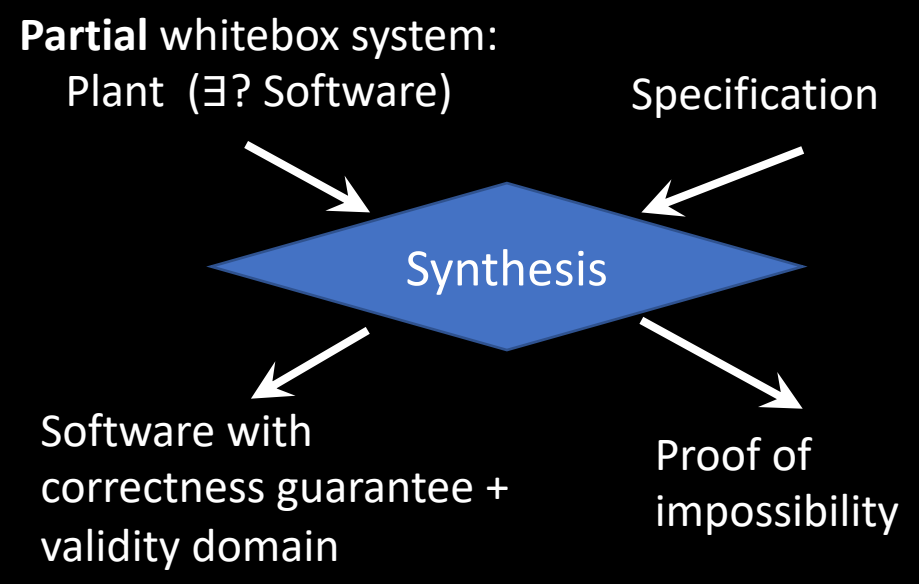
Synthesis

Software with correctness guarantee + validity domain

Proof of impossibility

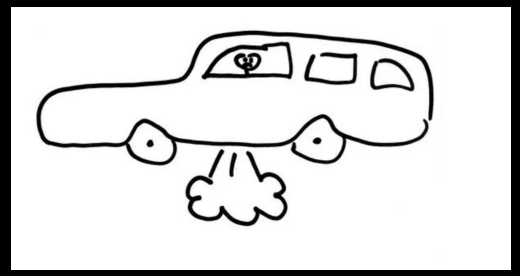
Deployment of synthesized controllers in Mcity

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



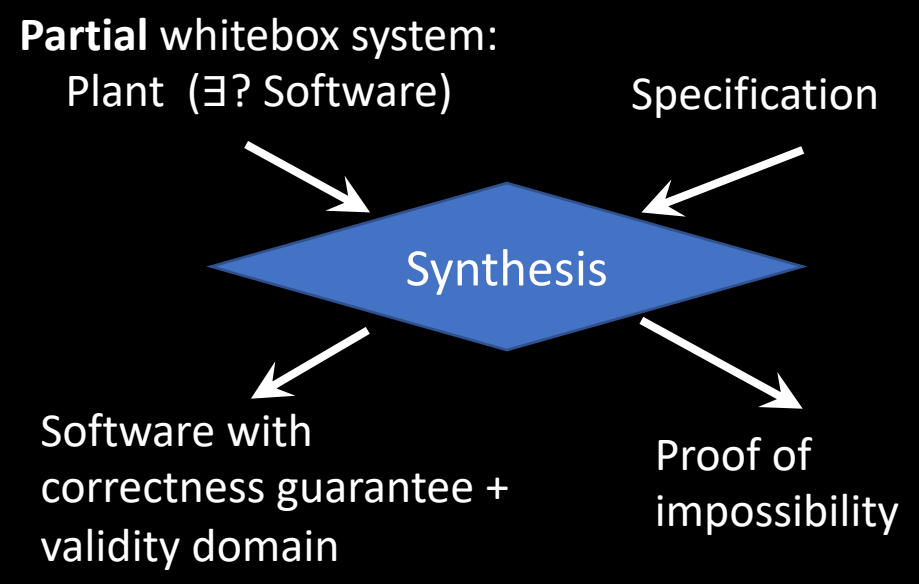
What we learned from early deployments?

- 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 (see the poster session)

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

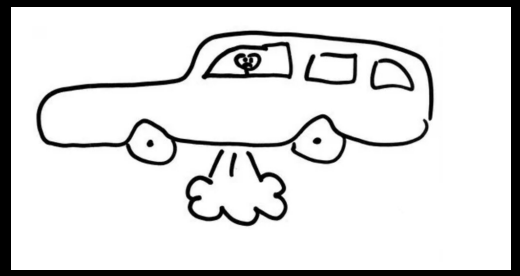


Theorem:

For an n dimensional linear system model with k steps of actuation delay (overall system dimension = n+k), there exist an auxiliary n dimensional system such that any safe and invariant set of the original delay system can be obtained from that of the auxiliary system.

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

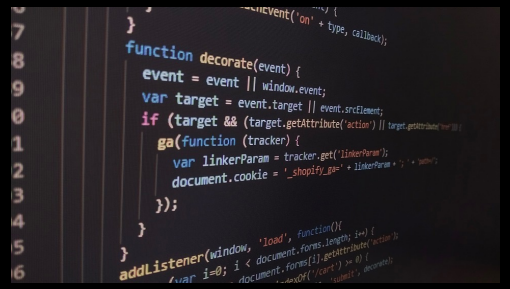


- Motivated future work on safety with learned models, delays, and predictions (see the poster session)

Formal methods landscape

Plant
(with a model)

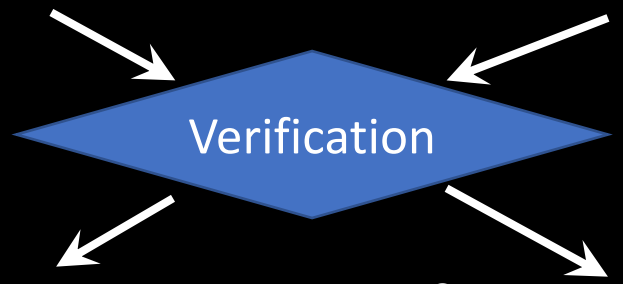
+



decision-making
(autonomy) software

Whitebox system:
Plant + software

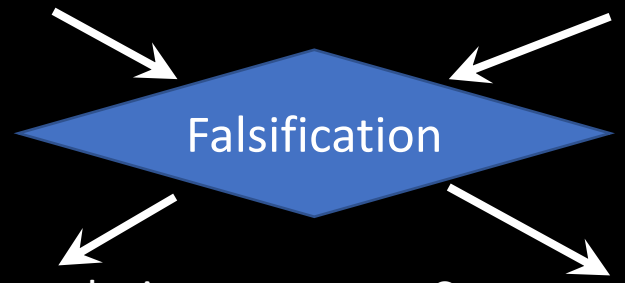
Specification



Alternative #2: Falsification

Blackbox system:
Plant + software

Specification



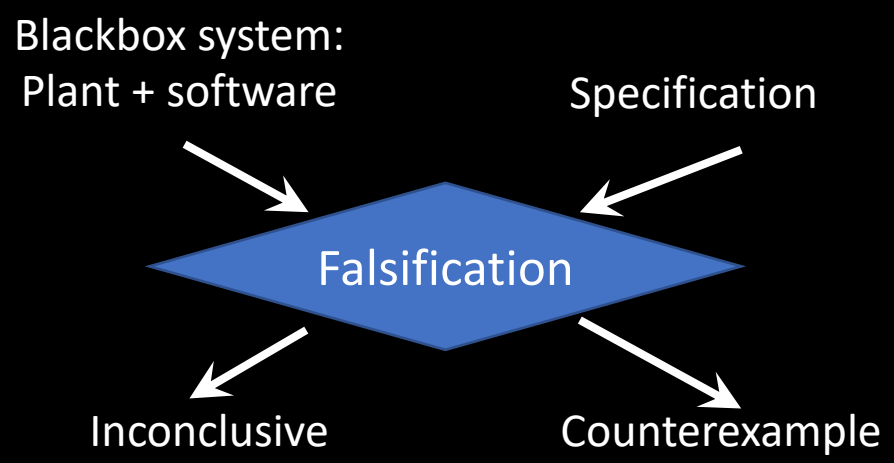
PROS:

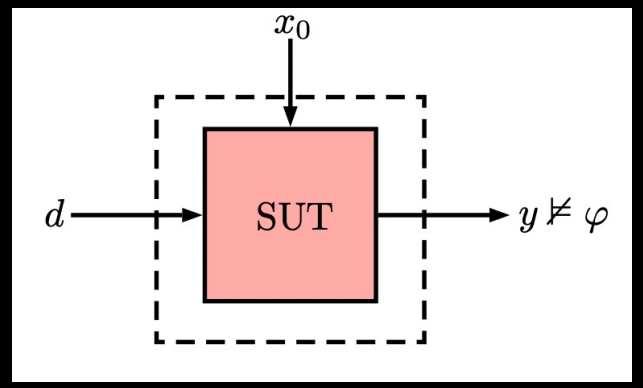
- Can handle arbitrarily complex models (plant + software) including learning-based components
- Industry-adopted tools (e.g., Breach, S-Taliro)

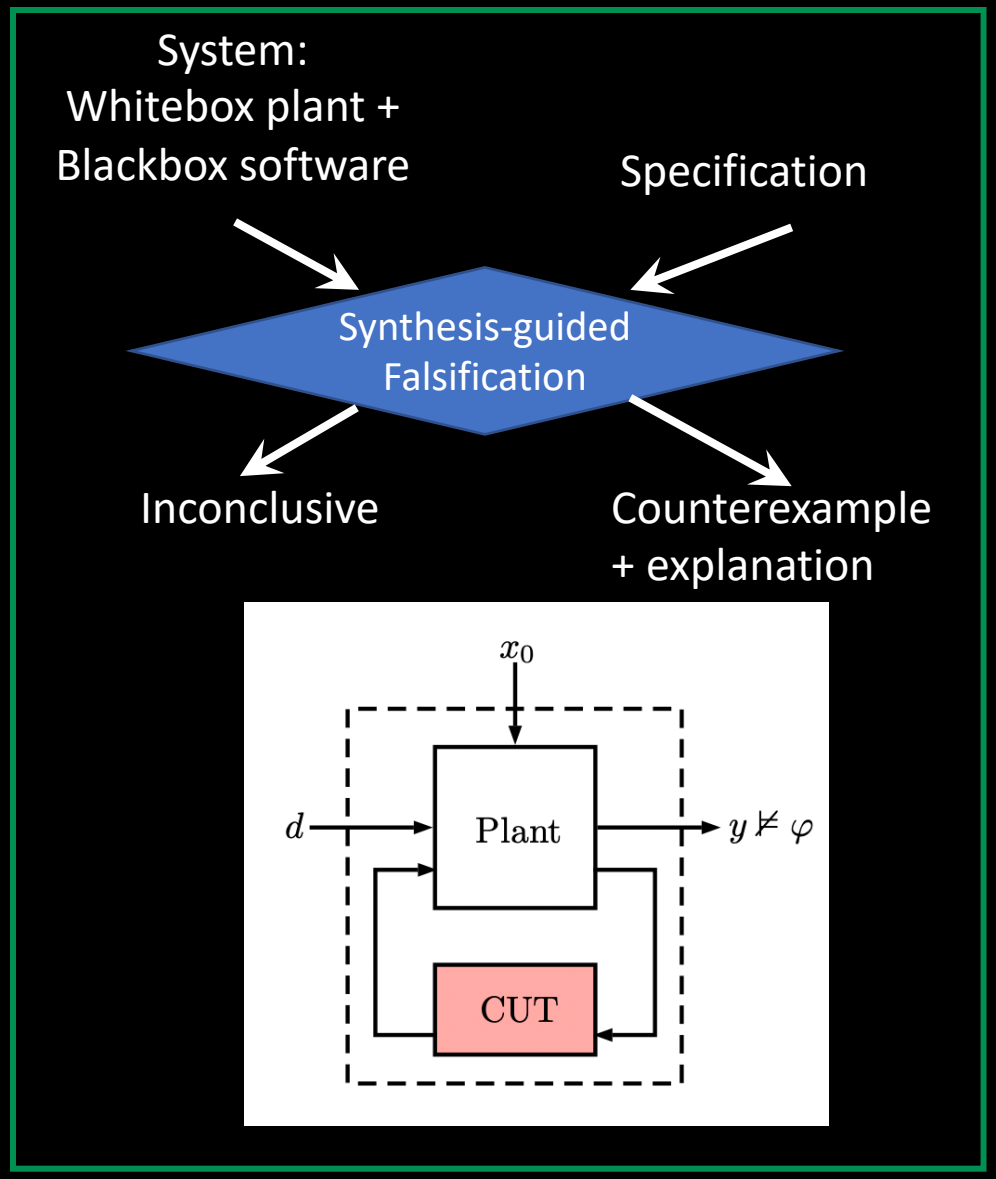
CONS:

- 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

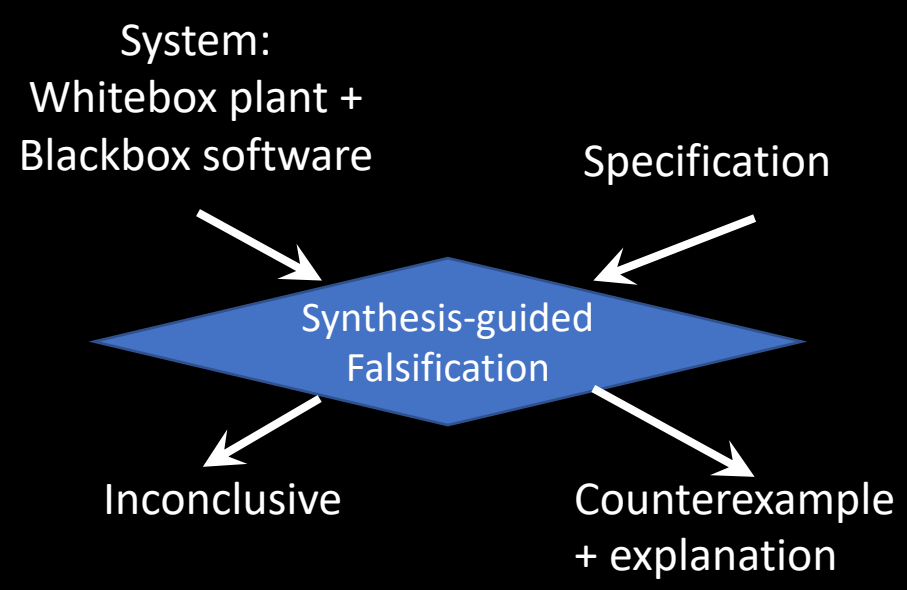






* SUT: system under test; CUT: controller (autonomy software) under test

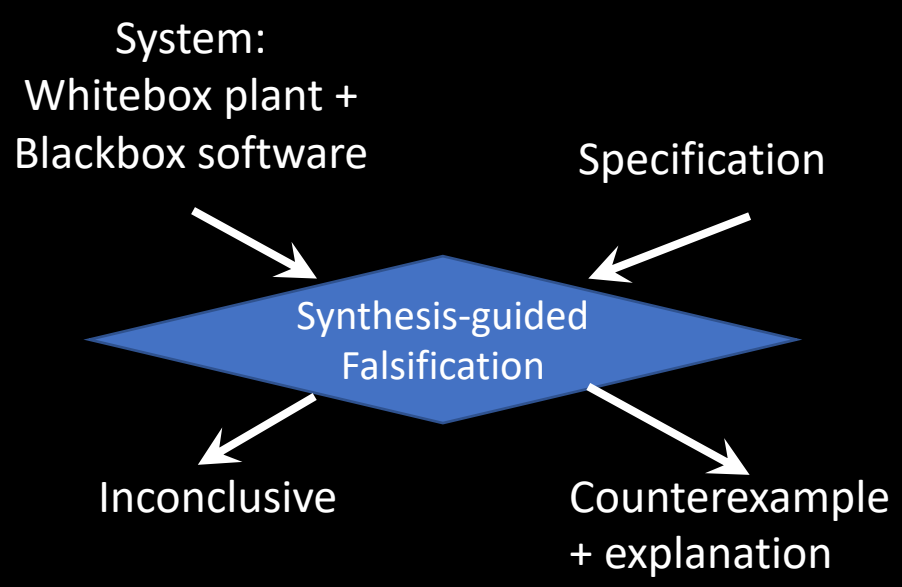
Alternative #2.5: Synthesis-guided falsification

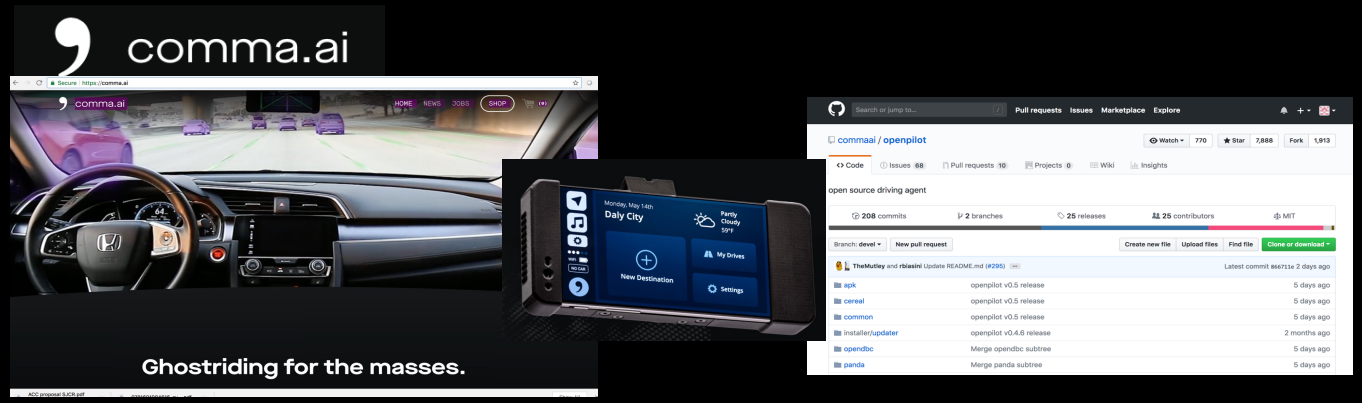


Key insights:

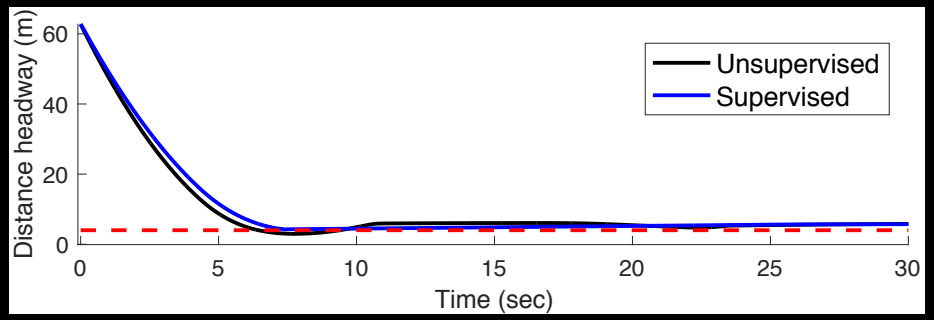
- Steering the search via specification-guided backward reachable sets of the plant model
- Querying the unknown controller only at specification-critical regions

Alternative #2.5: Synthesis-guided falsification

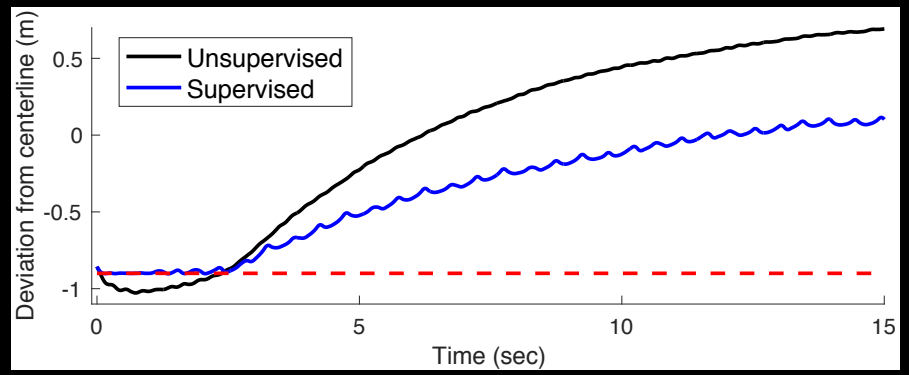




Adaptive cruise control



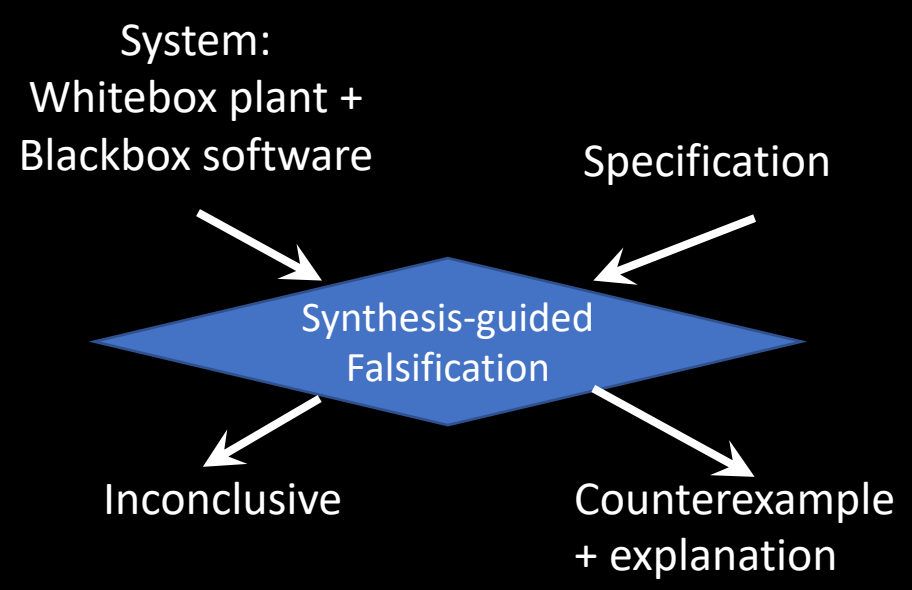
Lane keeping



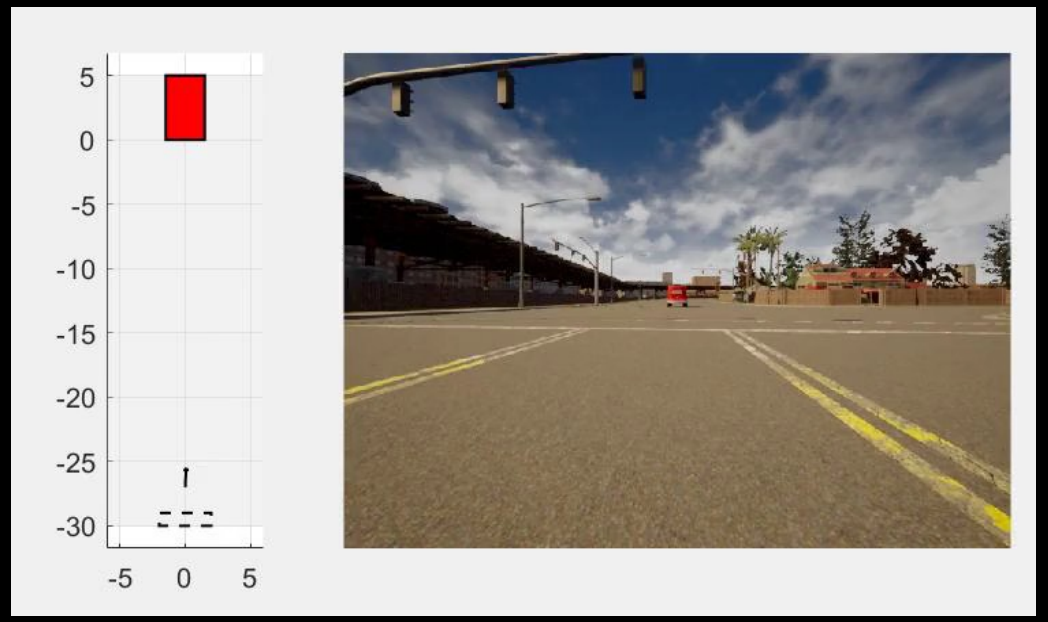
Key insights:

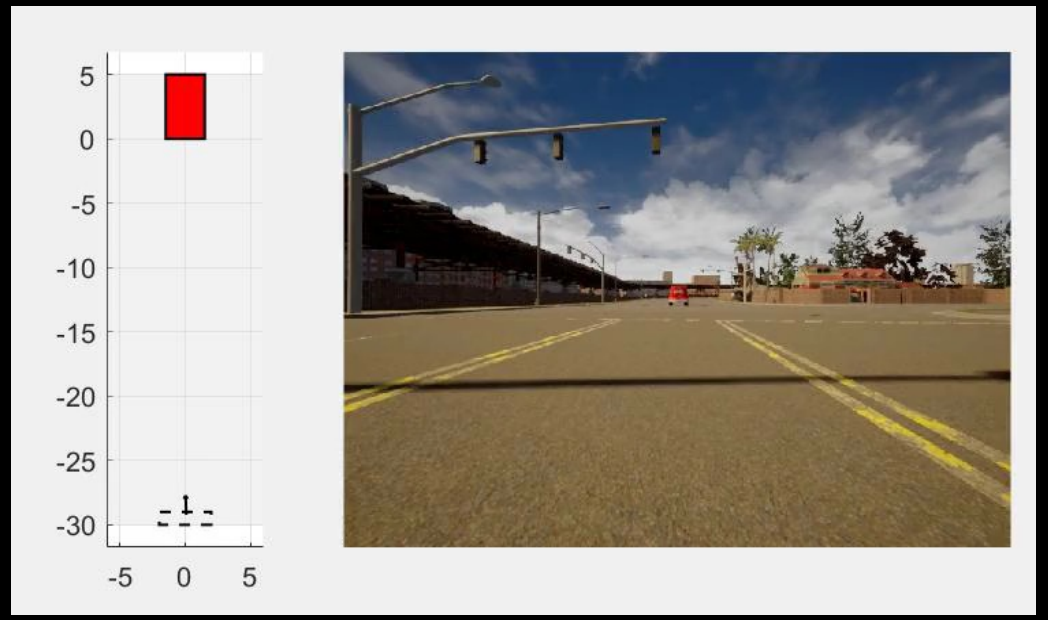
- Steering the search via specification-guided backward reachable sets of the plant model
- Querying the unknown controller only at specification-critical regions

Alternative #2.5: Synthesis-guided falsification



Extensions to vision-based controllers

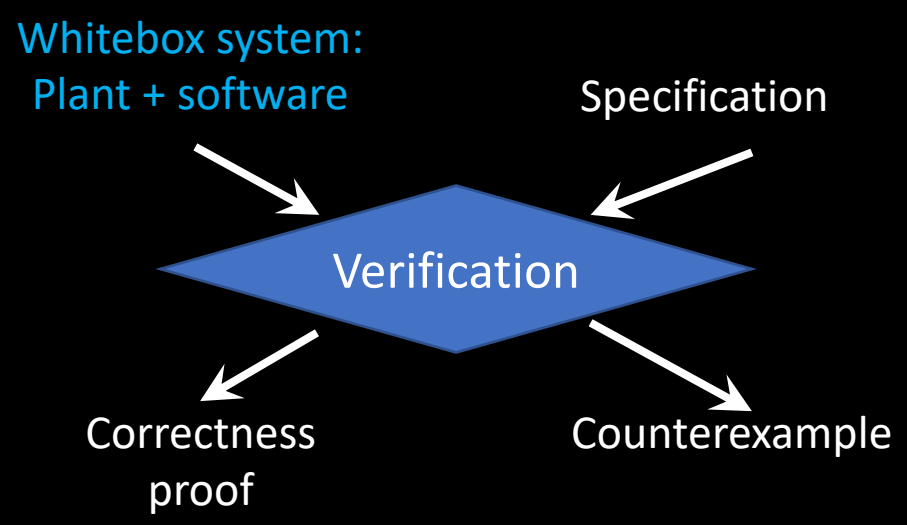


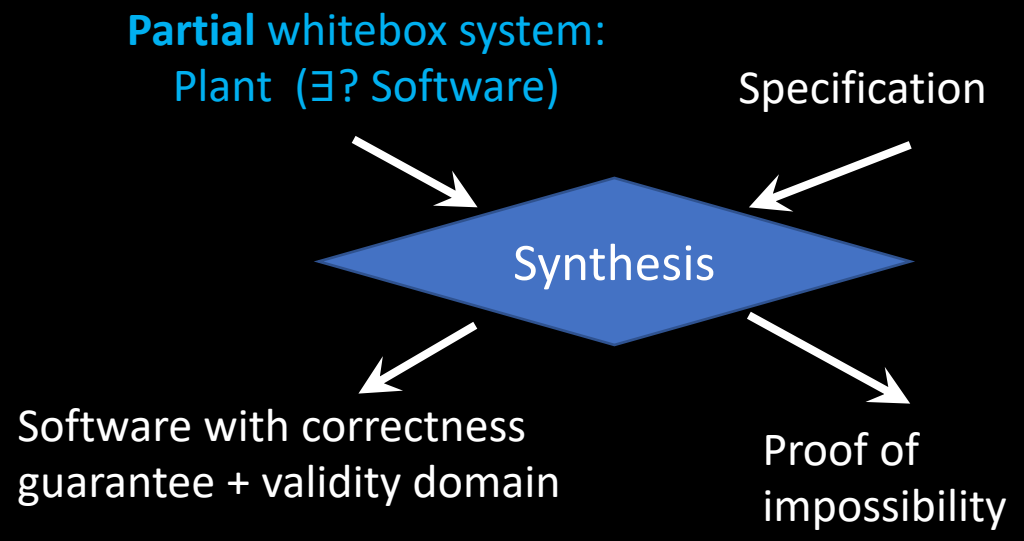


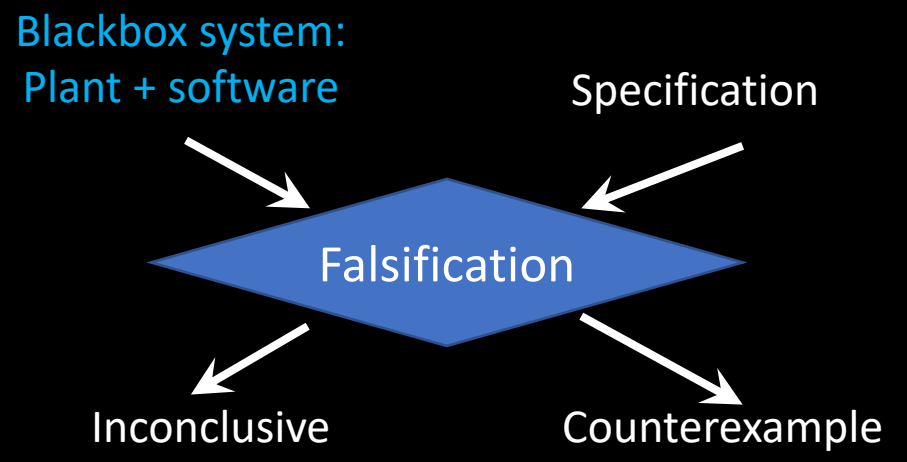
Key insights:

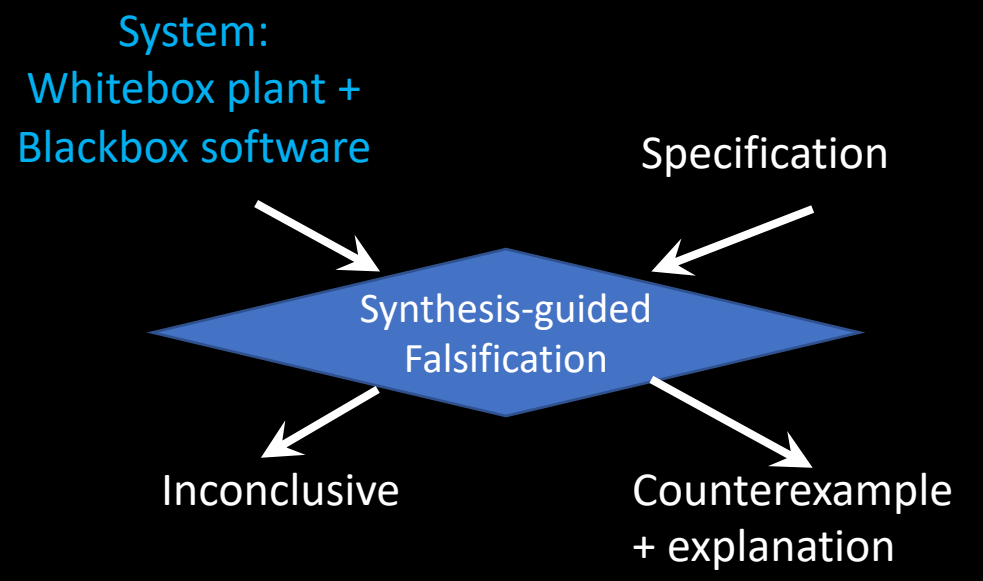
- Steering the search via specification-guided backward reachable sets of the plant model
- Querying the unknown controller only at specification-critical regions

