PhysCov: Physical Test Coverage for Autonomous Vehicles

Carl Hildebrandt, Meriel von Stein, and Sebastian Elbaum

hildebrandt.carl@virginia.edu

Motivation

Autonomous Systems are here

Waymo [1]

Oxa [2]

[1] https://waymo.com/
[2] https://oxa.tech/

Motivation

Soon they will be common place

Waymo [1]

Oxa [2]

AutoX [3]

Cruise [4]

May Mobility [5]

Motional [6]

Pony AI [7]

Zoom [8]

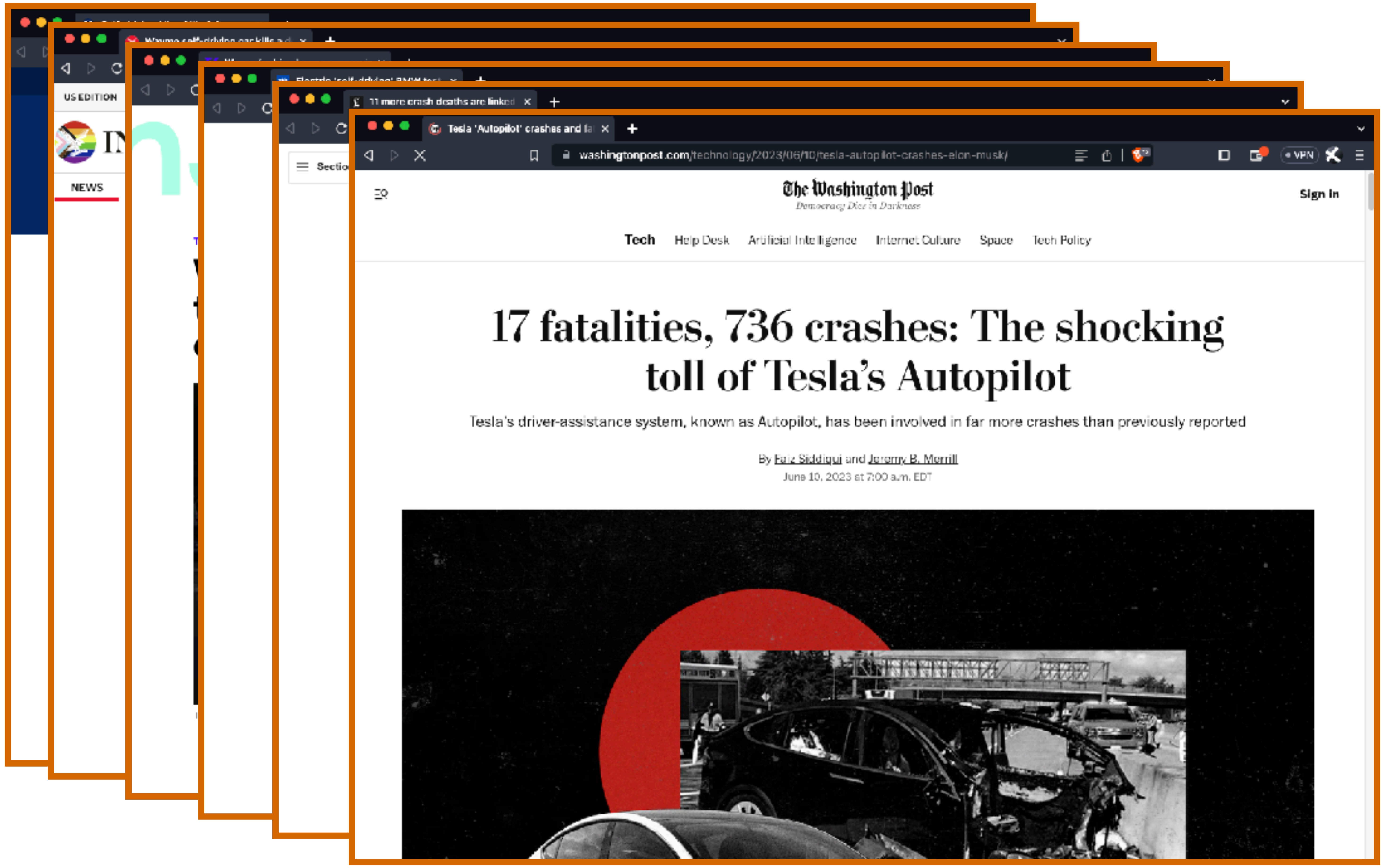
Tesla [9]

• • •

[1] https://waymo.com/
[2] https://oxa.tech/
[3] https://www.autox.ai
[4] https://getcruise.com
[5] https://maymobility.com
[6] https://motional.com
[7] https://www.pony.ai
[8] https://zoox.com/
[9] https://www.tesla.com/autopilot