Formal methods in CPS

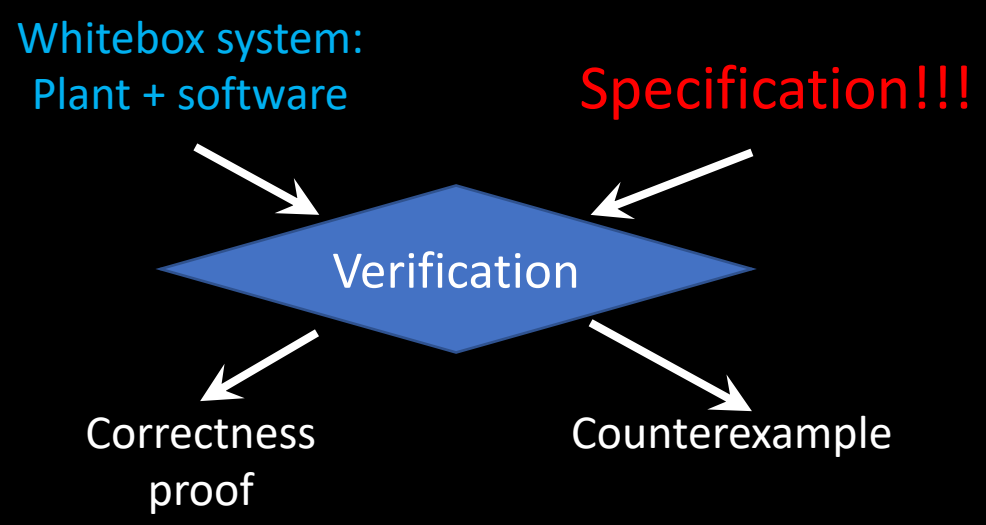


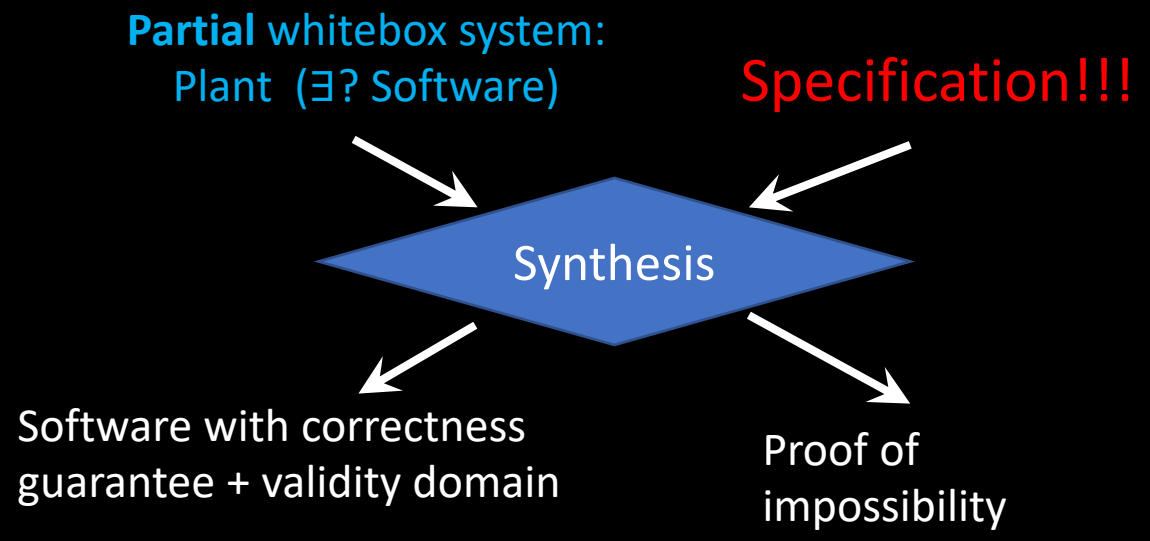


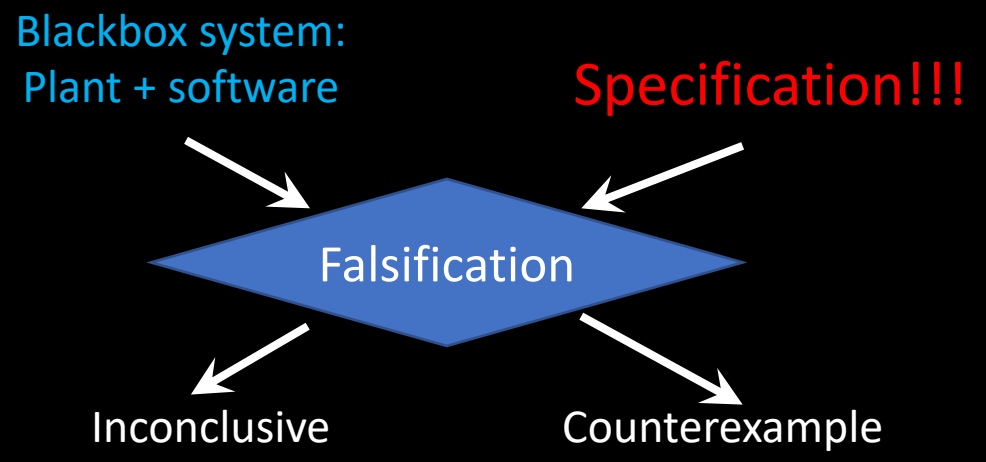


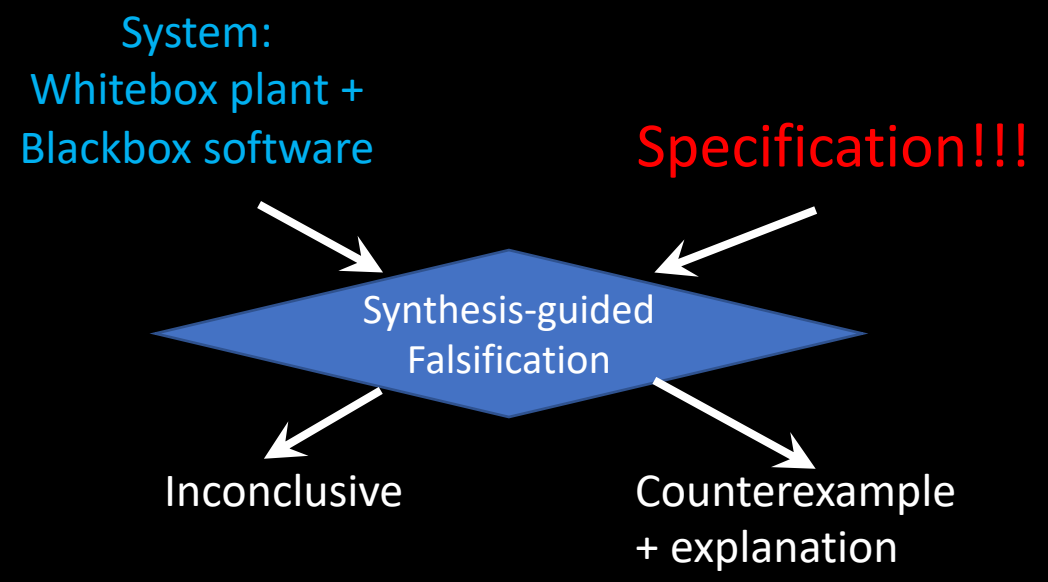


Formal methods in CPS



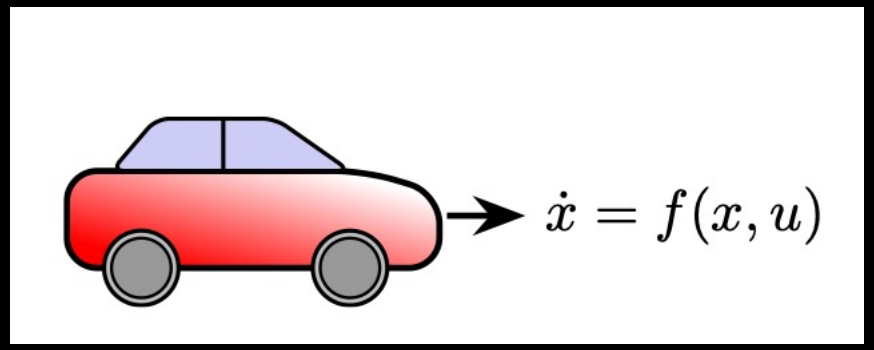


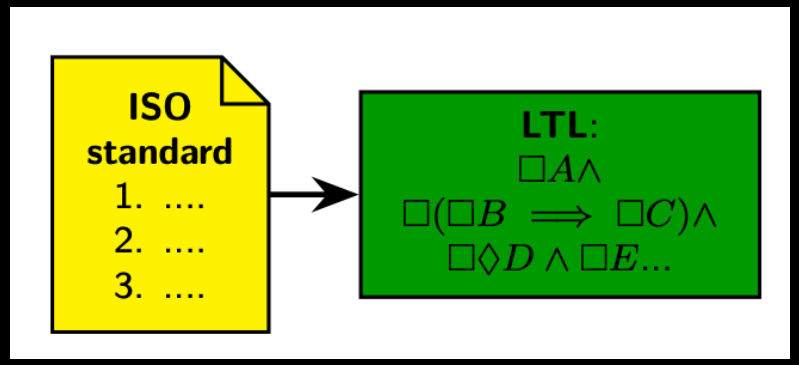




Back to: Adaptive Cruise Control

Mathematical model of the system





Formal specification

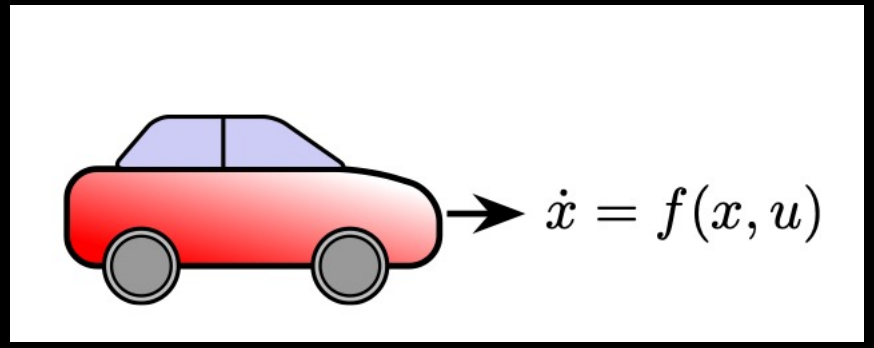
- A first attempt at specification:

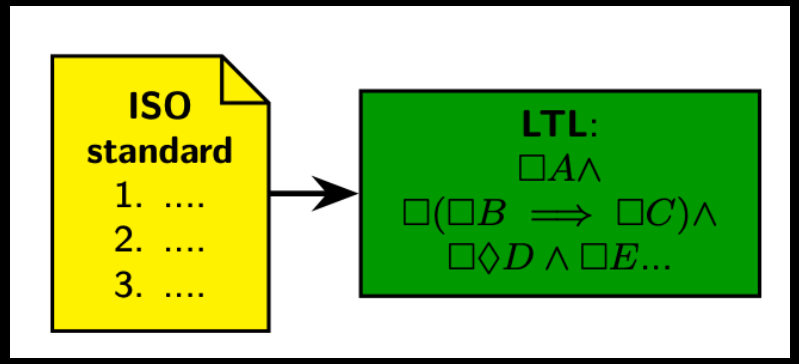
  "If there is no car in front, go at a set speed given by the user. If there is a car in front, follow the lead car."

- Would this work?

Back to: Adaptive Cruise Control

Mathematical model of the system





Formal specification

- Final specification after carefully studying ISO and SAE standards:

$$\psi_a = \Box(v_{1,\min} \leq v_1 \leq v_{1,\max}) \wedge \Box(a_{1,\min} \leq a_1 \leq a_{1,\max})$$

assumptions

$$\begin{aligned}
\psi_g = {} & \Box(a_{\min} \leq u \leq a_{\max}) \wedge \\
& \Box\,[(\Box(v_{\text{set}} \leq h/\tau_{\text{set}}) \rightarrow \Diamond\Box(|v - v_{\text{set}}| \leq \varepsilon)) \wedge \\
& (\Box(v_{\text{set}} > h/\tau_{\text{set}}) \rightarrow \Diamond\Box(|v - h/\tau_{\text{set}}| \leq \varepsilon)) \wedge \\
& ((v_{\text{set}} \leq h/\tau_{\text{set}}) \rightarrow ((0 \leq v \leq v_{\max}) \wedge (0 \leq h))) \wedge \\
& ((v_{\text{set}} > h/\tau_{\text{set}}) \rightarrow ((v \leq h/\tau_{\min}) \wedge (0 \leq v \leq v_{\max}) \wedge (0 \leq h)))]\,.
\end{aligned}$$

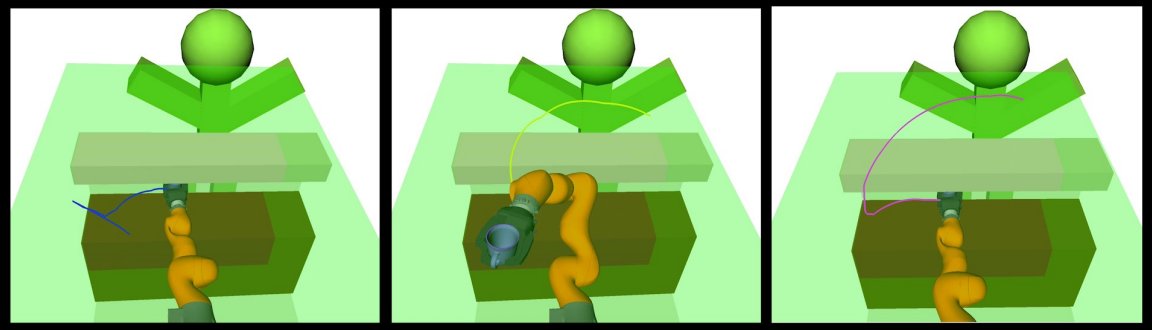
guarantees

$$\psi_a \rightarrow \psi_g$$

- Is there an easier way of coming up with specifications?

Data → Models + *Specification* → Control

Task specifications useful for (i) control design
(ii) system monitoring, (iii) anomaly detection, etc.



Teaching a robot to do a task

"Small" data

Learning from demonstrations → constraint/task learning

↓

Generalization

(performing the tasks in new environments)

Learning from demonstration (LfD)

- Specifying tasks by hand can be challenging.
- Goal: Given demonstrations of a task, learn a policy which completes the task in similar scenarios.
- Common approach: inverse optimal control (IOC) [Kalman '64, Ng et al. '00].
  - Assume demonstrator solves an *unconstrained* optimization problem.
  - Infer the cost function.
  - Plan using the learned cost function.

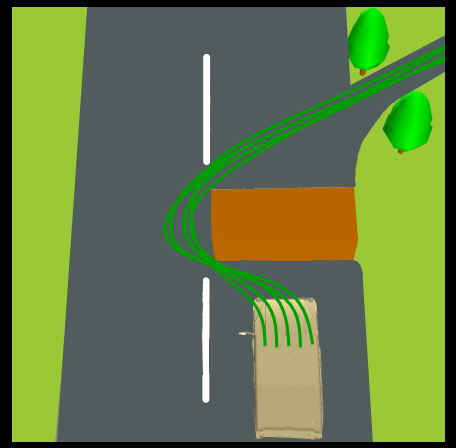
$$\underset{\xi}{\text{minimize}} \quad c(\xi) \approx \gamma^\top \phi(\xi)$$

demonstrated trajectory — cost function — learned parameters — known features

What if there are unknown constraints?