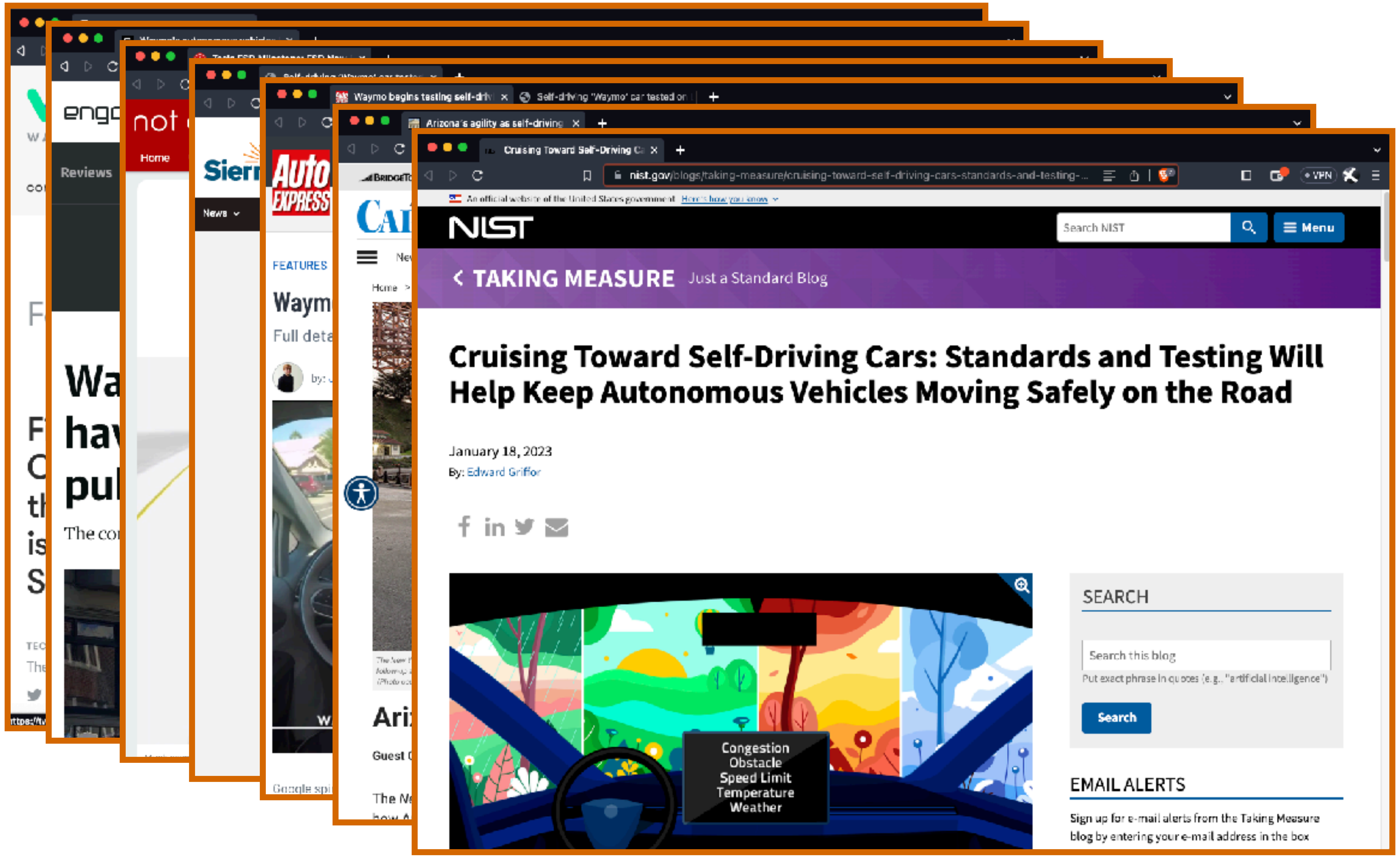
Motivation

These vehicles fail, resulting in the loss of life



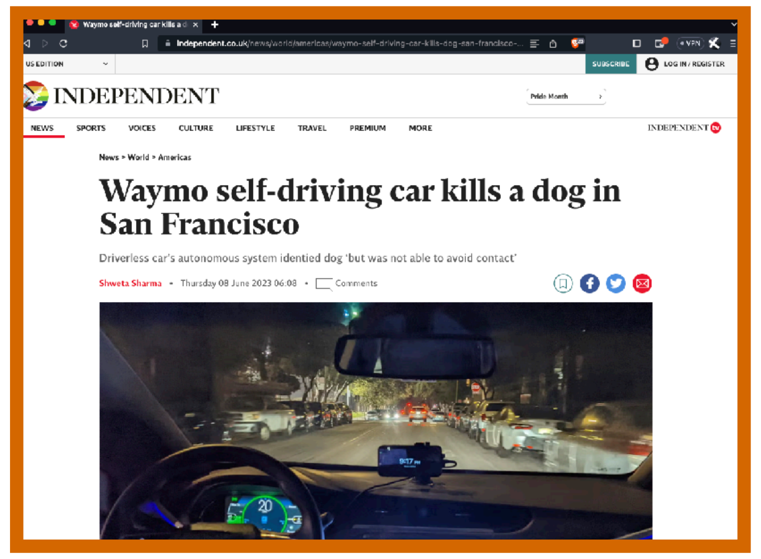
Motivation

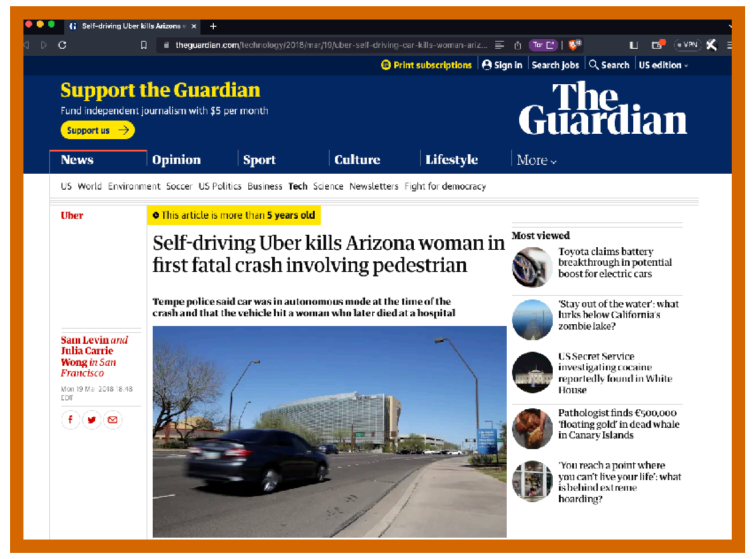
They are being tested

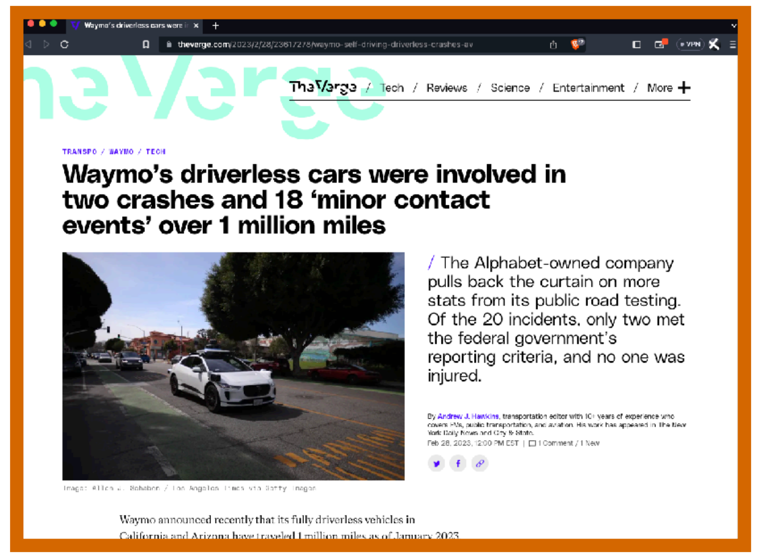


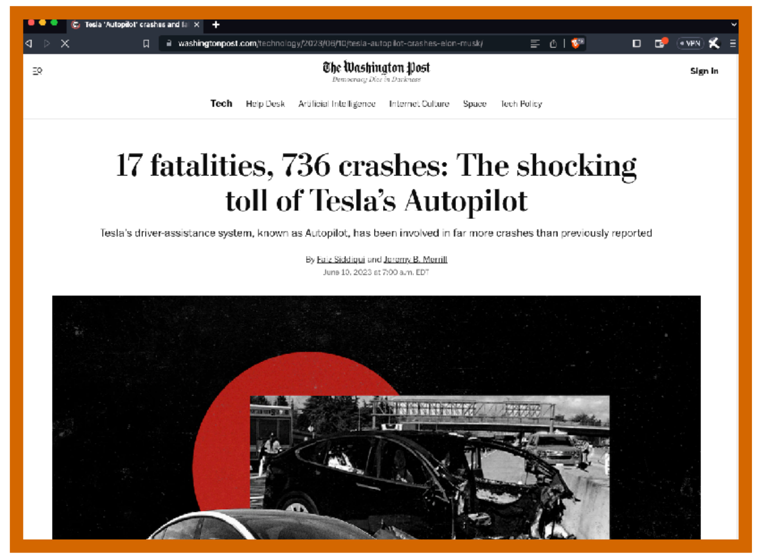
Motivation

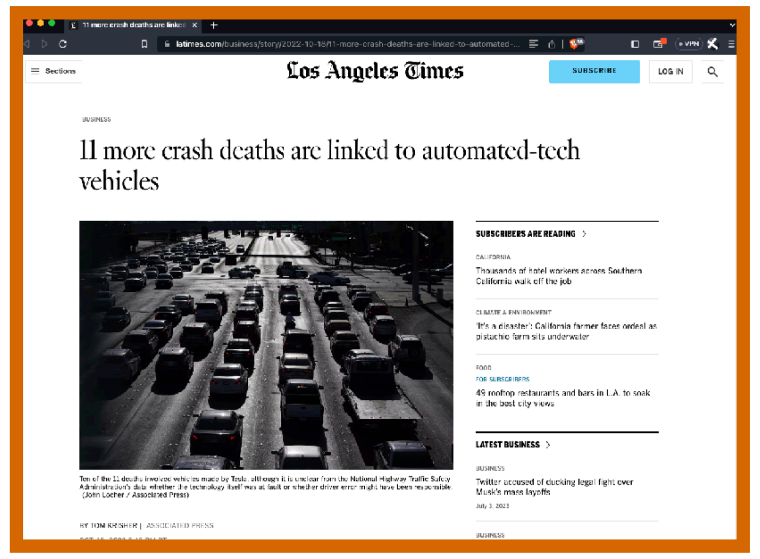
Failures





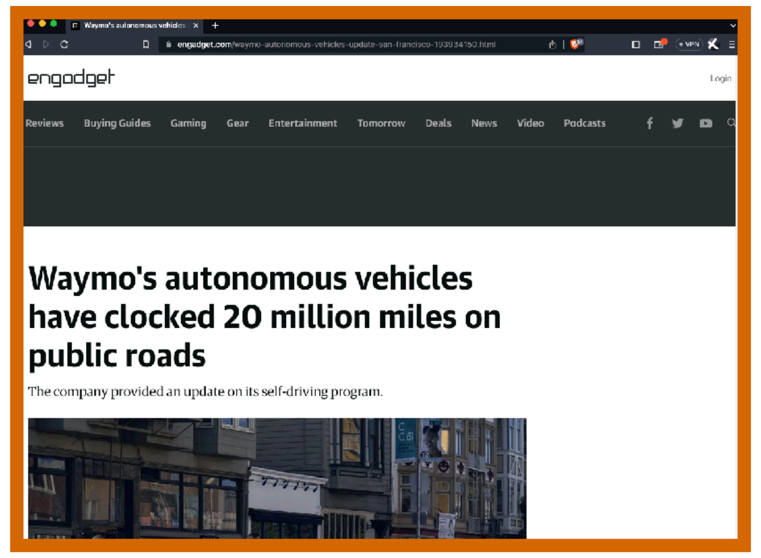


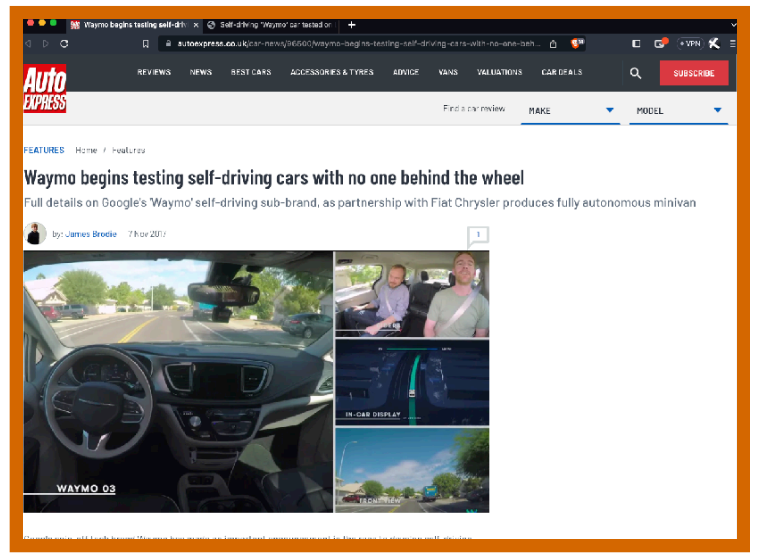


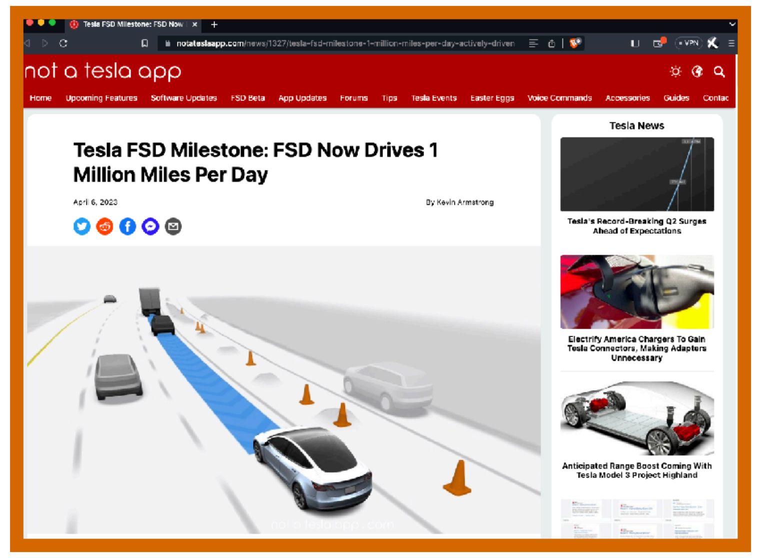


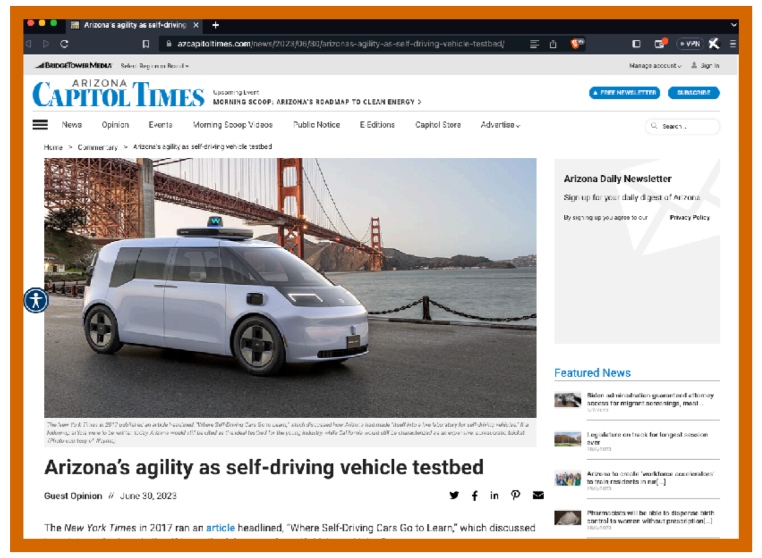
Where is the disconnect?

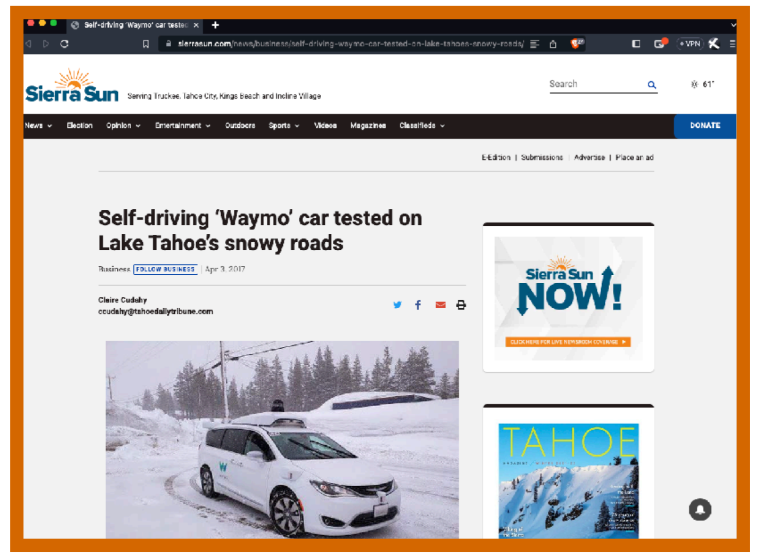
Testing

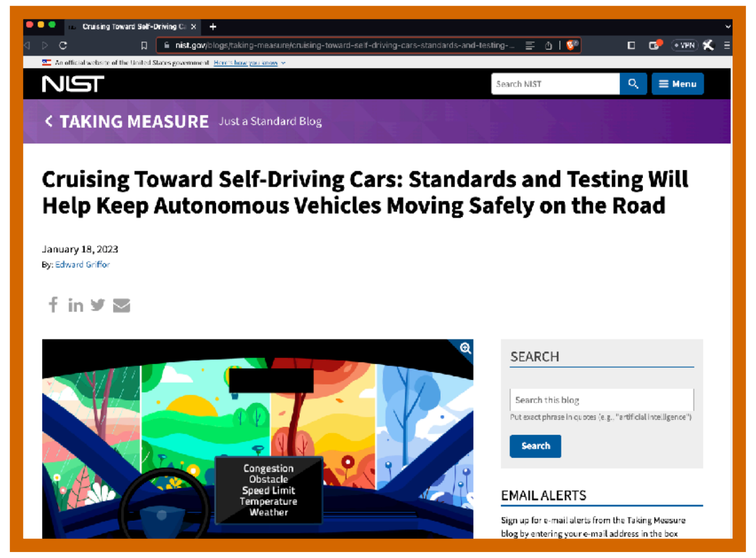












Disconnect

“Was the previous test useful?”

“How thoroughly is the current system tested?”

“When is it safe to stop testing?”

Disconnect

"How do we quantify an autonomous vehicle's test adequacy?"

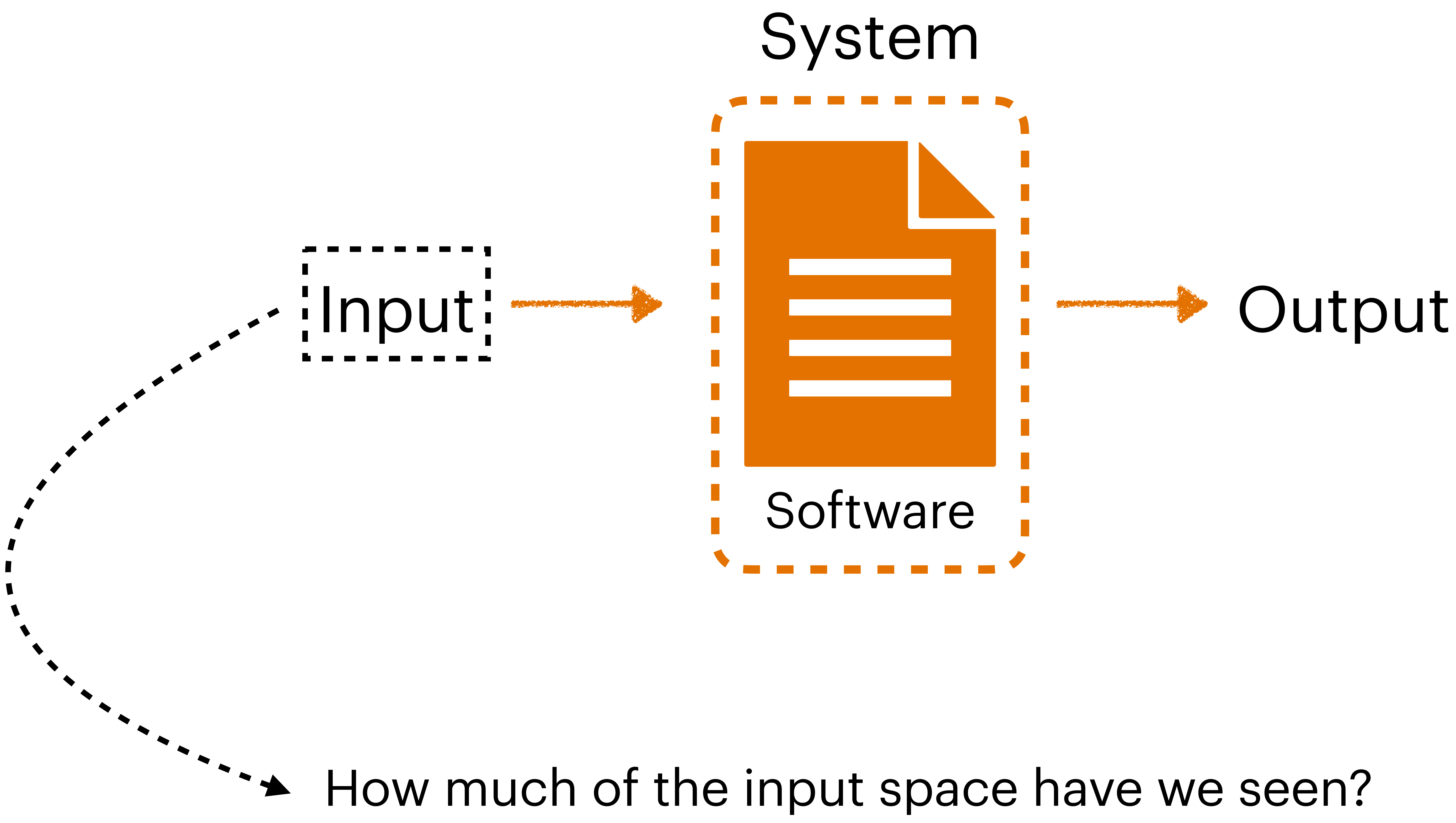
Problem

Traditional software uses test adequacy metrics

Input → System → Output

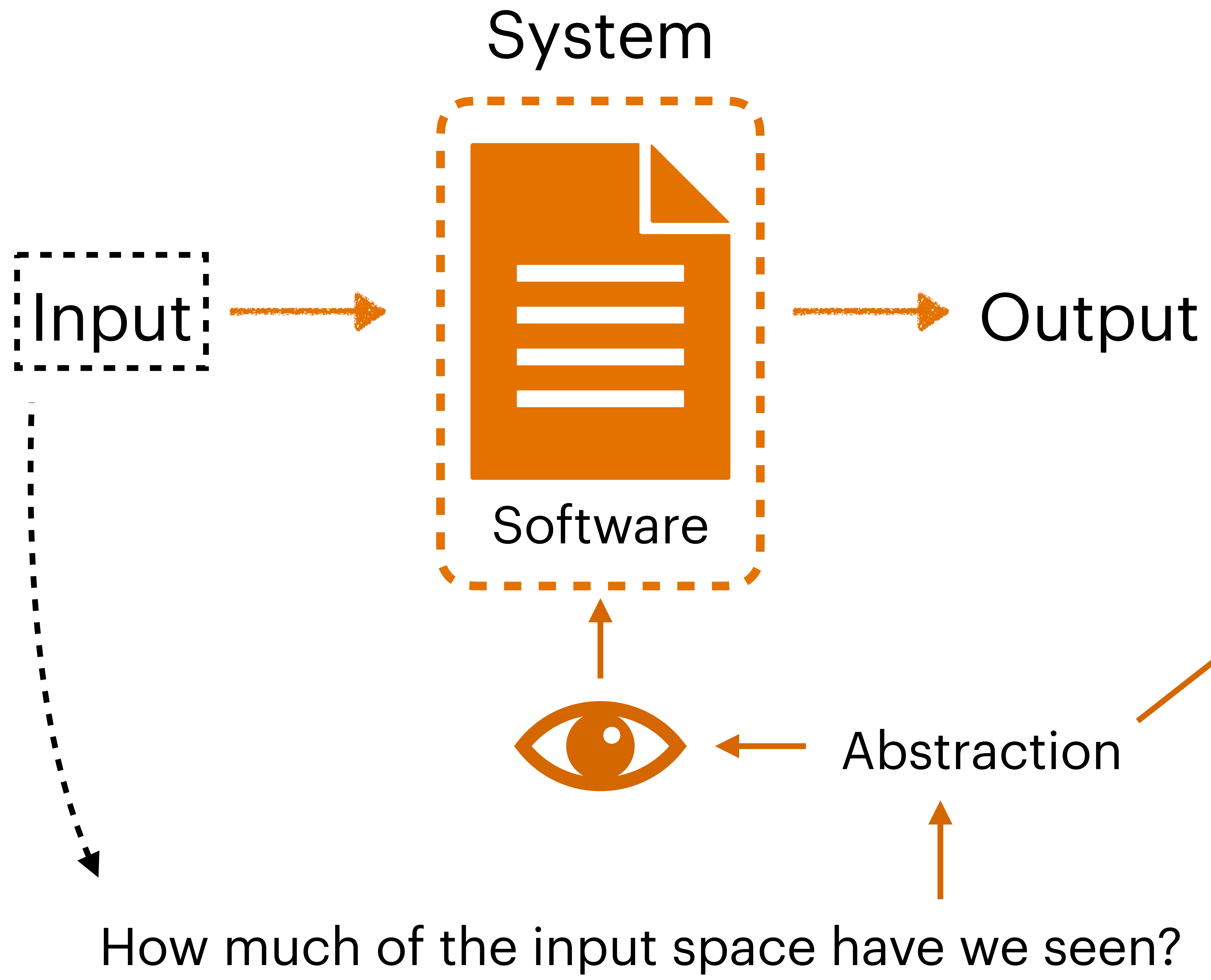
Problem

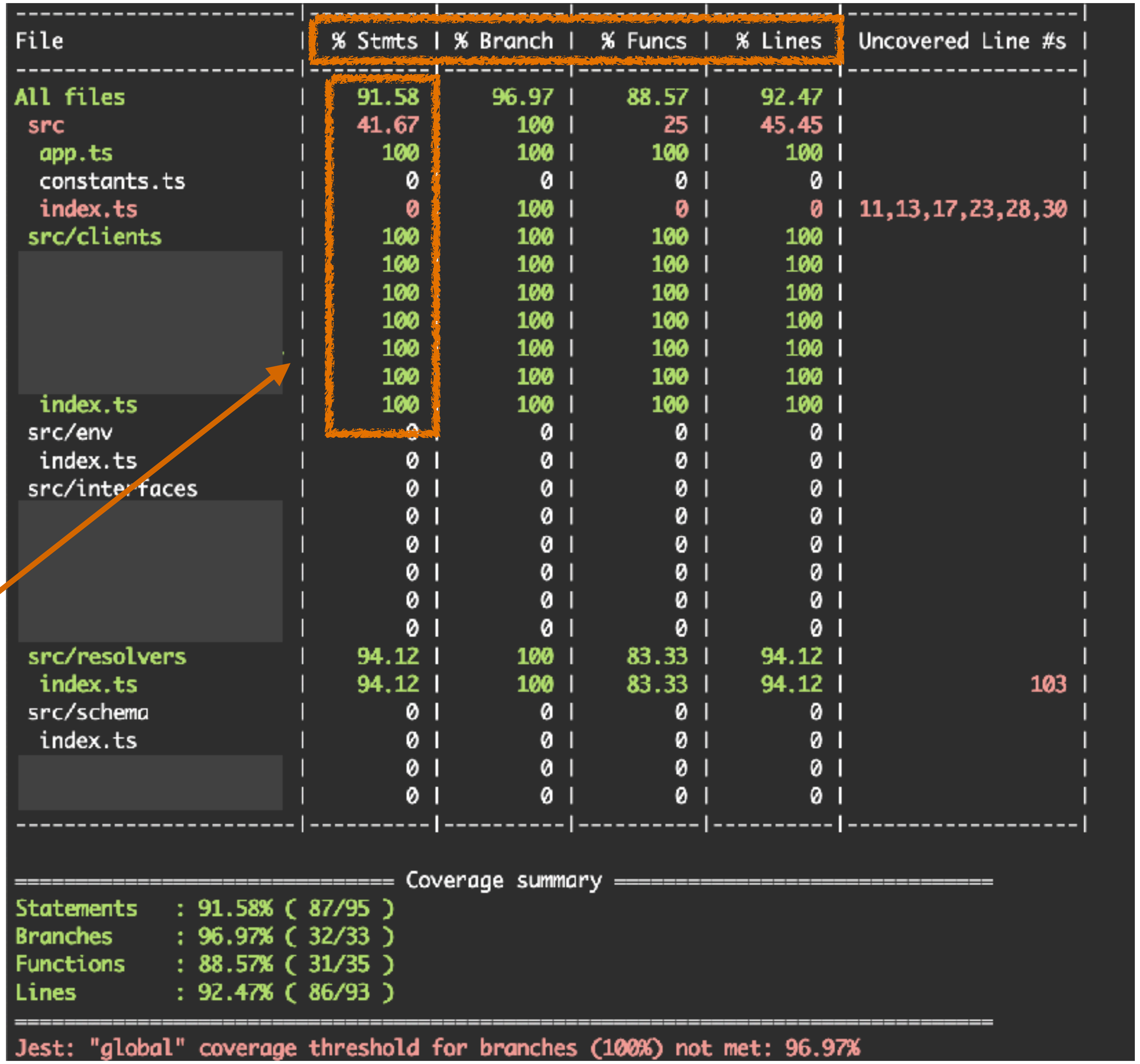
Traditional software uses test adequacy metrics



Problem

Traditional software uses test adequacy metrics





File	% Stmts	% Branch	% Funcs	% Lines	Uncovered Line #s
All files	91.58	96.97	88.57	92.47	
src	41.67	100	25	45.45	
app.ts	100	100	100	100	
constants.ts	0	0	0	0	
index.ts	0	100	0	0	11,13,17,23,28,30
src/clients	100	100	100	100	
[redacted]	100	100	100	100	
	100	100	100	100	
	100	100	100	100	
	100	100	100	100	
	100	100	100	100	
index.ts	100	100	100	100	
src/env	0	0	0	0	
index.ts	0	0	0	0	
src/interfaces	0	0	0	0	
[redacted]	0	0	0	0	
	0	0	0	0	
	0	0	0	0	
	0	0	0	0	
	0	0	0	0	
src/resolvers	94.12	100	83.33	94.12	
index.ts	94.12	100	83.33	94.12	103
src/schema	0	0	0	0	
index.ts	0	0	0	0	
[redacted]	0	0	0	0	
	0	0	0	0	

```
=============================== Coverage summary ===============================
Statements   : 91.58% ( 87/95 )
Branches     : 96.97% ( 32/33 )
Functions    : 88.57% ( 31/35 )
Lines        : 92.47% ( 86/93 )
================================================================================
Jest: "global" coverage threshold for branches (100%) not met: 96.97%
```

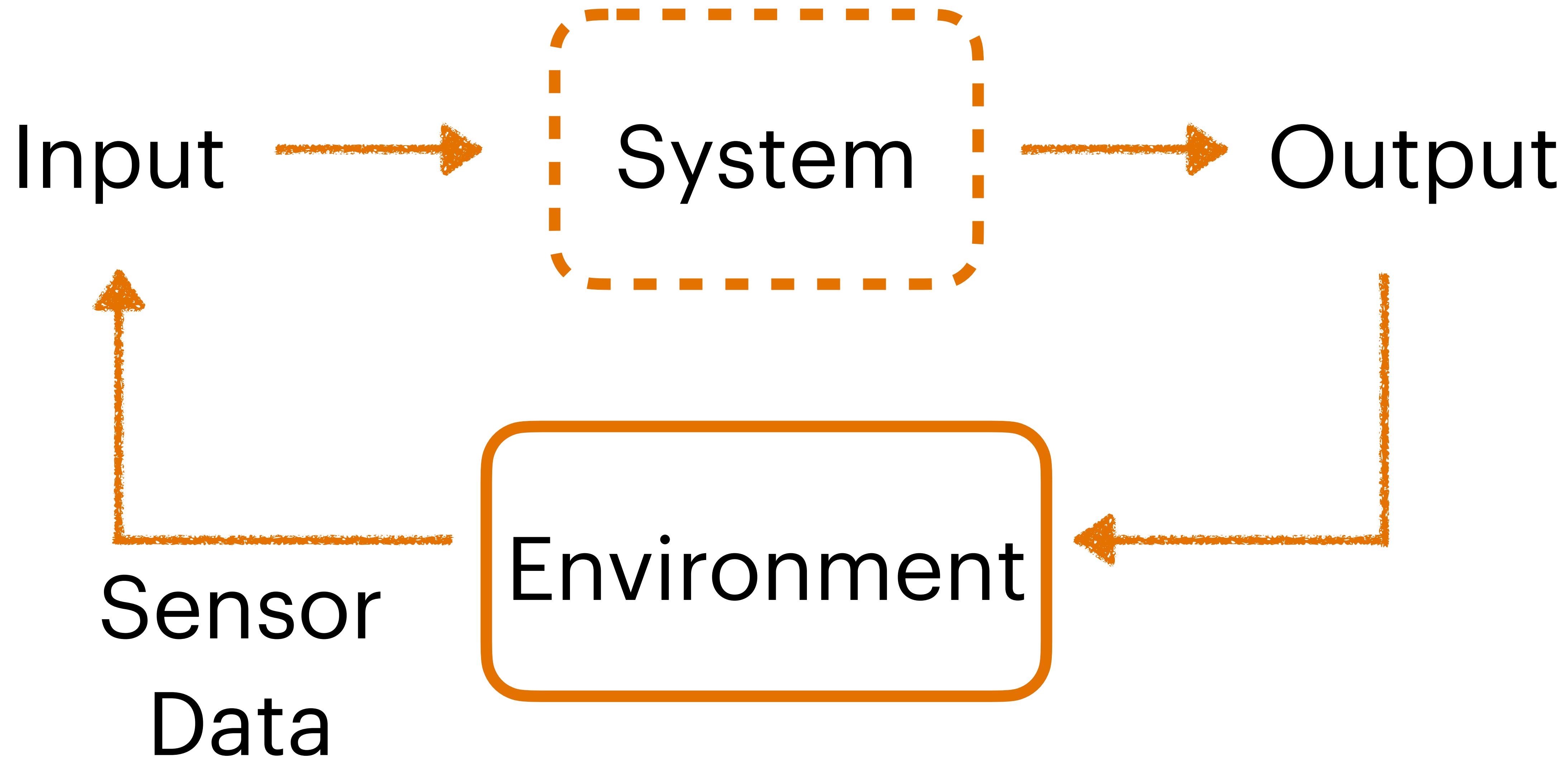
Problem

Why can't we do this with autonomous systems?

Input → System → Output

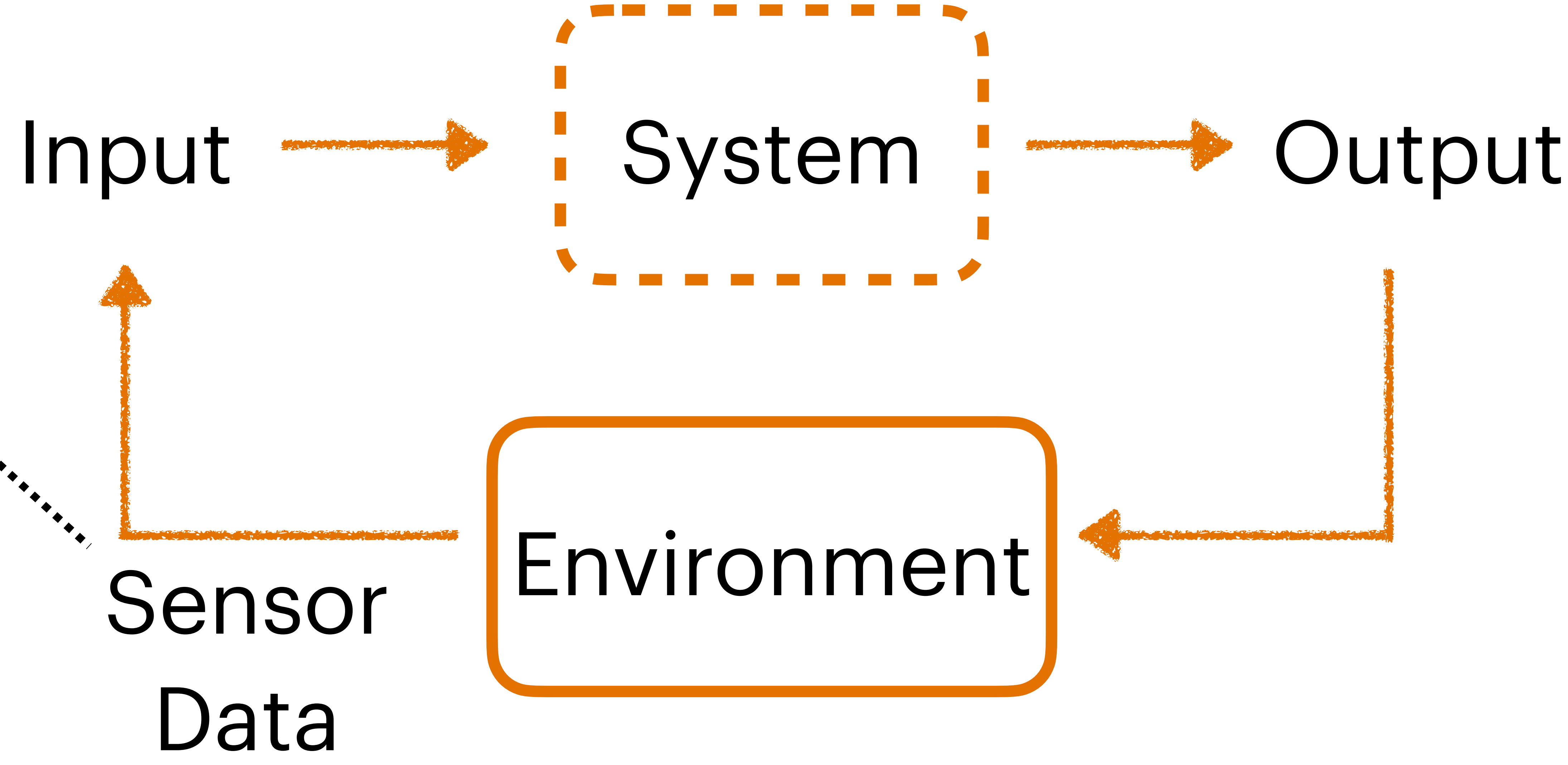
Problem

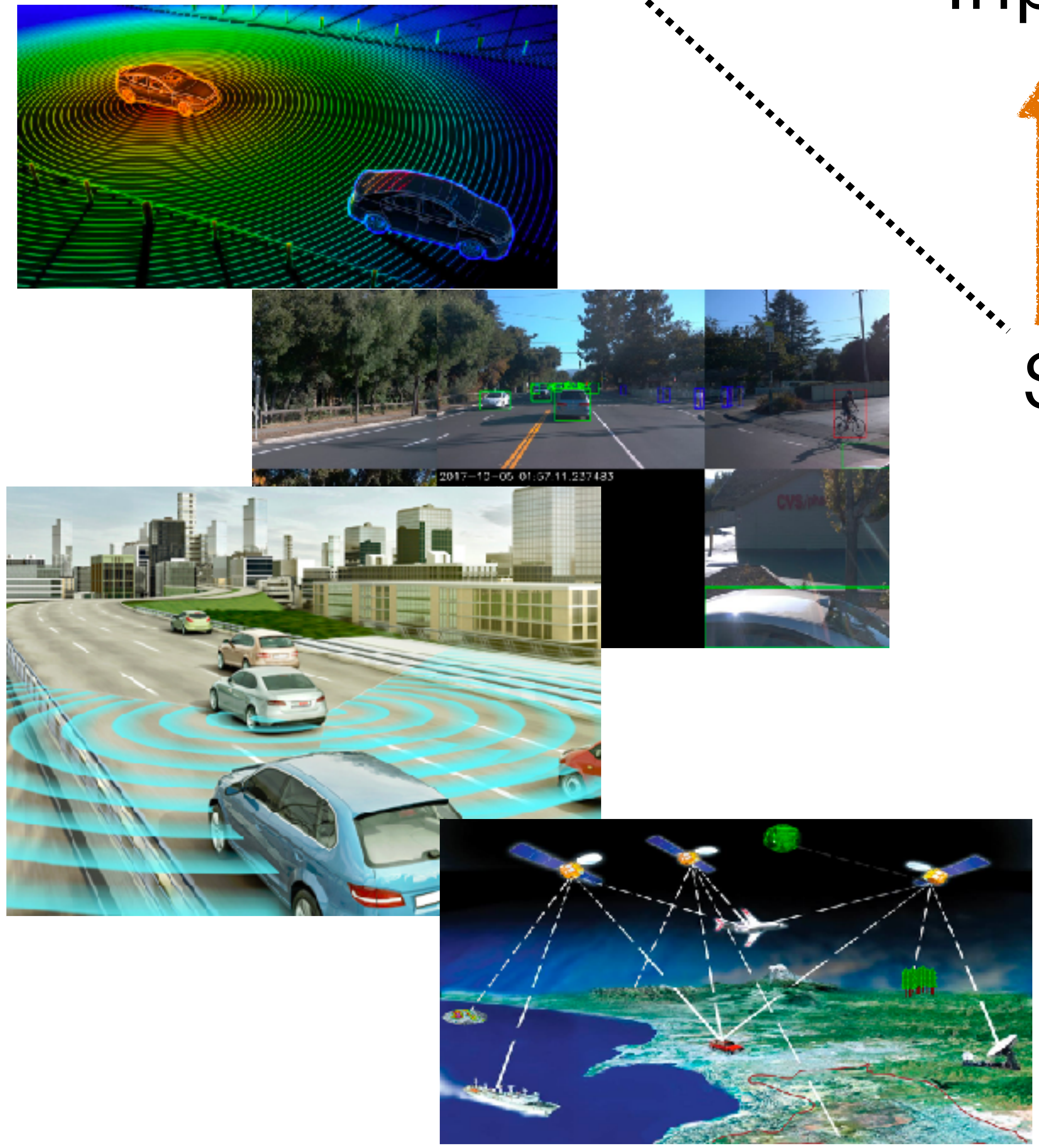
Why can’t we do this with autonomous systems?