- IOC replaces the hard constraints with a soft cost *penalty*
  - May not satisfy underlying constraint when generalizing
  - Difficult to write down interpretable cost function features
  - Very difficult to scale to multiple subtasks
- Our approach: directly learn the hard constraints

Task: Avoid obstacle



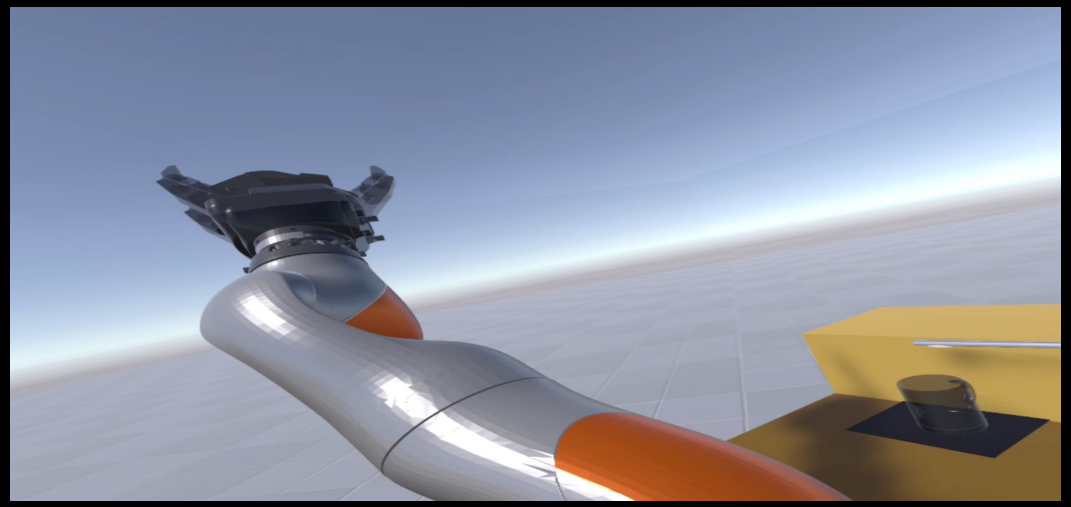
Task: Fill the cup

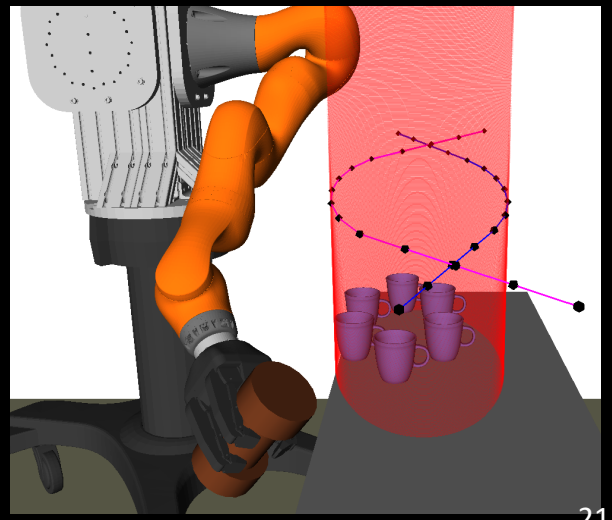
↓

Grasp the cup

↓

Deliver it without spills





Problem statement

$\xi = (\xi_x = [x_1, x_2, \ldots], \xi_u = [u_1, u_2, \ldots])$

Demonstrator's problem:

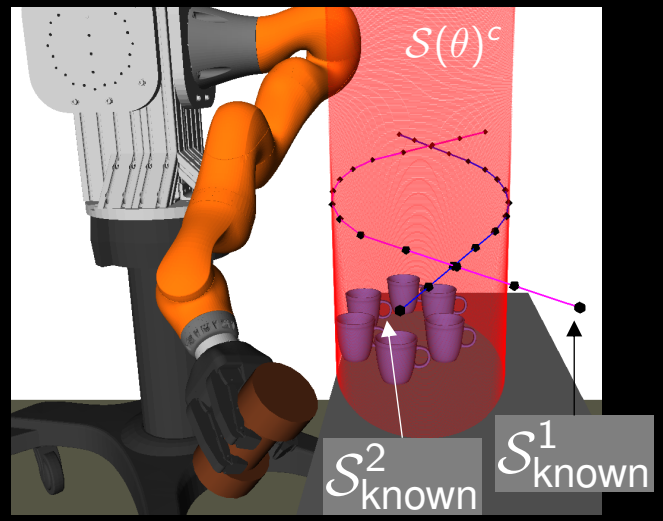
$$\underset{\xi}{\text{minimize}} \quad c(\xi)$$

$$\text{subject to} \quad \xi \in \mathcal{S}(\theta)$$

Safe set defined by θ: *unknown* to the learner

$$\xi \in \mathcal{S}^i_{\text{known}}$$

Known constraints, e.g. dynamics, start/goal state



Given: N optimal demos $\xi_1, \ldots, \xi_N$ (and their known constraints)

Learner's problem

$$\text{find} \quad \theta$$

$$\text{subject to} \quad \xi_1, \ldots, \xi_N \text{ optimal}$$

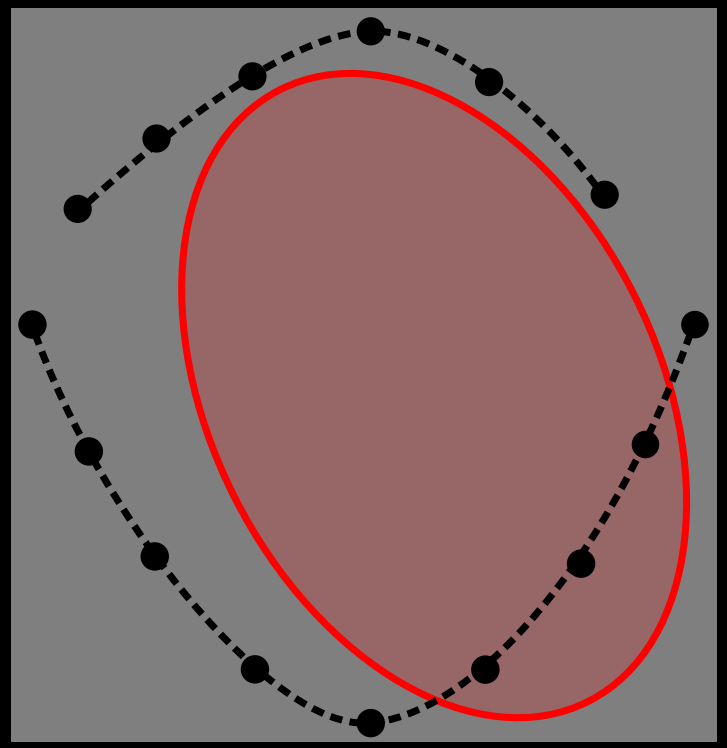
Core challenge:
Need to distinguish between valid and invalid behavior from only valid examples

How? Demonstrator optimality.

Key insight: Find constraints that make each demonstration satisfy its Karush-Kuhn-Tucker (KKT) optimality conditions.

KKT conditions for demonstration *i*, KKT(ξ_i, θ):

Primal feasibility	*Find a parameter which makes the demos safe*
Stationarity	*Local changes to the demo either increase the cost or cause infeasibility*
Dual feasibility + complementary slackness	*Auxiliary conditions needed to show stationarity*



Can dramatically reduce the set of valid constraints!

Learner's problem: Inverse-KKT

$$\begin{aligned} \text{find} \quad & \theta \\ \text{subject to} \quad & \text{KKT}(\xi_i, \theta), \quad i = 1, \dots N \end{aligned}$$

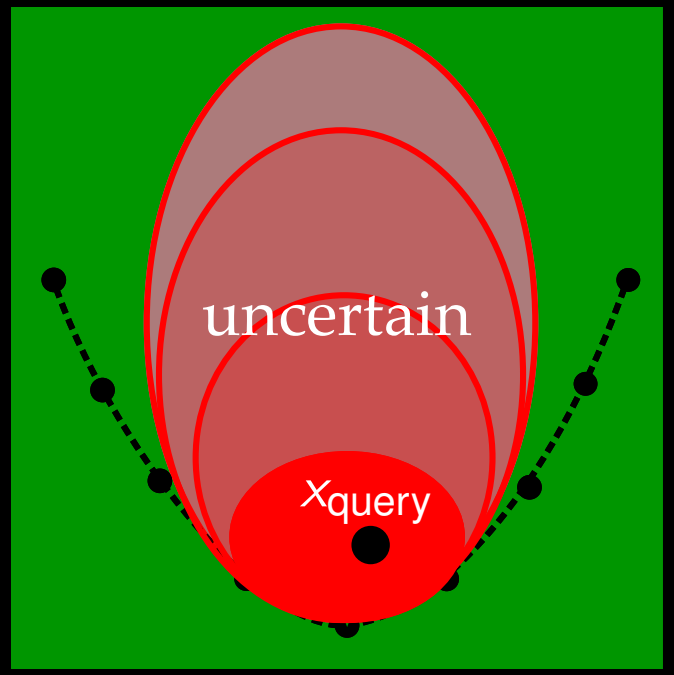
Mixed integer linear program (MILP)-representable for some constraint parameterizations (integer due to complementary slackness)

- Ill-posed: many parameters θ can be consistent with Inverse-KKT
  - But parameters may all agree on safety for some states
  - Simple way to check:

$$\begin{aligned} \text{find} \quad & \theta \\ \text{subject to} \quad & \text{KKT}(\xi_i, \theta), \quad i = 1, \dots N \\ & \boxed{x_{\text{query}} \text{ unsafe}} \end{aligned}$$

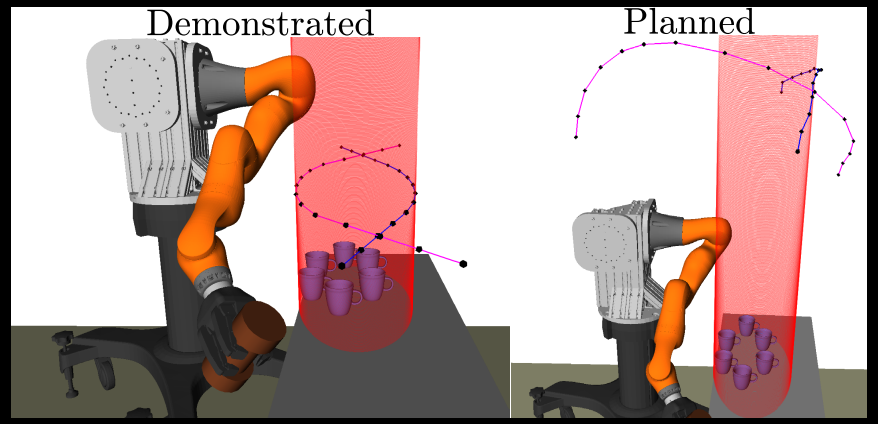
If infeasible, then x_{query} must be safe (if the parameterization is correct).

Can also do more sophisticated checks (reason about uncertainty directly).