Problem

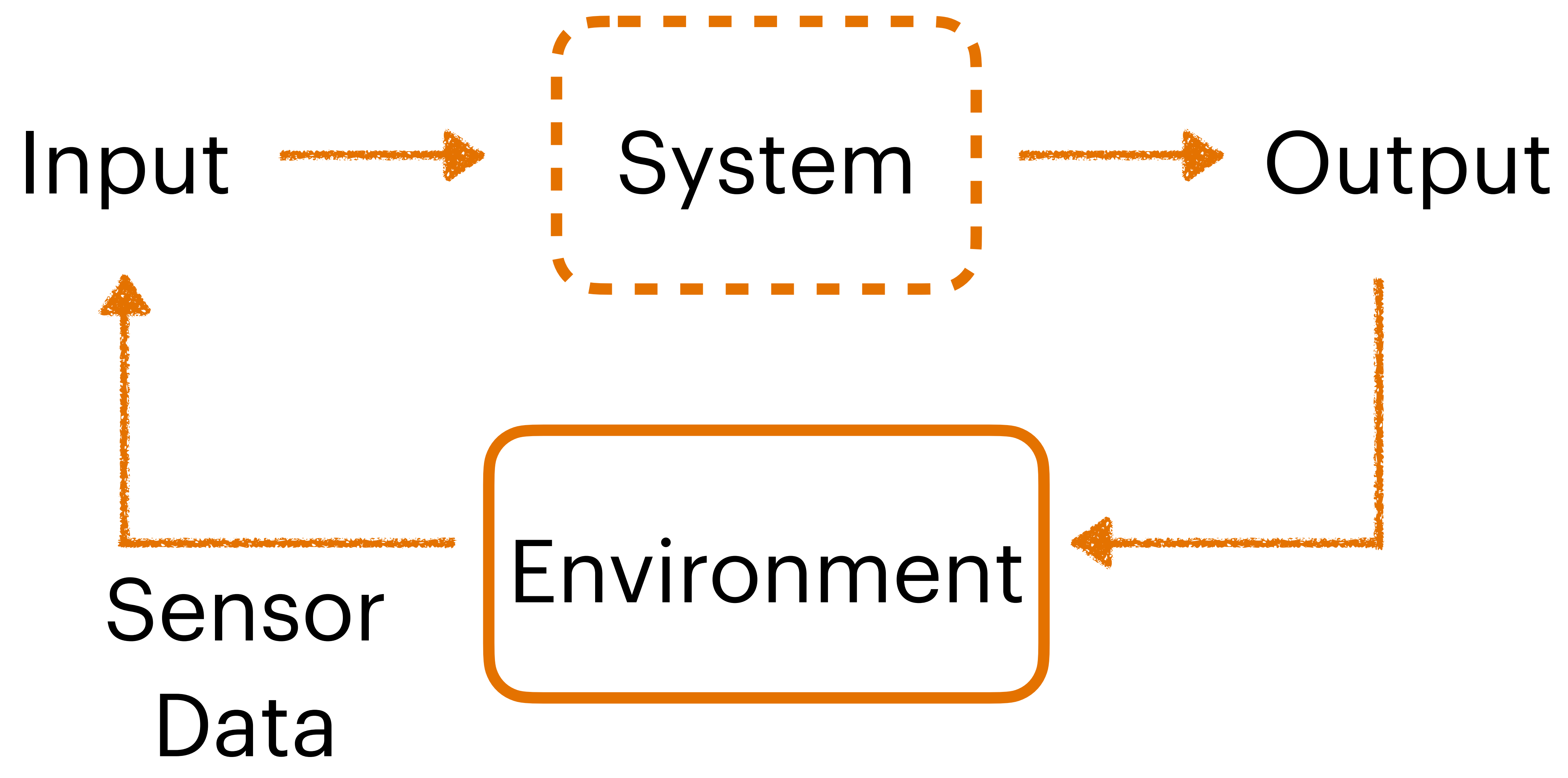
Why can't we do this with autonomous systems?





Problem

Why can't we do this with autonomous systems?



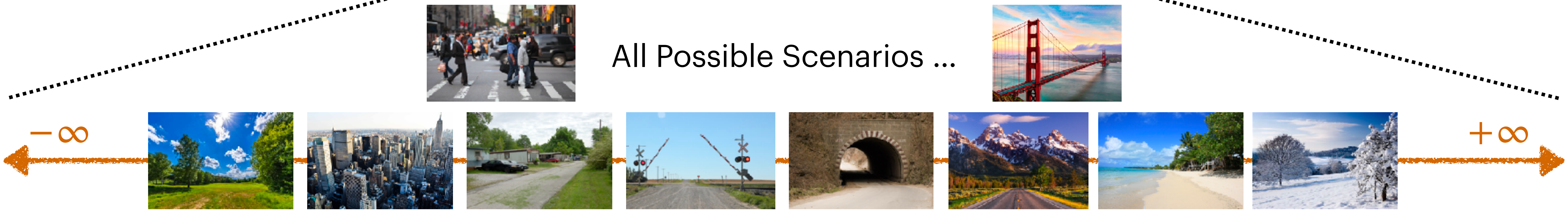
Problem

Why can't we do this with autonomous systems?

Input → System → Output

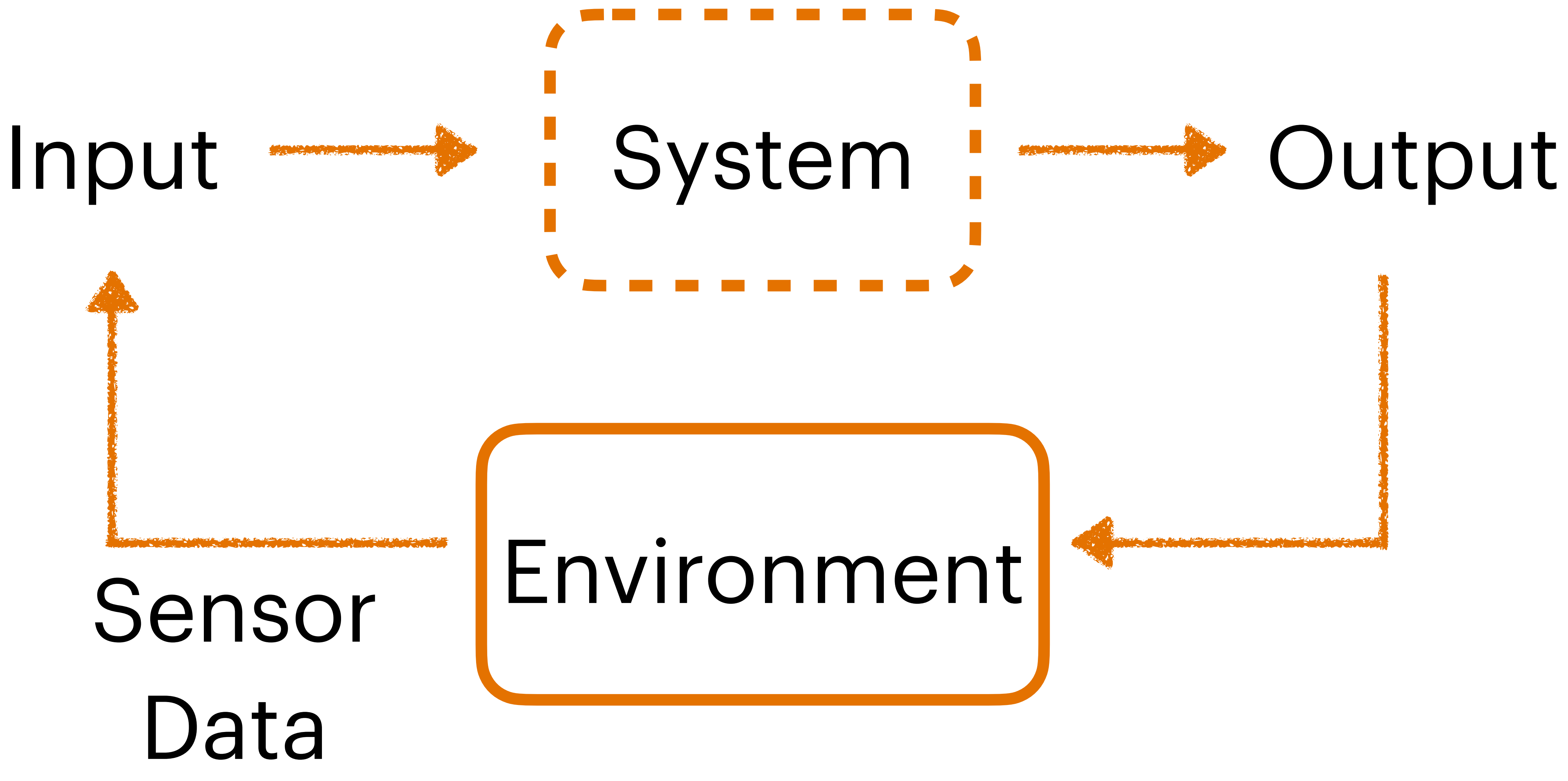
Environment

Sensor Data



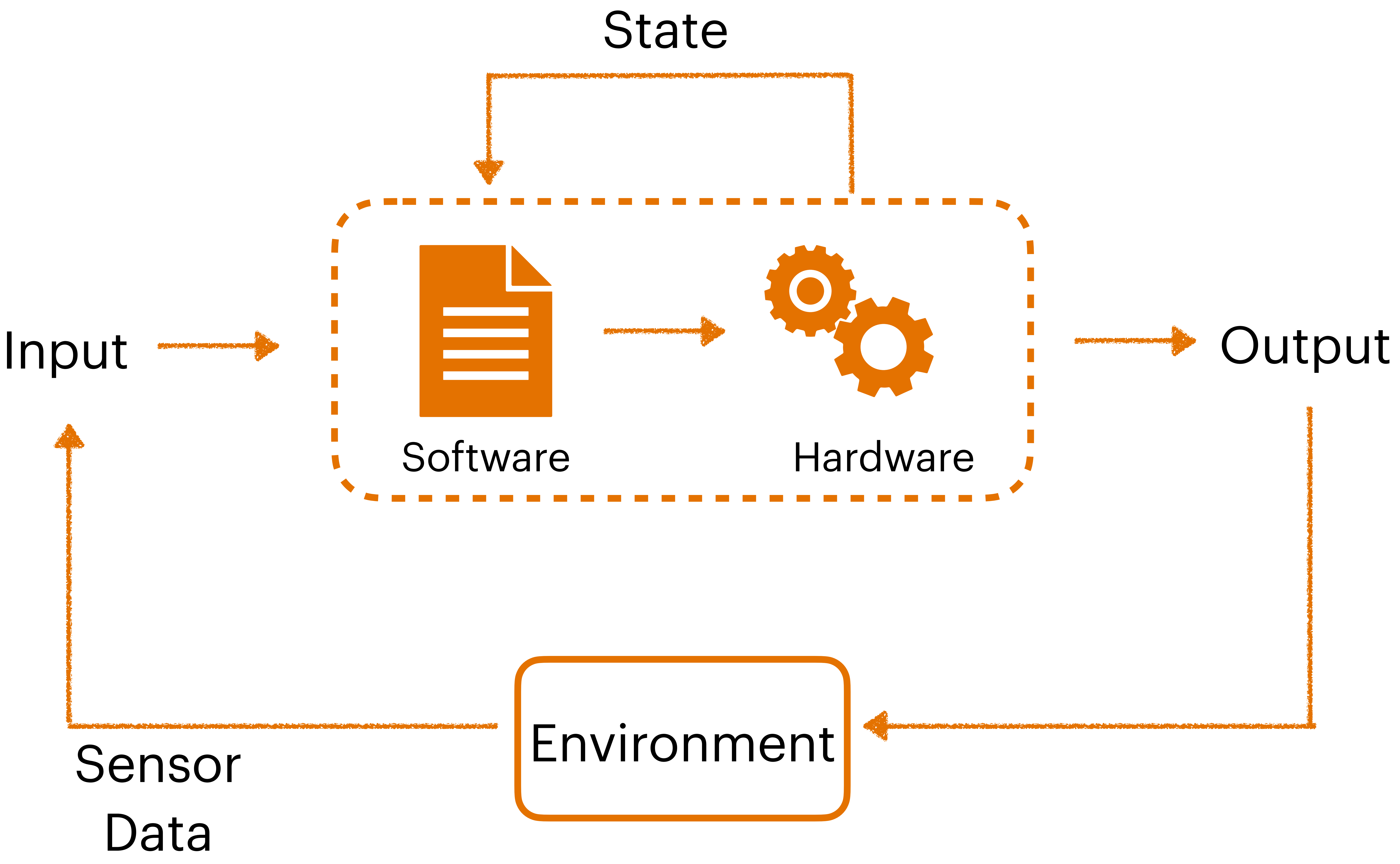
Problem

Why can’t we do this with autonomous systems?



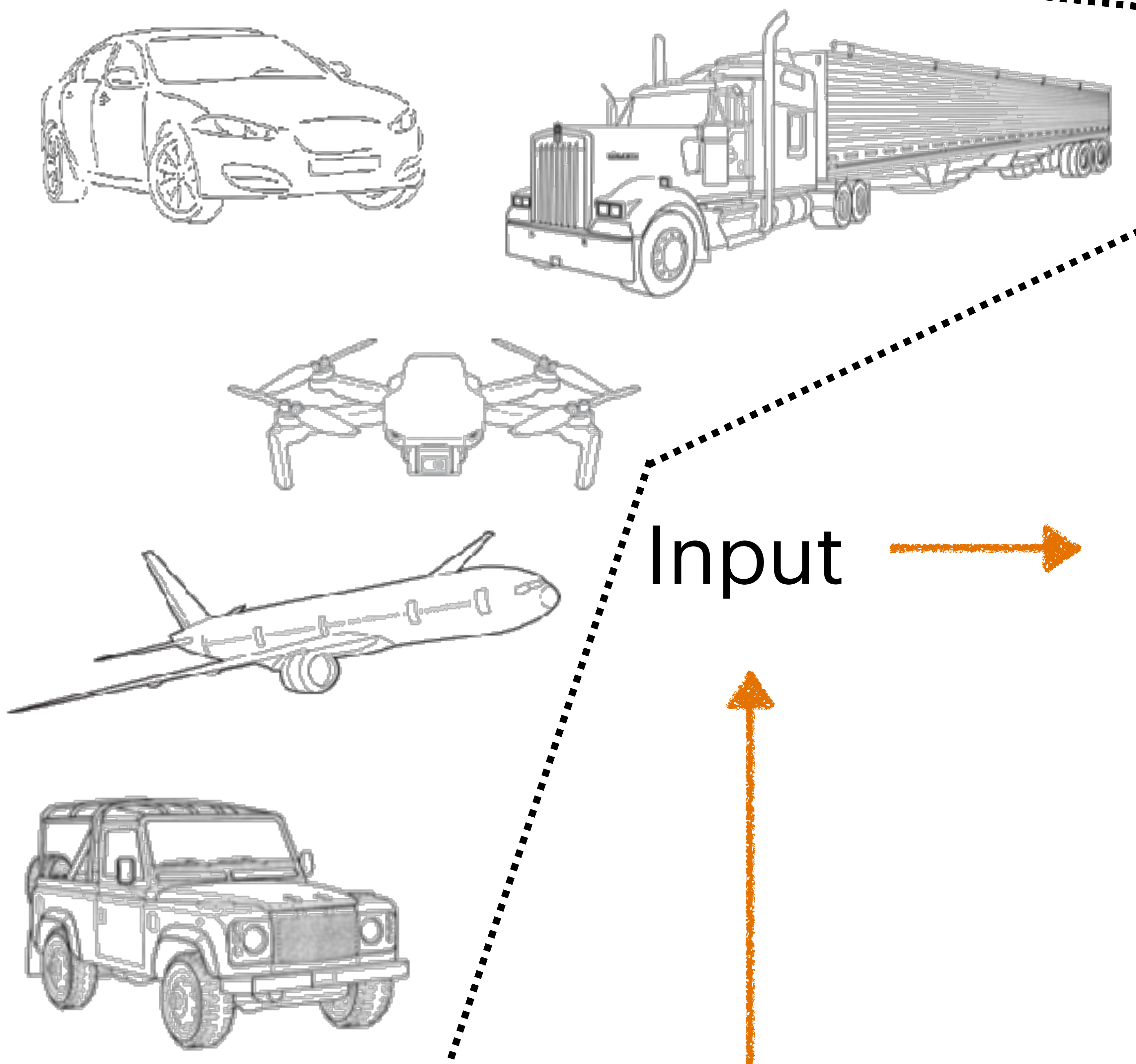
Problem

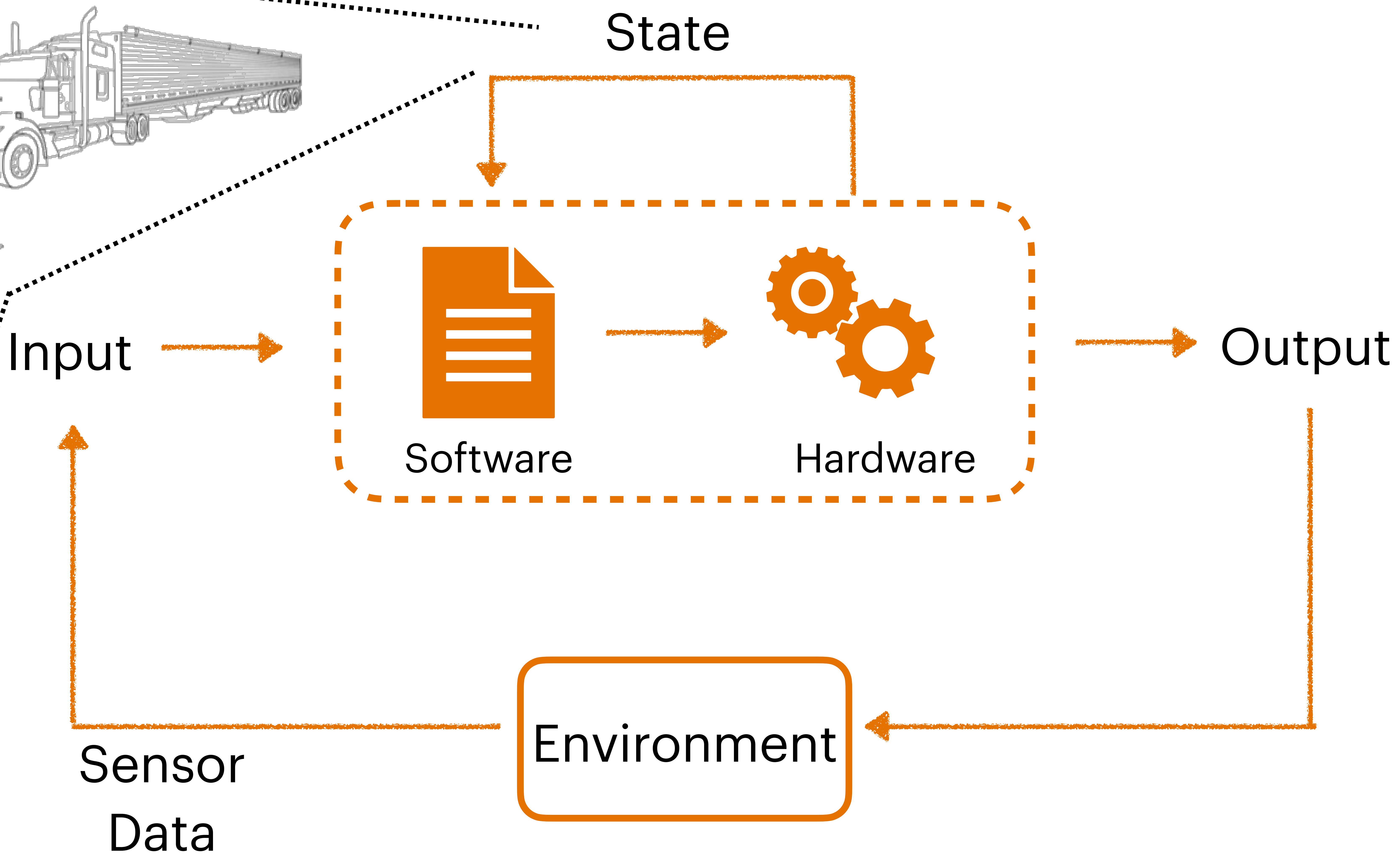
Why can’t we do this with autonomous systems?



Problem

Why can't we do this with autonomous systems?



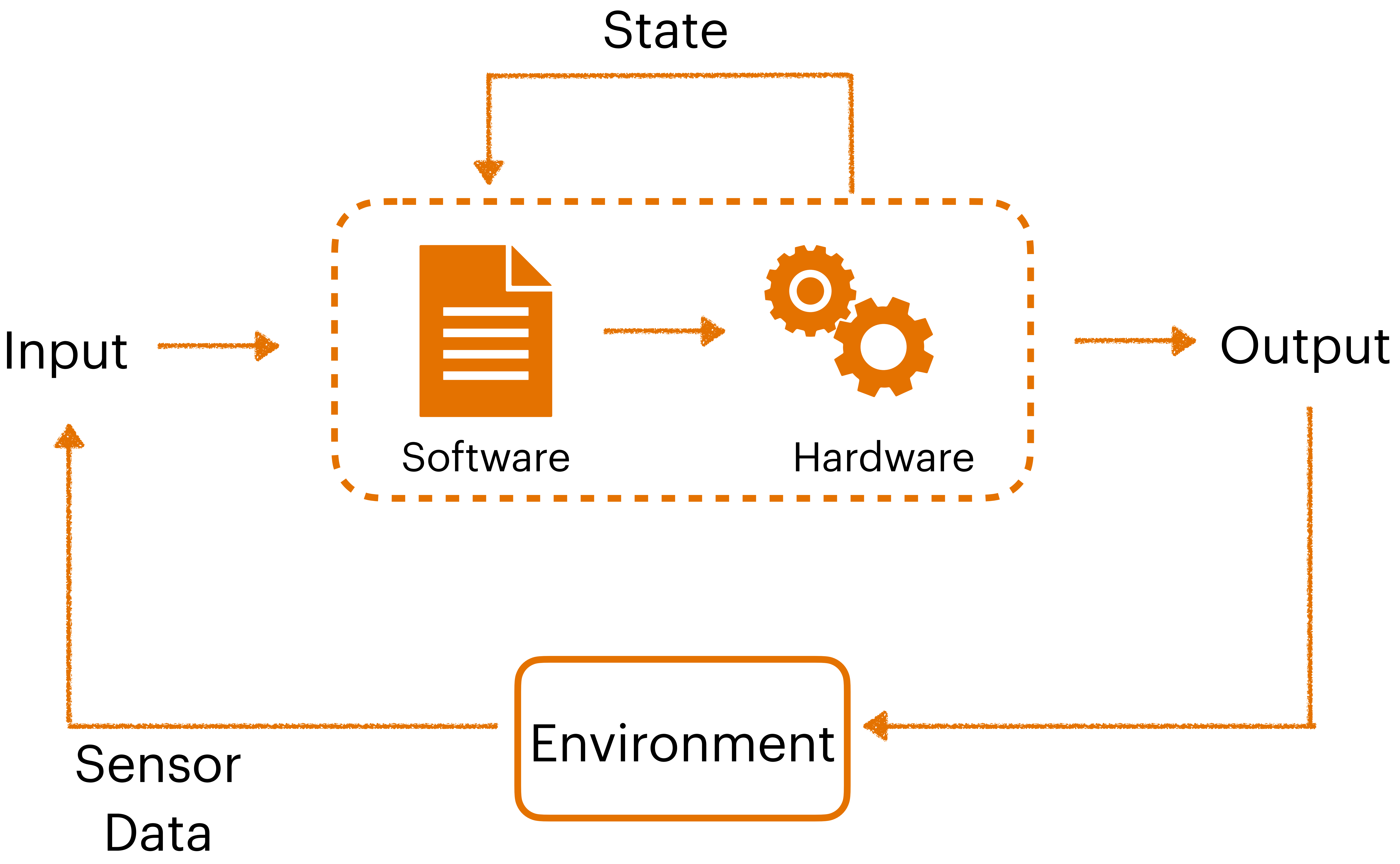


Problem

Environment

State

Why can’t we do this with autonomous systems?



Current Solutions

Current approaches are not cognizant of the environment and state

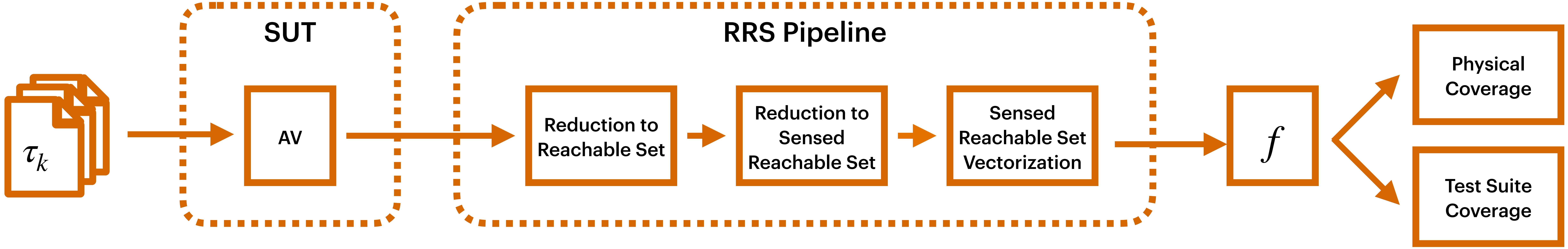
Coverage Metric	Account for Environments	Account for State
Structural Code Coverage	✗	✗
Miles Driven / Incident per Miles	✓	✗
Requirement Coverage	✓	✓
Scenario Coverage	✓	✗
Trajectory Coverage	✓	✓
Physical Coverage	✓	✓

Insight

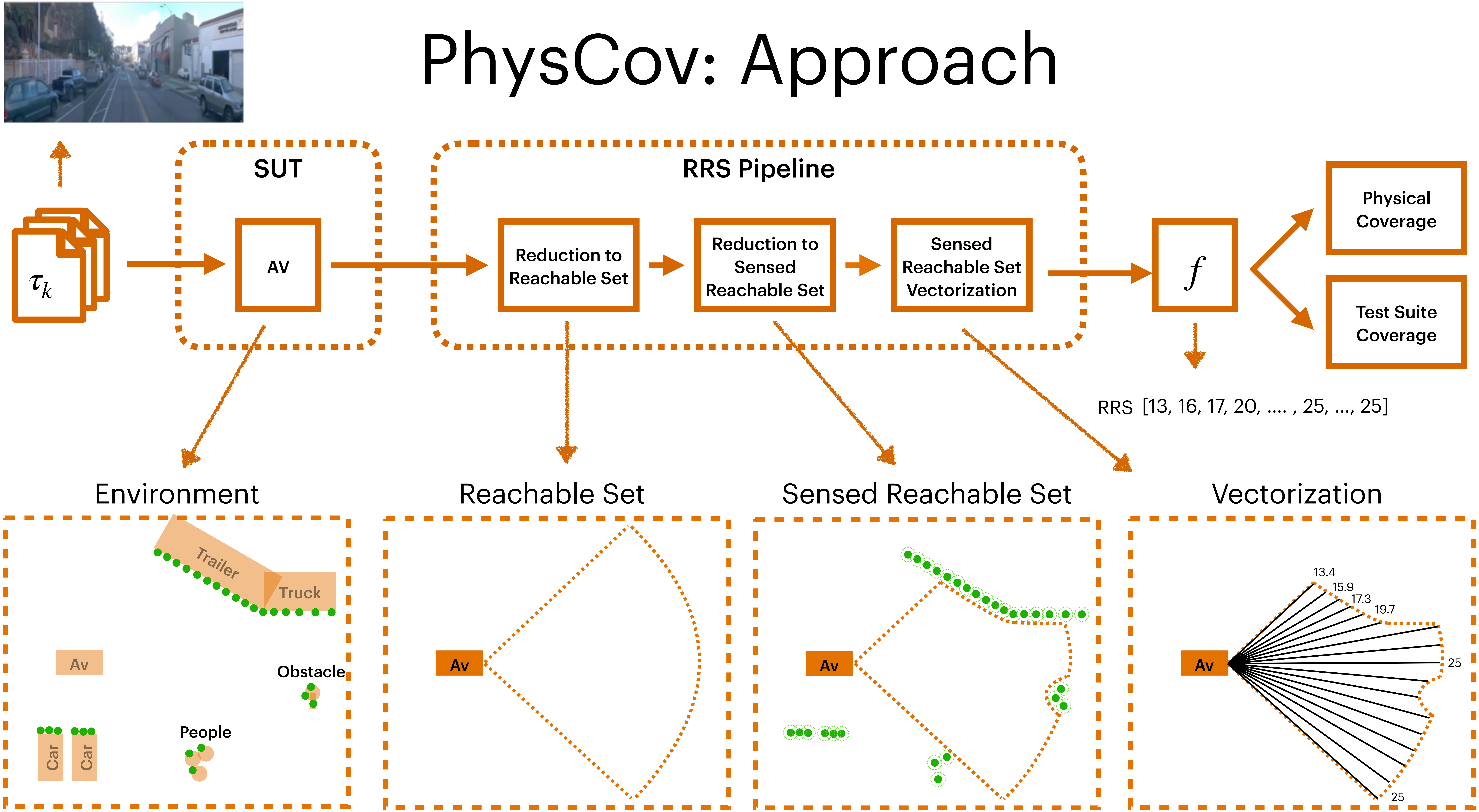
1) The environment is highly complex and practically infinite:
Only the sensed environment, which the vehicle can reach is important to the vehicles current behavior.