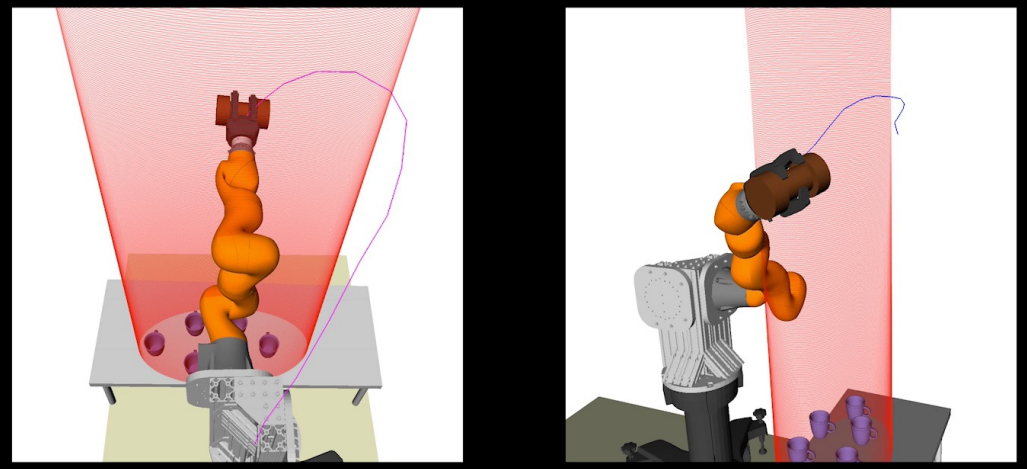


Results

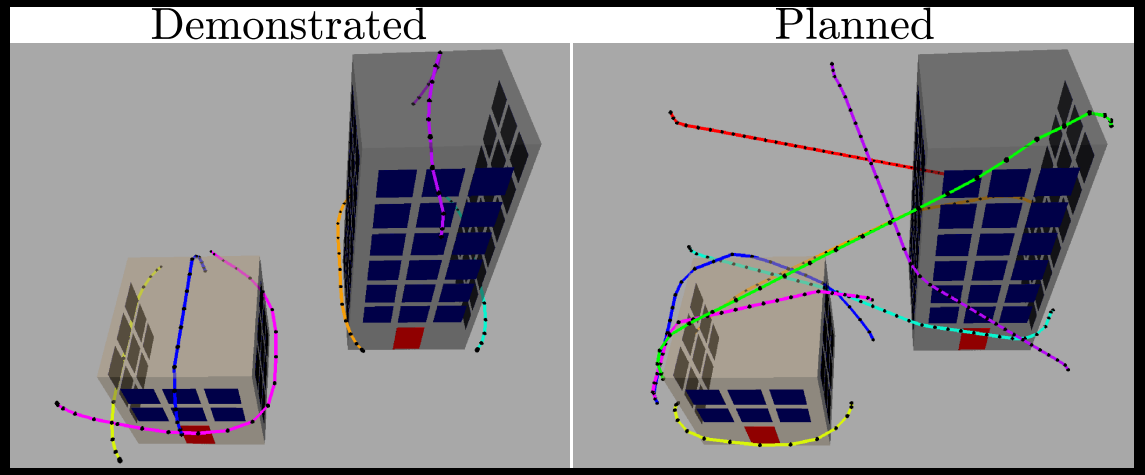
7DOF arm



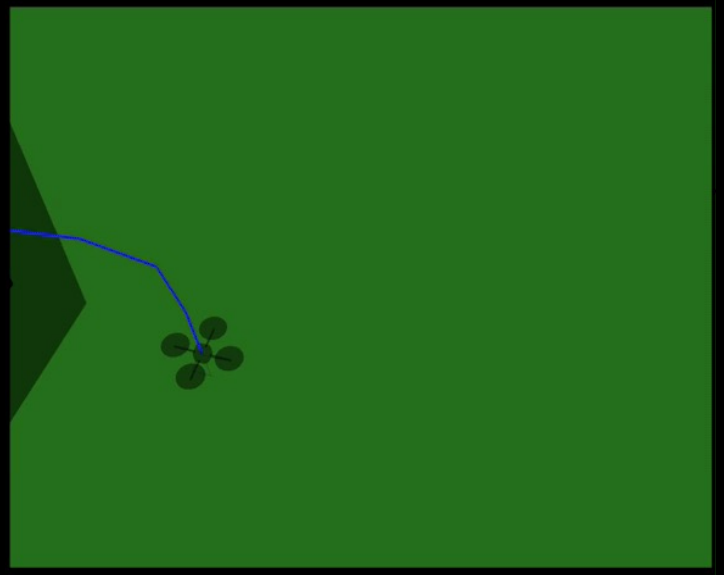
7DOF arm (animated plan)

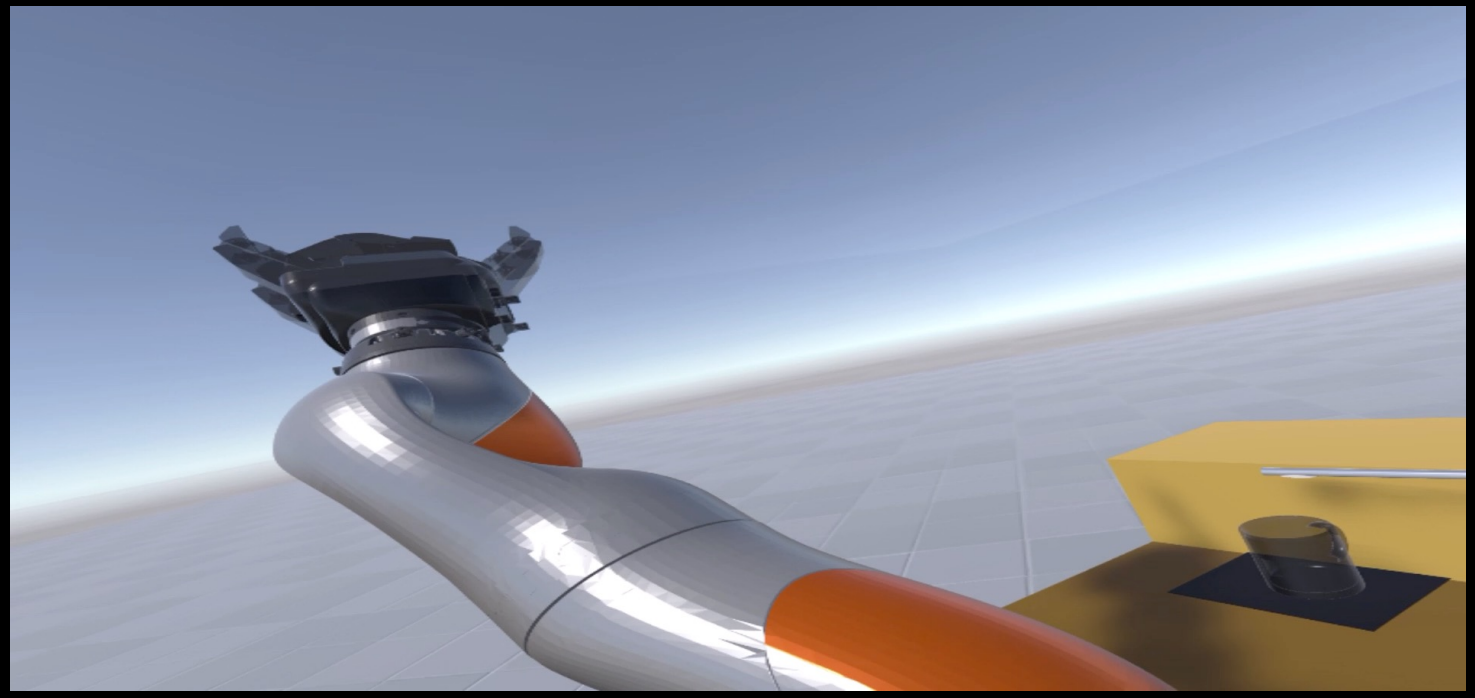


Quadrotor with 12D dynamics



Quadrotor (animated plan)





Multi-stage task:

Fill glass $\rightarrow$ Grasp glass $\rightarrow$ Deliver

Challenges:

- Hard time-varying constraints must be satisfied to complete the task
- Only given a small number of positive demonstrations

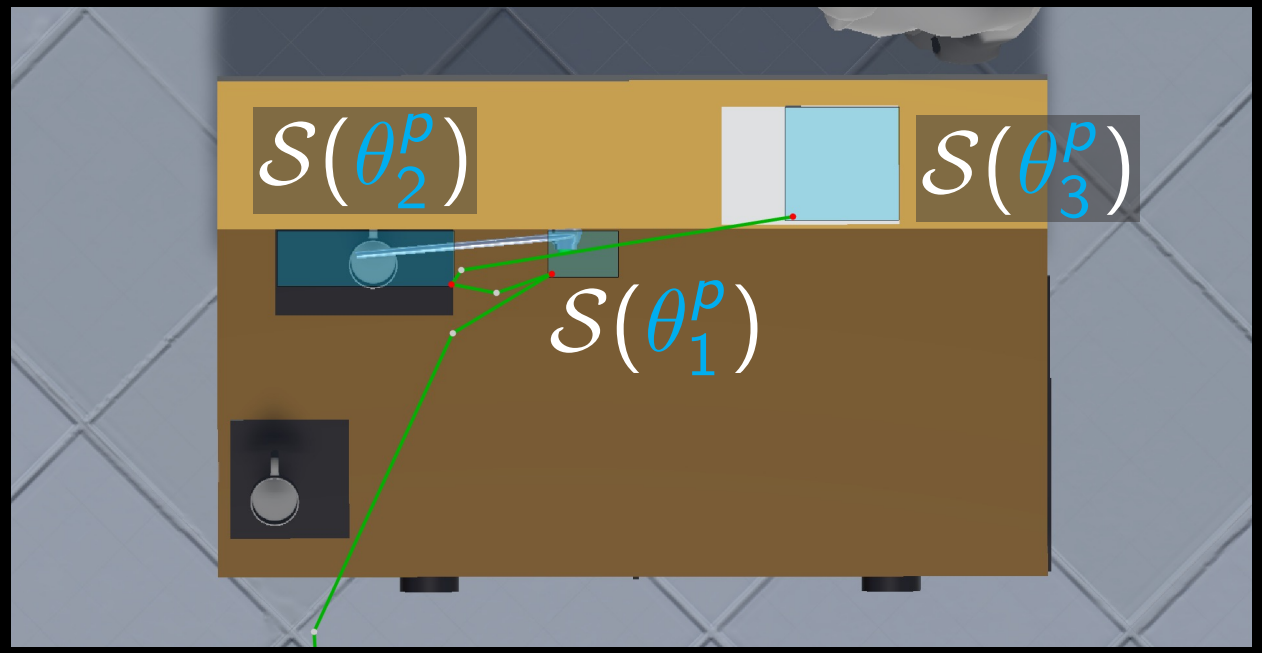
Question:

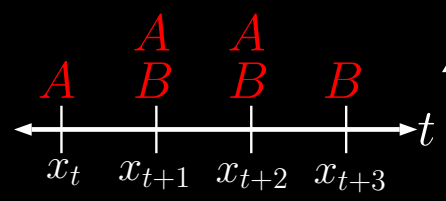
Can we learn multi-stage, constrained tasks from only safe, locally-optimal demonstrations, and use them to complete tasks in new environments while guaranteeing safety?

Multi-stage constrained tasks

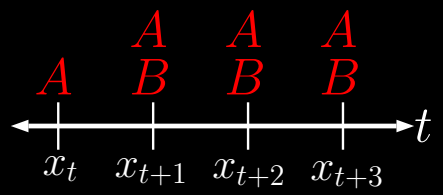
- Need: hybrid continuous-discrete constraints on trajectory over time
- *Use linear temporal logic (LTL):* extends propositional logic to trajectories
  - Until: $A\,\mathcal{U}\,B$: prop. A must hold until prop. B holds for the first time
  - Always: $\mathbf{G}A$: prop. A must always hold (**G**lobally)
  - Eventually: $\mathbf{F}C$: prop. C must hold at some time (**F**inally)
- Connect to the continuous world via the atomic propositions (APs)

  at time t, $p_i(\theta^p) =$ True $\Leftrightarrow x_t \in \mathcal{S}(\theta_i^p)$

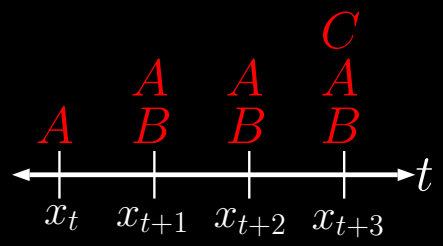




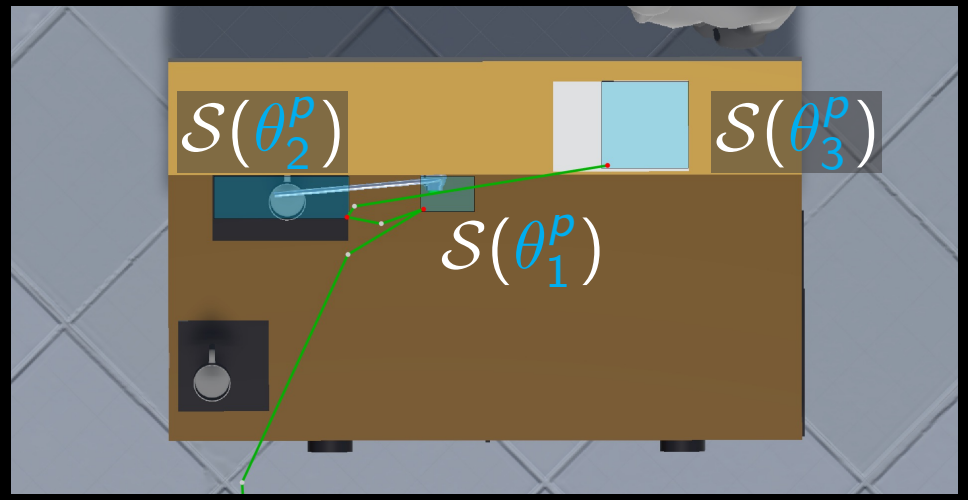
$A\,\mathcal{U}\,B$



$\mathbf{G}A$



$\mathbf{F}C$



θ^p : Captured by the **geometry**

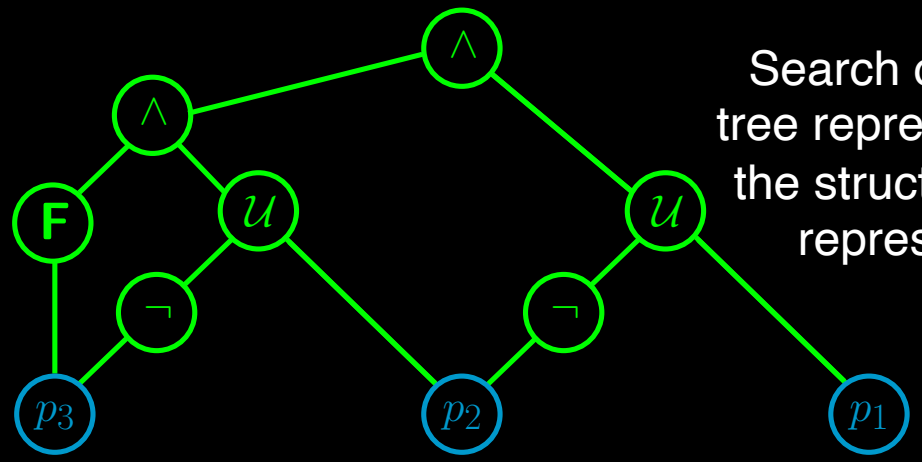
θ^s : Captured by the **grammar**

<u>Task:</u> Fill glass → Grasp glass → Deliver

$$\varphi(\theta^s, \theta^p) = (\neg p_3(\theta^p)\ \mathcal{U}\ p_2(\theta^p)) \wedge (\neg p_2(\theta^p)\ \mathcal{U}\ p_1(\theta^p)) \wedge (\mathbf{F} p_3(\theta^p))$$