2) The vehicles state is dependent the specific systems hardware:
Kinematic models offer a way to abstract the state for any vehicle.

PhysCov: Approach



PhysCov: Approach



PhysCov: Approach

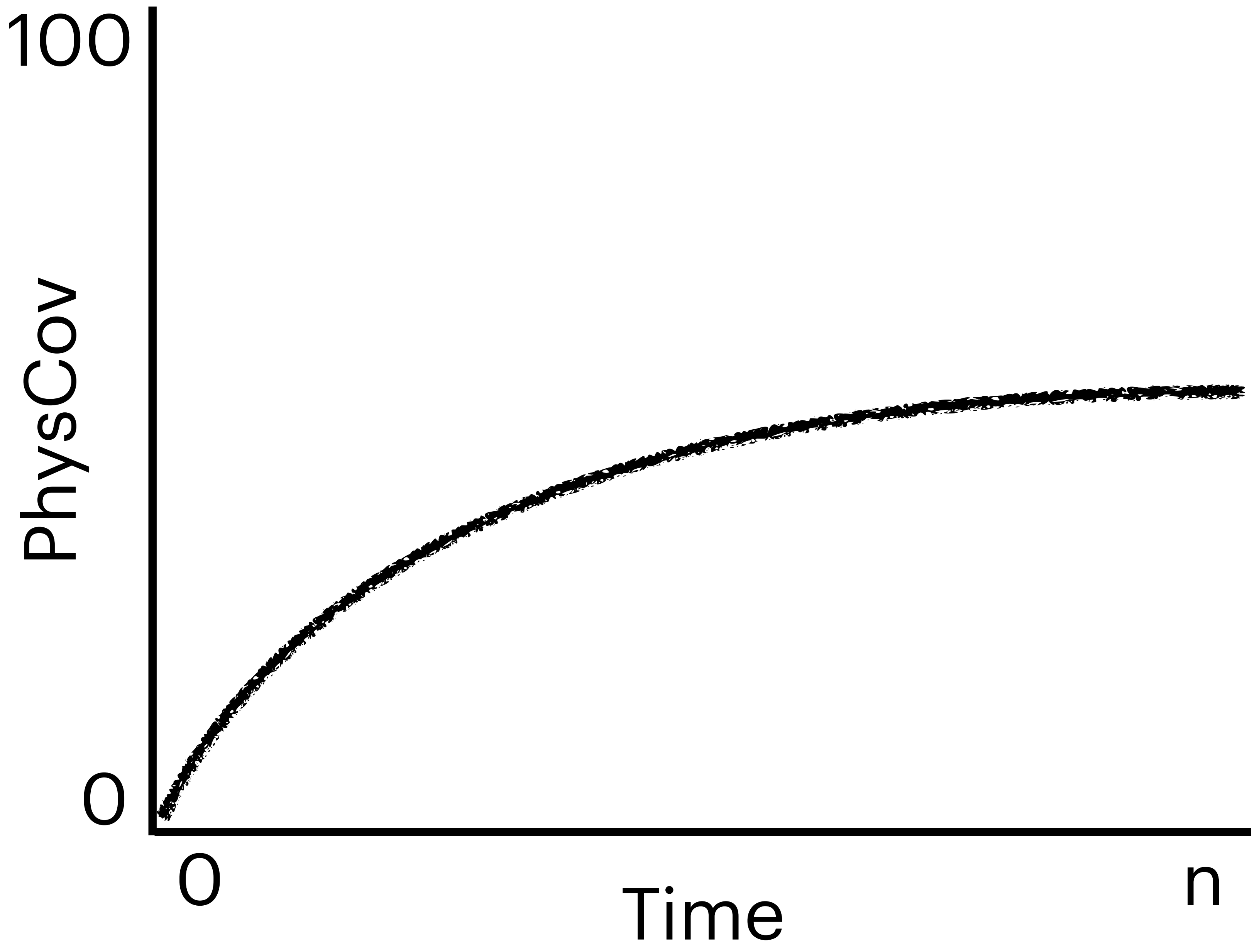
PhysCov: Approach

$\alpha = \{(e_1^{sen}, s_1), \ldots\}$

[5. 5. 15. 15. 35. 35. 25. 35. 25. 5.]

$\beta = E \times S$

$$PhysCov = \frac{\alpha}{\beta}$$



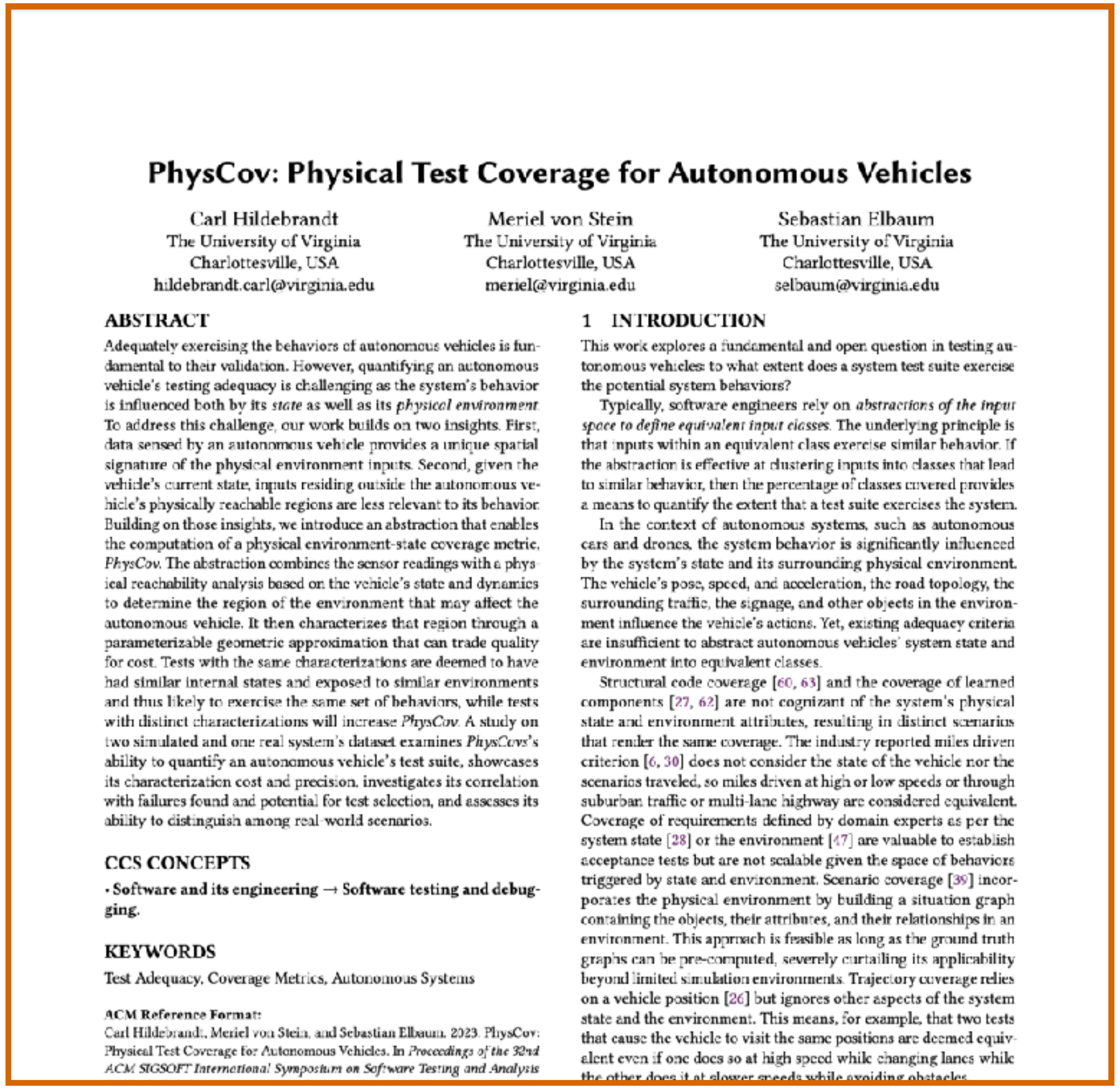
Start: [0, 0, 0, ... 0]

X = Max size reachable set

End: [X, X, X, ..., X]

PhysCov: Approach

We couldn't cover all the details and we encourage you to read the paper!



PhysCov: Physical Test Coverage for Autonomous Vehicles

Carl Hildebrandt
The University of Virginia
Charlottesville, USA
hildebrandt.carl@virginia.edu

Meriel von Stein
The University of Virginia
Charlottesville, USA
meriel@virginia.edu

Sebastian Elbaum
The University of Virginia
Charlottesville, USA
selbaum@virginia.edu

ABSTRACT

Adequately exercising the behaviors of autonomous vehicles is fundamental to their validation. However, quantifying an autonomous vehicle's testing adequacy is challenging as the system's behavior is influenced both by its *state* as well as its *physical environment*. To address this challenge, our work builds on two insights. First, data sensed by an autonomous vehicle provides a unique spatial signature of the physical environment inputs. Second, given the vehicle's current state, inputs residing outside the autonomous vehicle's physically reachable regions are less relevant to its behavior. Building on those insights, we introduce an abstraction that enables the computation of a physical environment-state coverage metric, *PhysCov*. The abstraction combines the sensor readings with a physical reachability analysis based on the vehicle's state and dynamics to determine the region of the environment that may affect the autonomous vehicle. It then characterizes that region through a parameterizable geometric approximation that can trade quality for cost. Tests with the same characterizations are deemed to have had similar internal states and exposed to similar environments and thus likely to exercise the same set of behaviors, while tests with distinct characterizations will increase *PhysCov*. A study on two simulated and one real system's dataset examines *PhysCov's* ability to quantify an autonomous vehicle's test suite, showcases its characterization cost and precision, investigates its correlation with failures found and potential for test selection, and assesses its ability to distinguish among real-world scenarios.

CCS CONCEPTS

• **Software and its engineering → Software testing and debugging.**

KEYWORDS

Test Adequacy, Coverage Metrics, Autonomous Systems

ACM Reference Format:
Carl Hildebrandt, Meriel von Stein, and Sebastian Elbaum. 2023. PhysCov: Physical Test Coverage for Autonomous Vehicles. In *Proceedings of the 32nd ACM SIGSOFT International Symposium on Software Testing and Analysis*

1 INTRODUCTION

This work explores a fundamental and open question in testing autonomous vehicles: to what extent does a system test suite exercise the potential system behaviors?

Typically, software engineers rely on *abstractions of the input space to define equivalent input classes*. The underlying principle is that inputs within an equivalent class exercise similar behavior. If the abstraction is effective at clustering inputs into classes that lead to similar behavior, then the percentage of classes covered provides a means to quantify the extent that a test suite exercises the system.

In the context of autonomous systems, such as autonomous cars and drones, the system behavior is significantly influenced by the system's state and its surrounding physical environment. The vehicle's pose, speed, and acceleration, the road topology, the surrounding traffic, the signage, and other objects in the environment influence the vehicle's actions. Yet, existing adequacy criteria are insufficient to abstract autonomous vehicles' system state and environment into equivalent classes.

Structural code coverage [60, 63] and the coverage of learned components [27, 62] are not cognizant of the system's physical state and environment attributes, resulting in distinct scenarios that render the same coverage. The industry reported miles driven criterion [6, 30] does not consider the state of the vehicle nor the scenarios traveled, so miles driven at high or low speeds or through suburban traffic or multi-lane highway are considered equivalent. Coverage of requirements defined by domain experts as per the system state [28] or the environment [47] are valuable to establish acceptance tests but are not scalable given the space of behaviors triggered by state and environment. Scenario coverage [39] incorporates the physical environment by building a situation graph containing the objects, their attributes, and their relationships in an environment. This approach is feasible as long as the ground truth graphs can be pre-computed, severely curtailing its applicability beyond limited simulation environments. Trajectory coverage relies on a vehicle position [26] but ignores other aspects of the system state and the environment. This means, for example, that two tests that cause the vehicle to visit the same positions are deemed equivalent even if one does so at high speed while changing lanes while the other does it at slower speeds while avoiding obstacles.

Study

We asked three different research questions:

RQ1) How effective RRS at grouping equivalent environment inputs such that they cause similar behaviors?

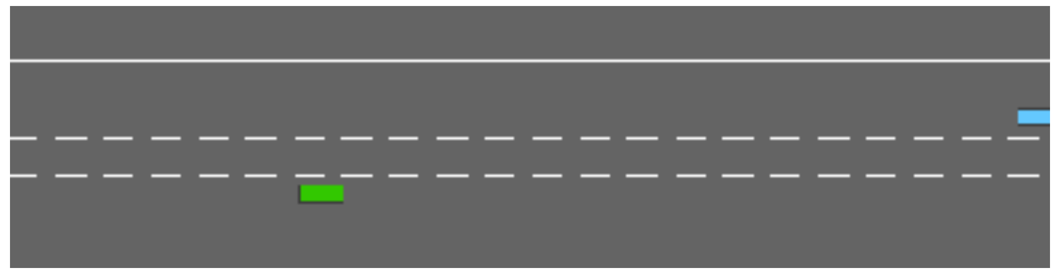
RQ2) How effective is PhysCov at selecting tests that induce unique failures?

RQ3) Can PhysCov distinguish similar from different scenarios?

Environments

HighwayEnv

1,000,000 tests



BeamNG

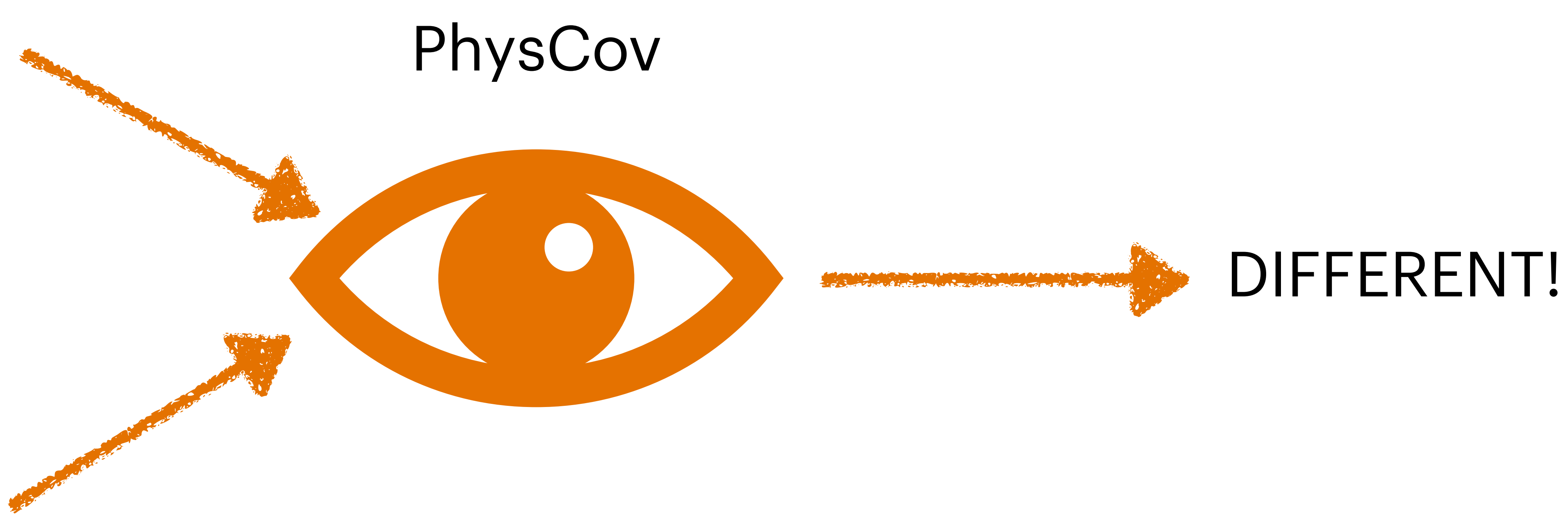
10,000 tests

Waymo Open Dataset

4 Hours 26 Minutes Driving

Research Question 1

How effective RRS at grouping equivalent environment inputs such that they cause similar behaviors?



Research Question 1

How effective RRS at grouping equivalent environment inputs such that they cause similar behaviors?

Class 1	Class 2	Class 3
Test a ✓	Test a ✗	Test a ✗
Test b ✓	Test b ✗	Test b ✓

Research Question 1

How effective RRS at grouping equivalent environment inputs such that they cause similar behaviors?

10,000 tests

Structural Code Coverage

- Line Coverage
- Branch Coverage
- Intraprocedural prime path coverage
- Intraprocedural path coverage
- Absolute path coverage

Trajectory Coverage

- Improved to include irregular maps

PhysCov

- Ψ_1 - RRS of length 1
- Ψ_5 - RRS of length 5
- Ψ_{10} - RRS of length 10

Class 1

Test a
Test b

Class 2

Test a
Test b

Class 3

Test a
Test b

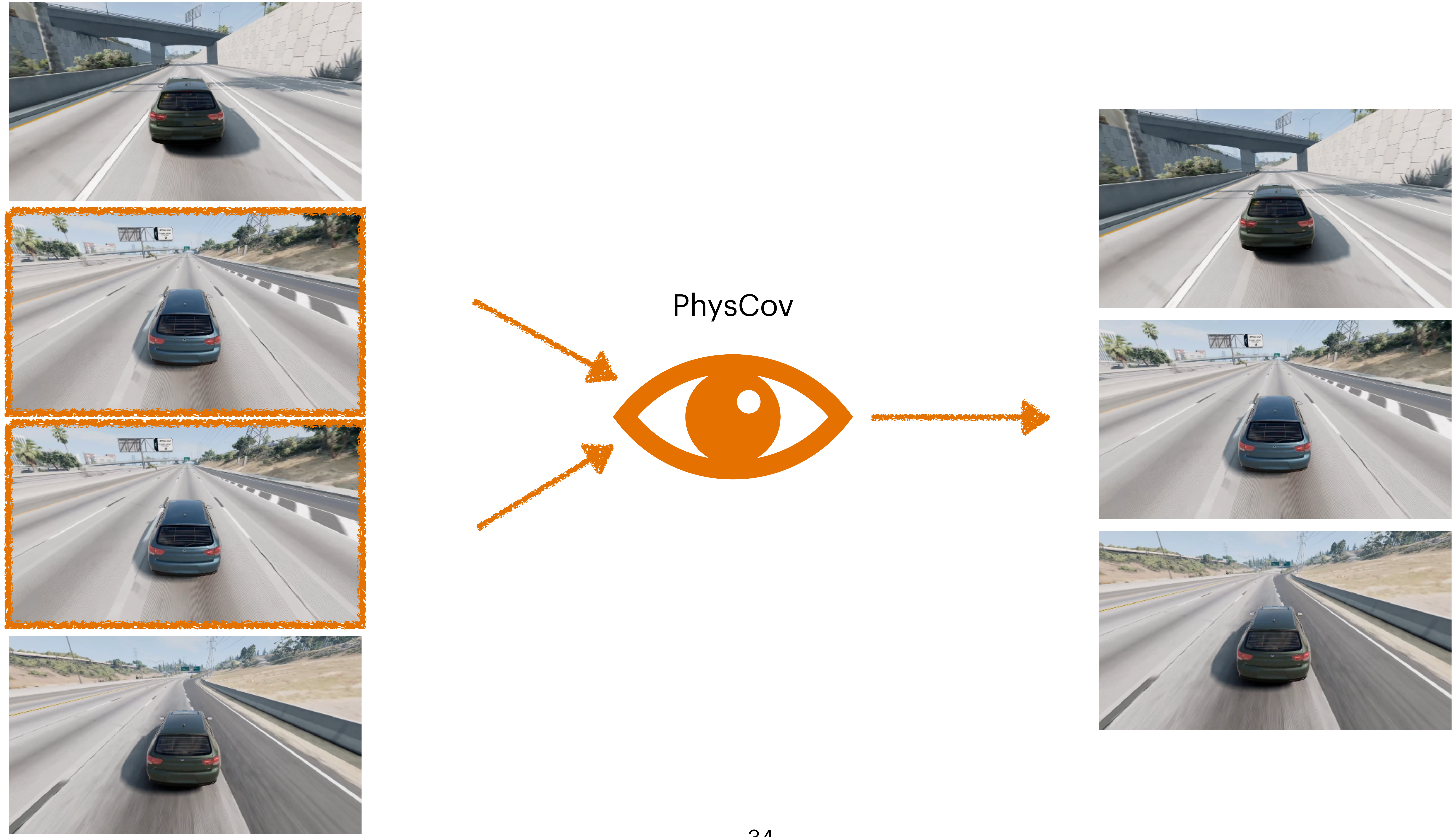
Research Question 1

How effective RRS at grouping equivalent environment inputs such that they cause similar behaviors?

Coverage Metric	Equivalent Classes	Percentage Inconsitency
Line	151	65%
Branch	146	58%
Intraprocedural Prime Path Coverage	421	75%
Intraprocedural Path Coverage	10000	——
Absolute Path Coverage	10000	——
Trajectory Coverage	10000	——
Physical Coverage: Ψ_1	682	57%
Physical Coverage: Ψ_5	1594	40%
Physical Coverage: Ψ_{10}	3628	32%

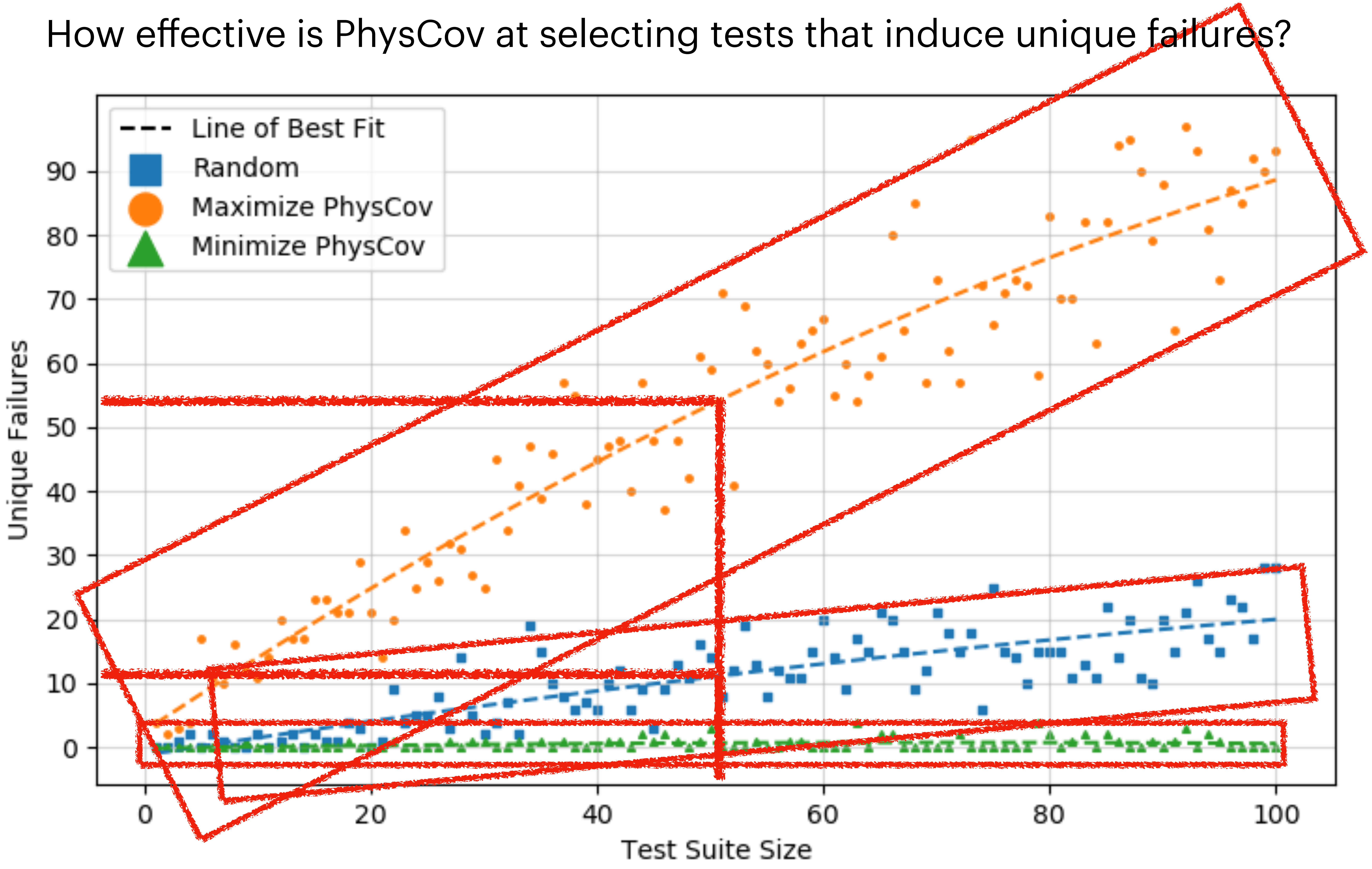
Research Question 2

How effective is PhysCov at selecting tests that induce unique failures?



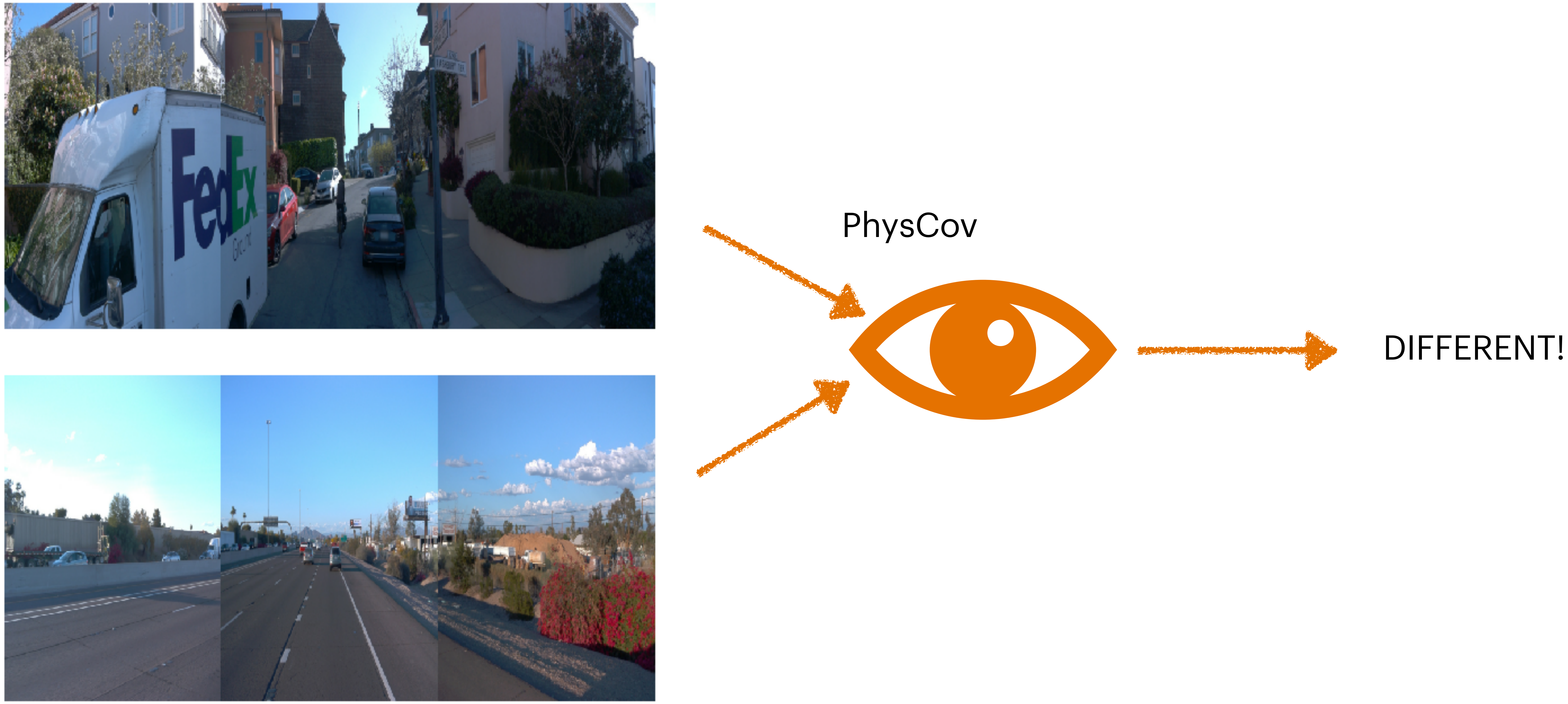
Research Question 2

How effective is PhysCov at selecting tests that induce unique failures?