Don't visit p_3 before p_2 — Don't visit p_2 before p_1 — Eventually visit p_3

$$\mathcal{S}(\theta) = \{\xi \mid \xi \text{ satisfies } \varphi(\theta^s, \theta^p)\}$$



Search over parse tree representation of the structure; MILP-representable

Goal: learn θ^s, θ^p

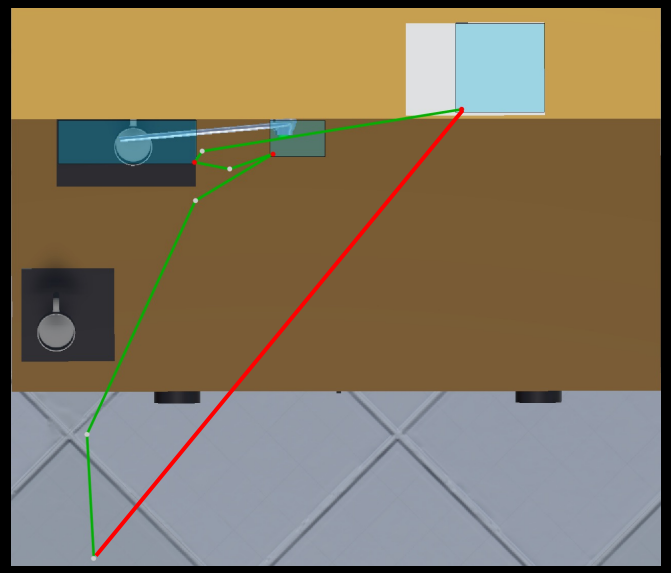
Learning θ^s, θ^p

Core ideas:

Enforce the demonstrations satisfy their KKT conditions (*continuous* optimality conditions) to learn θ^p.

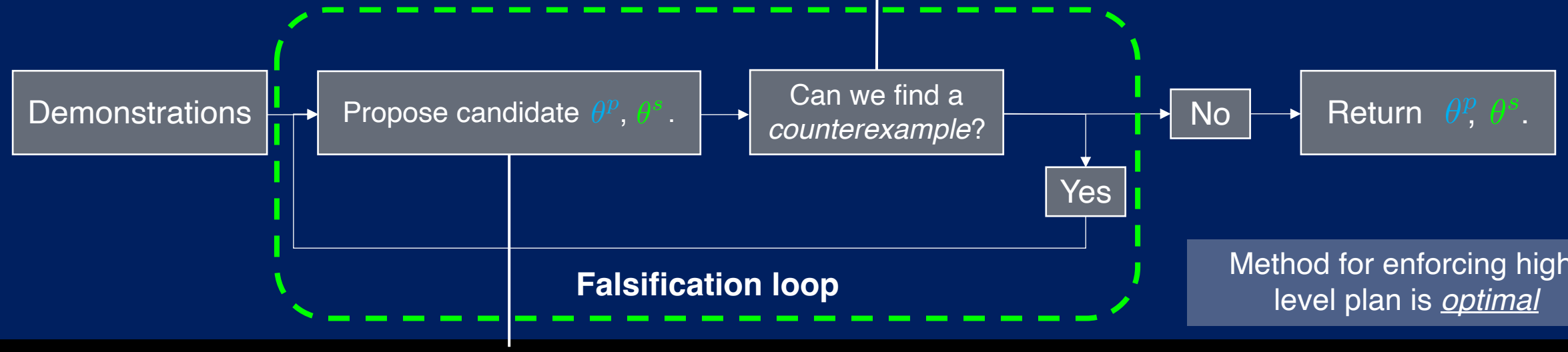
Enforce the demonstrations don't perform unnecessary subtasks (i.e. APs) for the learned θ^s (enforces *discrete* optimality).

- Why is discrete optimality important?
  - Demonstrations are also feasible for far weaker formula structures
- Harder to enforce. Why?
  - Boils down to exhaustive enumeration of lower-cost solutions.



Learning θ^s, θ^p

Counterexample:
a trajectory satisfying $\varphi(\theta^s, \theta^p)$ at lower cost than the demonstration



Method for enforcing high-level plan is *optimal*

Learner's problem:

$$\begin{aligned} \text{find} \quad & \theta^s, \theta^p \\ \text{subject to} \quad & \mathrm{KKT}(\xi_i, \theta^p), \quad i = 1, \dots N \\ & \text{counterexamples violate } \varphi(\theta^s, \theta^p) \end{aligned}$$

Theoretical guarantees:

- *Consistent*: makes the demonstrations optimal
- *Probabilistically complete*: returns consistent formula if one exists
- *Simplest*: returns shortest consistent formula
- *Conservative*: obtain safe inner-approximations of the AP constraints

7-DOF arm bartender task

Task:

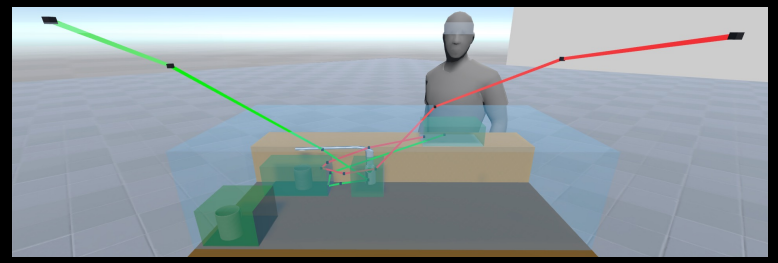
Fill glass → Grasp glass → Deliver

Learned:

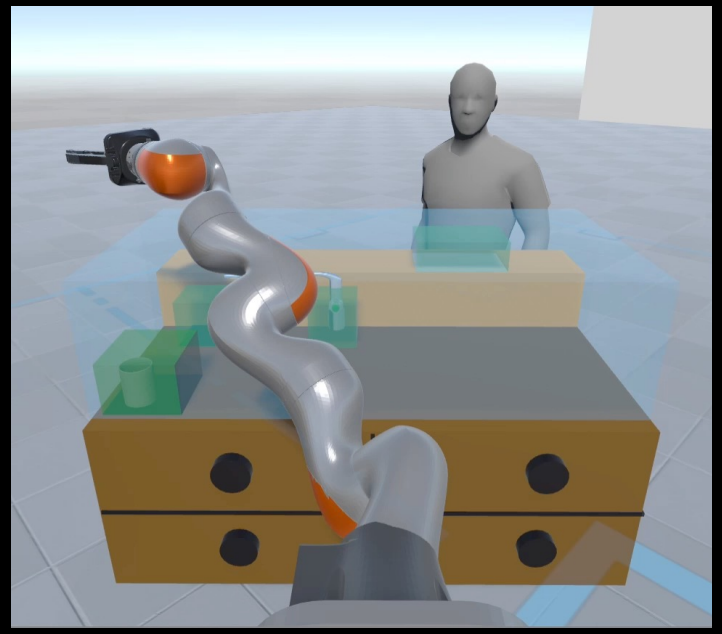
AP parameters (location of relevant objects)

LTL formula structure (task sequence)

$$\varphi(\theta^s, \theta^p) = (\neg p_3(\theta^p)\ \mathcal{U}\ p_2(\theta^p)) \wedge (\neg p_2(\theta^p)\ \mathcal{U}\ p_1(\theta^p)) \wedge (\mathbf{F} p_3(\theta^p))$$



Demos



Planned trajectory

Quadrotor surveillance

Task:

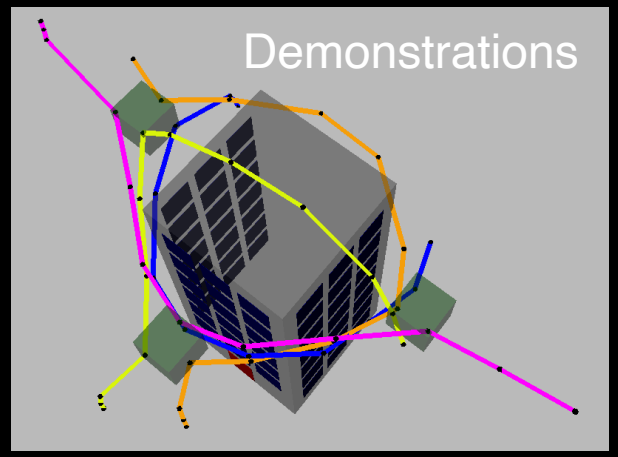
visit green regions without colliding with the building

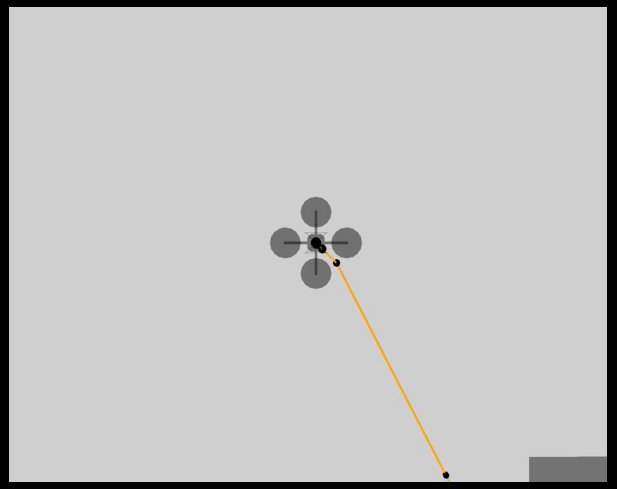
Learned:

AP parameters (locations of interest)

LTL formula structure:

$$\varphi(\theta^s, \theta^p) = \mathbf{F} p_1(\theta^p) \wedge \mathbf{F} p_2(\theta^p) \wedge \mathbf{F} p_3(\theta^p)$$

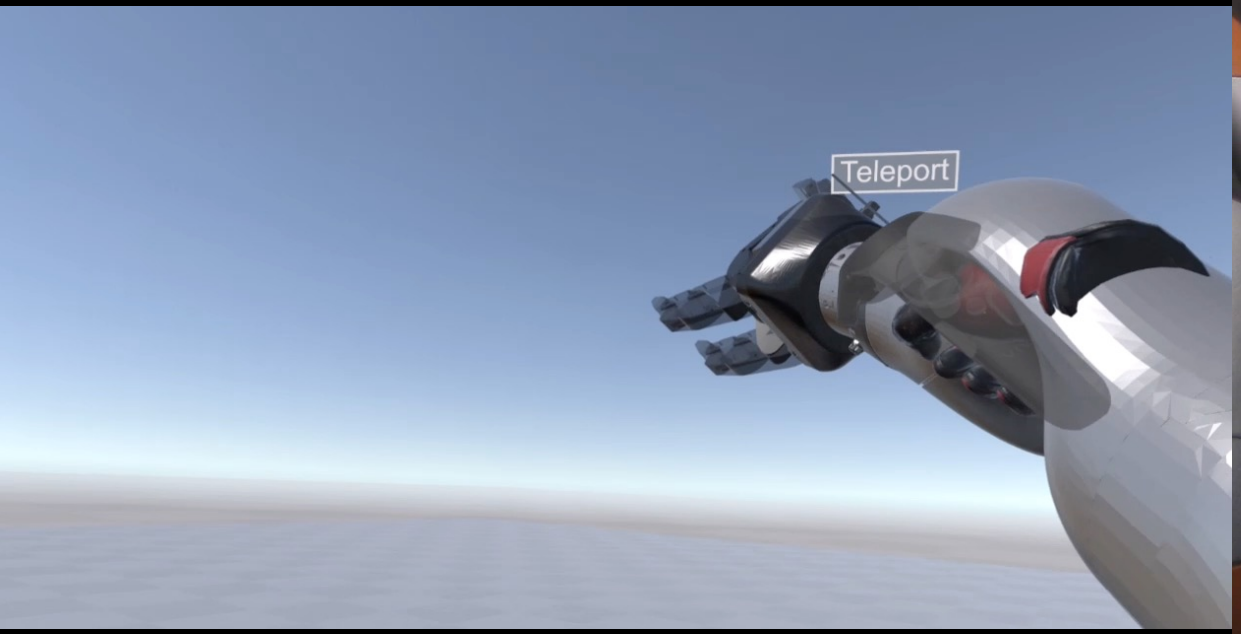




Planned trajectory

7-DOF multi-stage package delivery task

Demonstration (in VR)



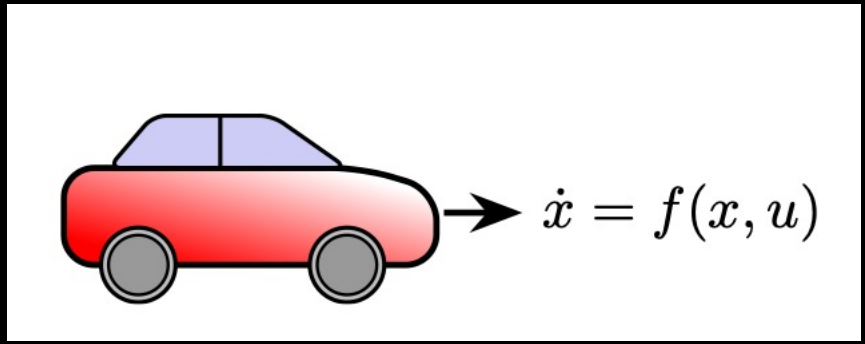
Execution in new scene

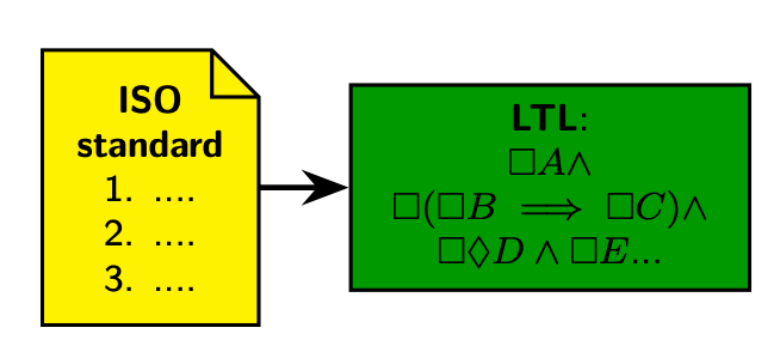
Task:

grasp soup $\rightarrow$ place in small box $\rightarrow$ deliver to blue region $\rightarrow$ grasp Cheez-Its $\rightarrow$ place in large box

Learned: LTL formula structure, APs sensed $\rightarrow$ *plan with learned LTL structure in entirely new environment*

Mathematical model of the system





Formal specification

- Synthesis or formal verification require models and specifications:
  - both modeling and specifying requirements can be hard, learning can help

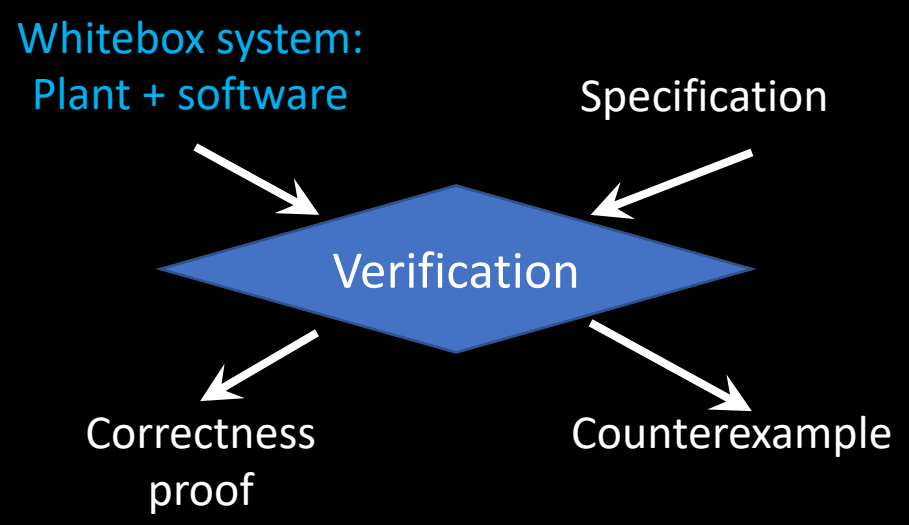
Key message for learning specifications:

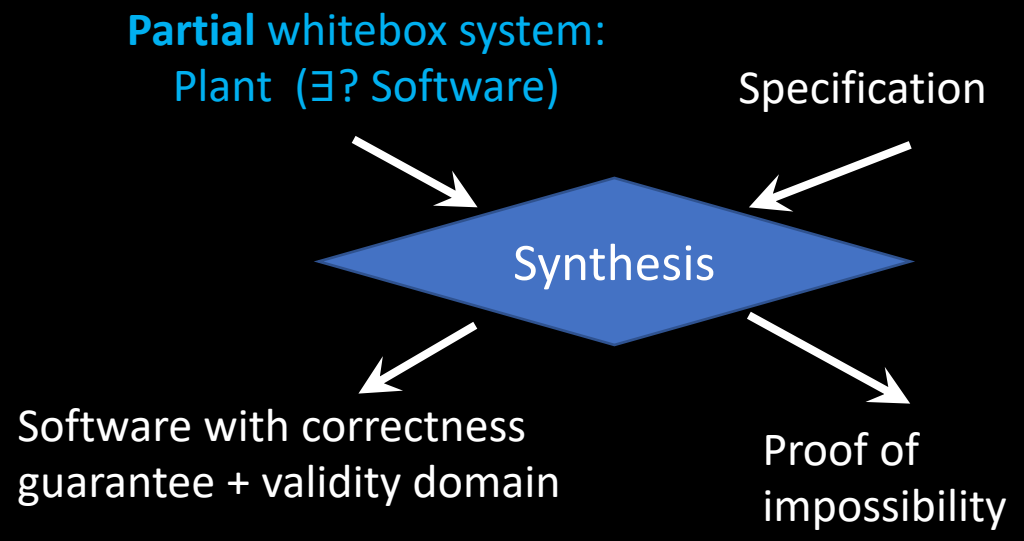
- Optimality (or being close to optimal) is a very strong prior that can be utilized when learning from limited data

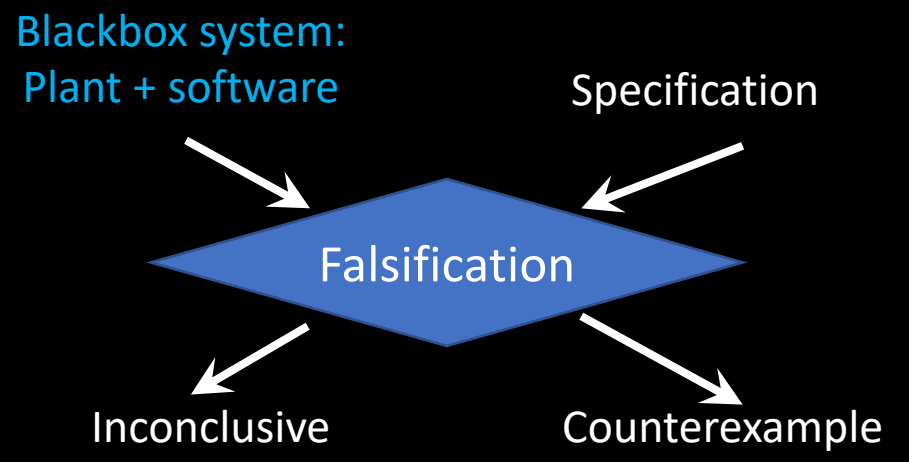
Current work:

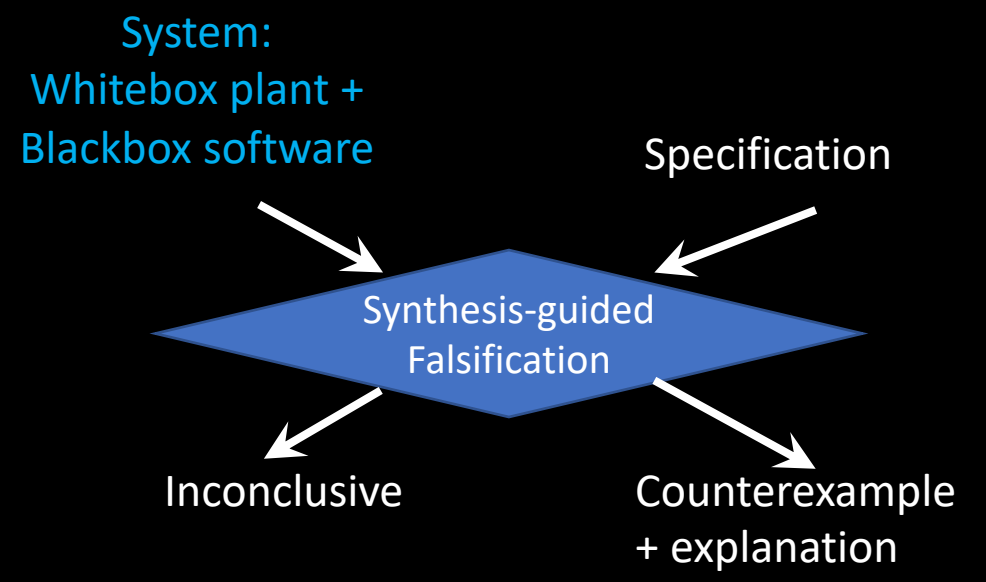
- Learning probabilistic constraints for safety of very large scale complex systems (in the poster session)

Formal methods in CPS



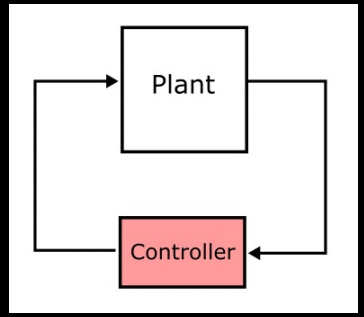






Conclusions and future challenges

- We have a good repertoire of formal techniques for specifying, designing, and analyzing cyber-physical systems in a way to increase *trustworthiness.*
- Challenges:
  - Scalability
    - more complex, high-dimensional, nonlinear models with learning-based components
    - more complex specifications
  - Learning: how to trust the learned artifacts or datasets?
  - Formal methods for humans CPS
    - both modeling and specifying is much harder
  - Formal methods for multi-agent CPS (with multiple decision-makers, including humans)
    - How to scale?
    - How to safely evaluate?
    - How to analyze?

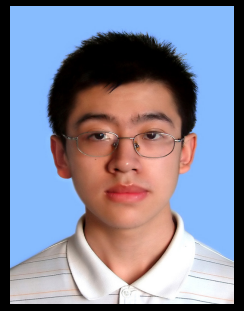


ACKNOWLEDGMENTS


Glen Chou


Zexiang Liu


Petter Nilsson

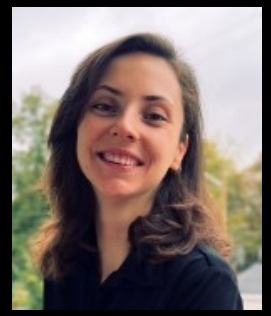

Liren Yang

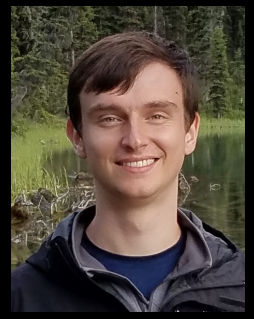

Kwesi Rutledge


Yuxiao Chen


Yunus Sahin


Sunho Jang


Ruya Karagulle


Stan Smith

And other team members…

Other CPS collaborators: Dmitry Berenson, Johanna Mathieu, Stephane Lafortune, Dimitra Panagou, Sze Zheng Yong, Samet Oymak, Jessy Grizzle, Huei Peng, Paulo Tabuada, Aaron Ames, Mario Sznaier

Funding Agencies: NSF CPS Program, ONR, Toyota Research Institute, Ford Research, Collins Aerospace

Special thanks to late Kishan Baheti, who was the PM for my NSF CAREER Award (joint between CPS and EPCN)