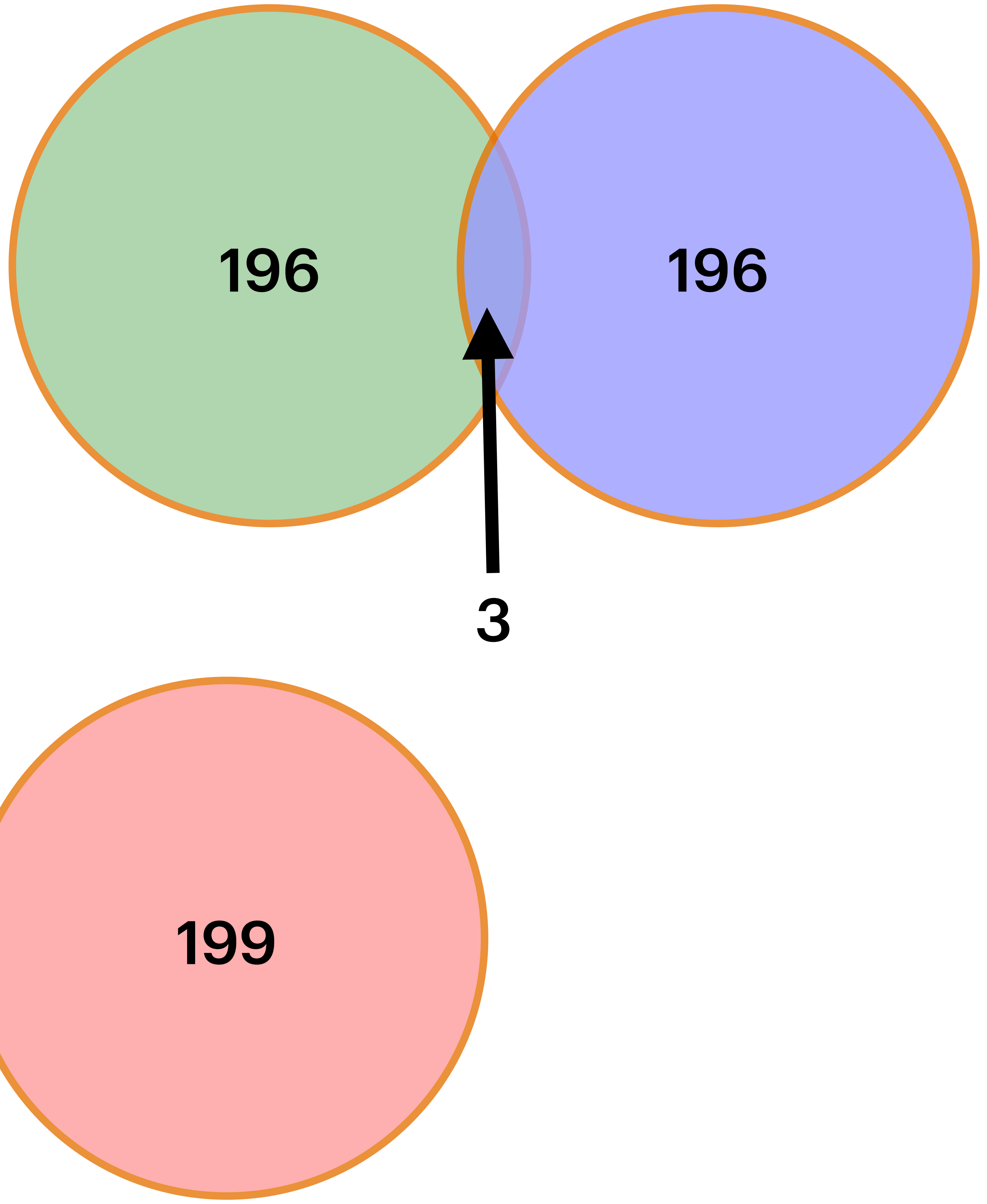
Research Question 3

Can PhysCov distinguish similar from different scenarios?



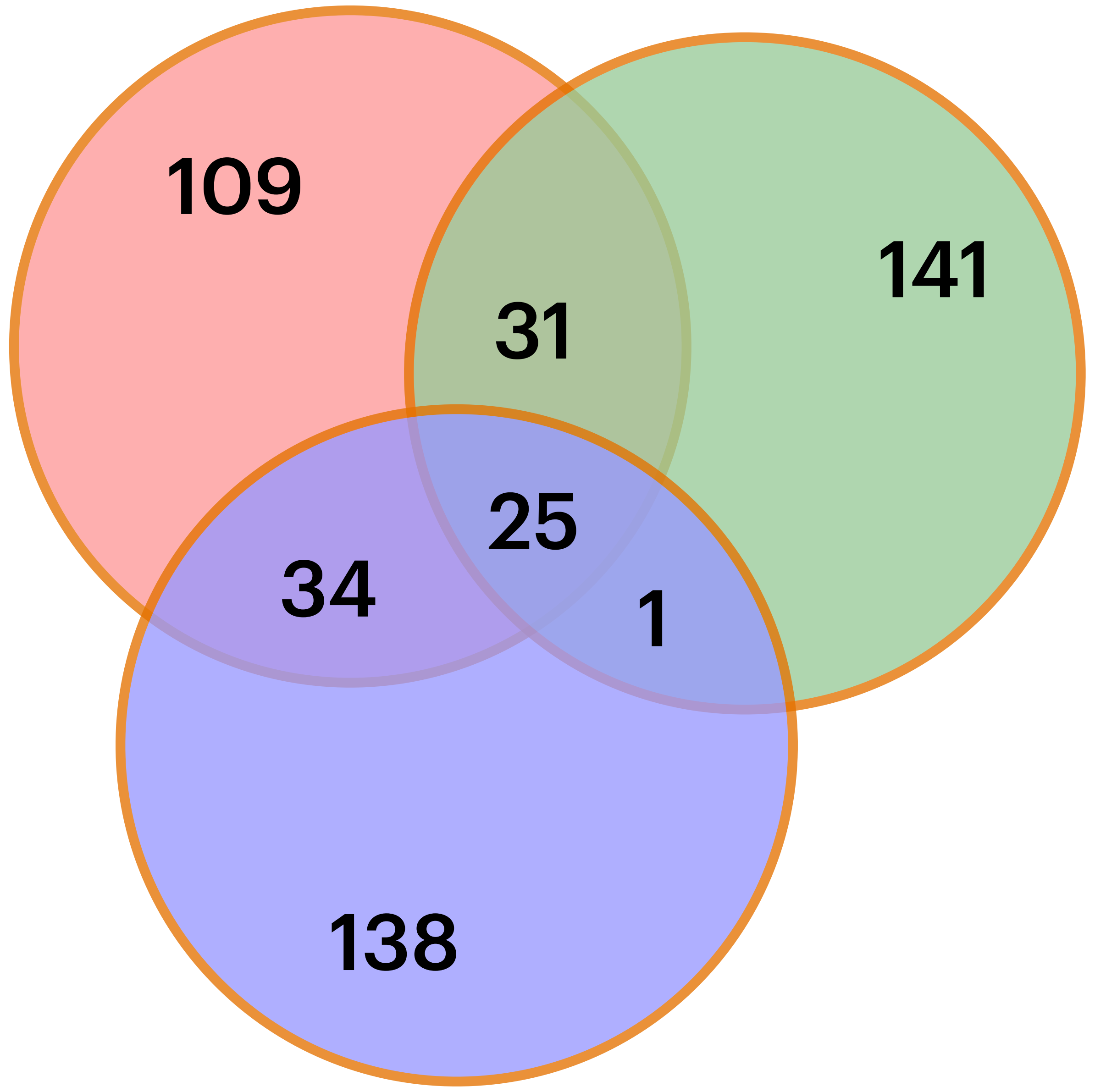
Research Question 3

Can PhysCov distinguish similar from different scenarios?



Research Question 3

Can PhysCov distinguish similar from different scenarios?



Research Question 3

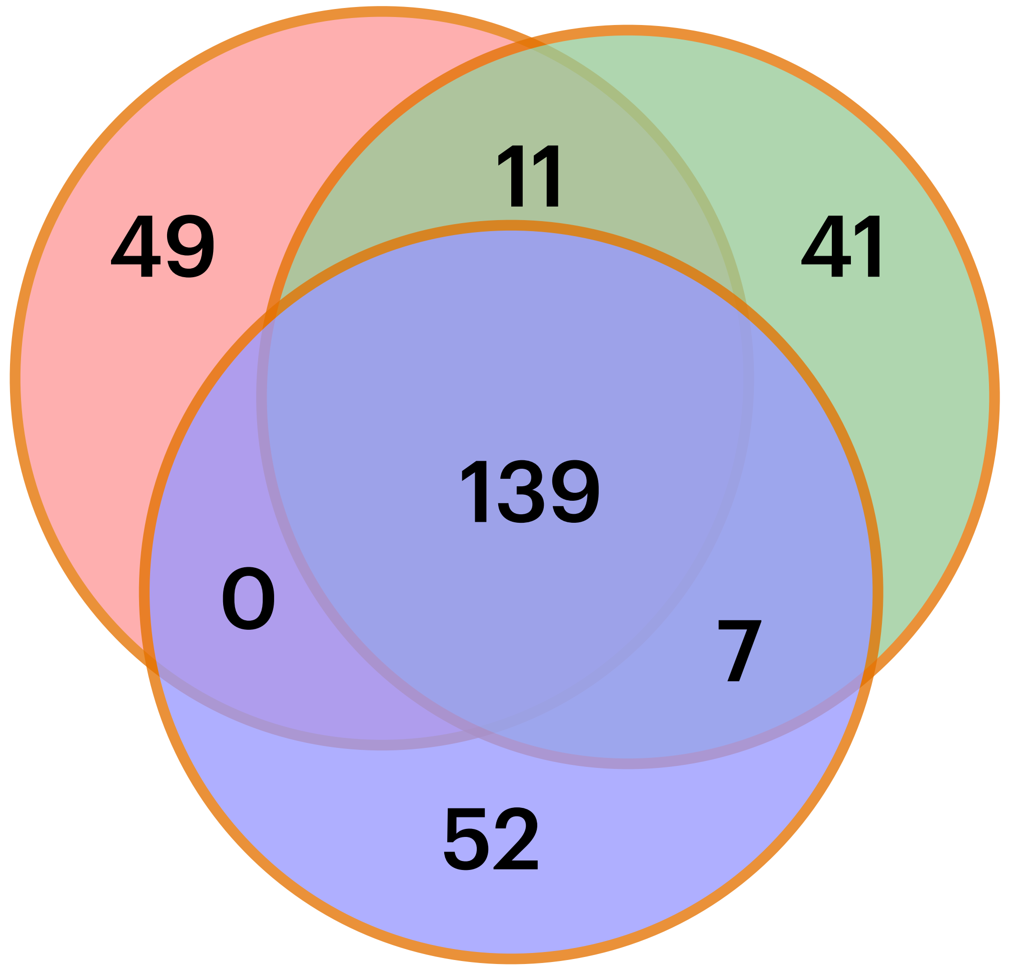
Can PhysCov distinguish similar from different scenarios?

PhysCov

DIFFERENT!

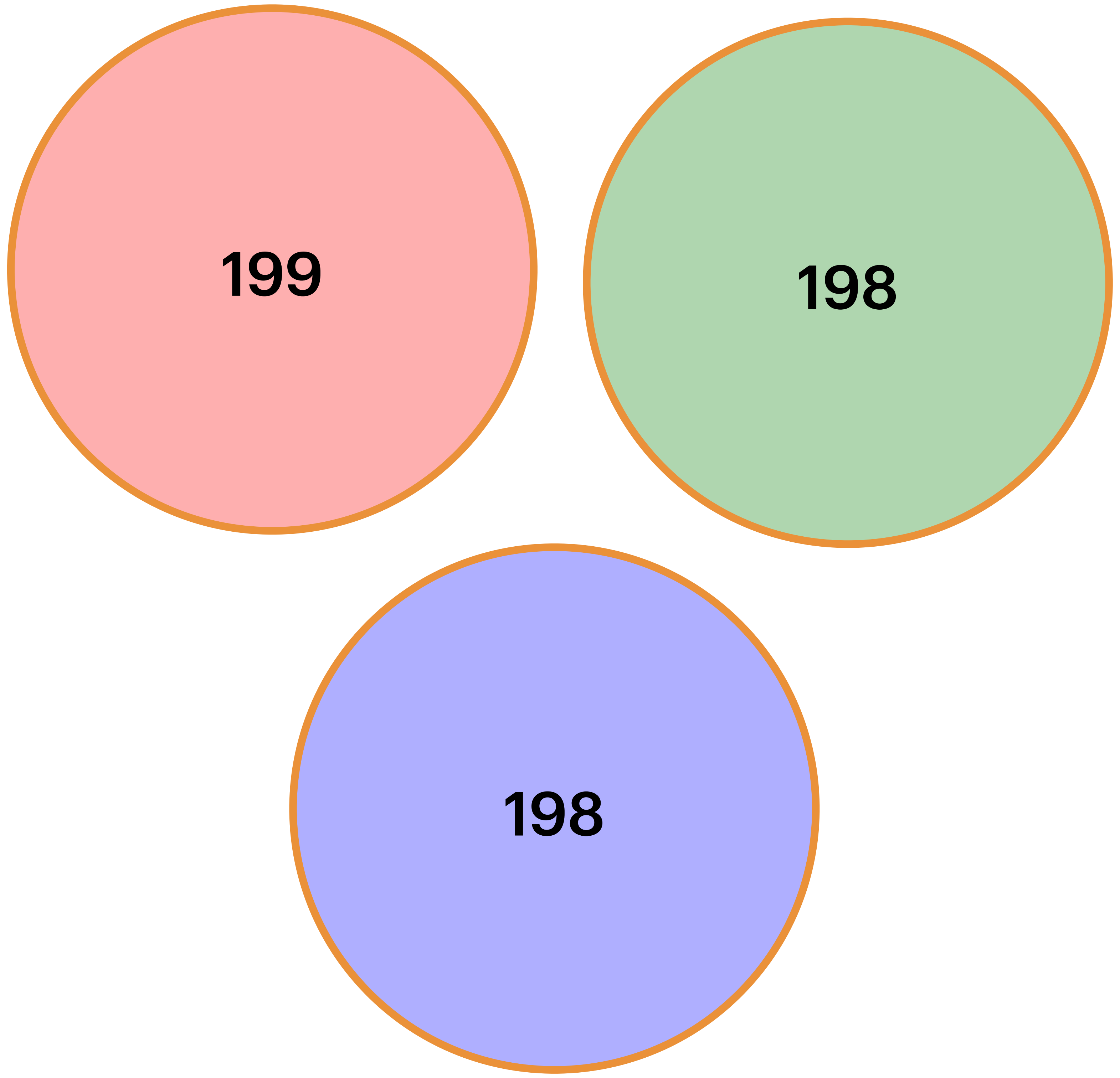
Research Question 3

Can PhysCov distinguish similar from different scenarios?

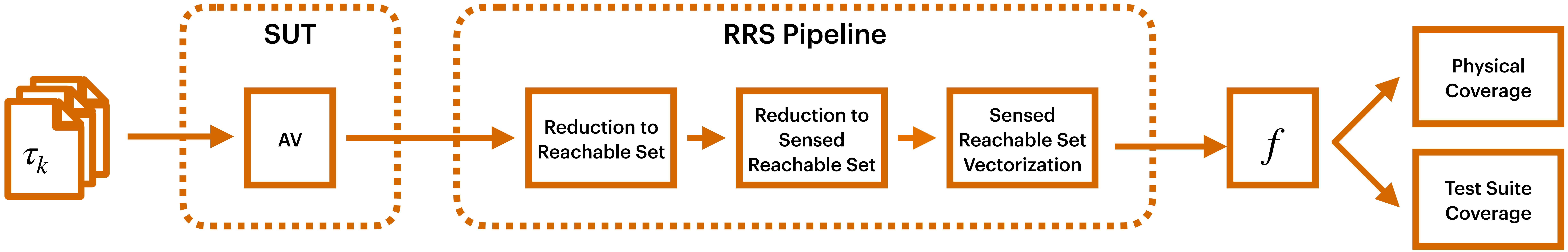


Research Question 3

Can PhysCov distinguish similar from different scenarios?



Conclusion



Coverage Metric	Equivalent Classes	Percentage
Line	151	65%
Branch	146	58%
Intraprocedural Prime Path Coverage	421	75%
Intraprocedural Path Coverage	10000	——
Absolute Path Coverage	10000	——
Trajectory Coverage	10000	——
Physical Coverage:	682	57%
Physical Coverage:	1594	40%
Physical Coverage:	3628	32%