# Introduction to the course: Verifying cyberphysical systems

Verifying cyberphysical systems

August 23th 2011

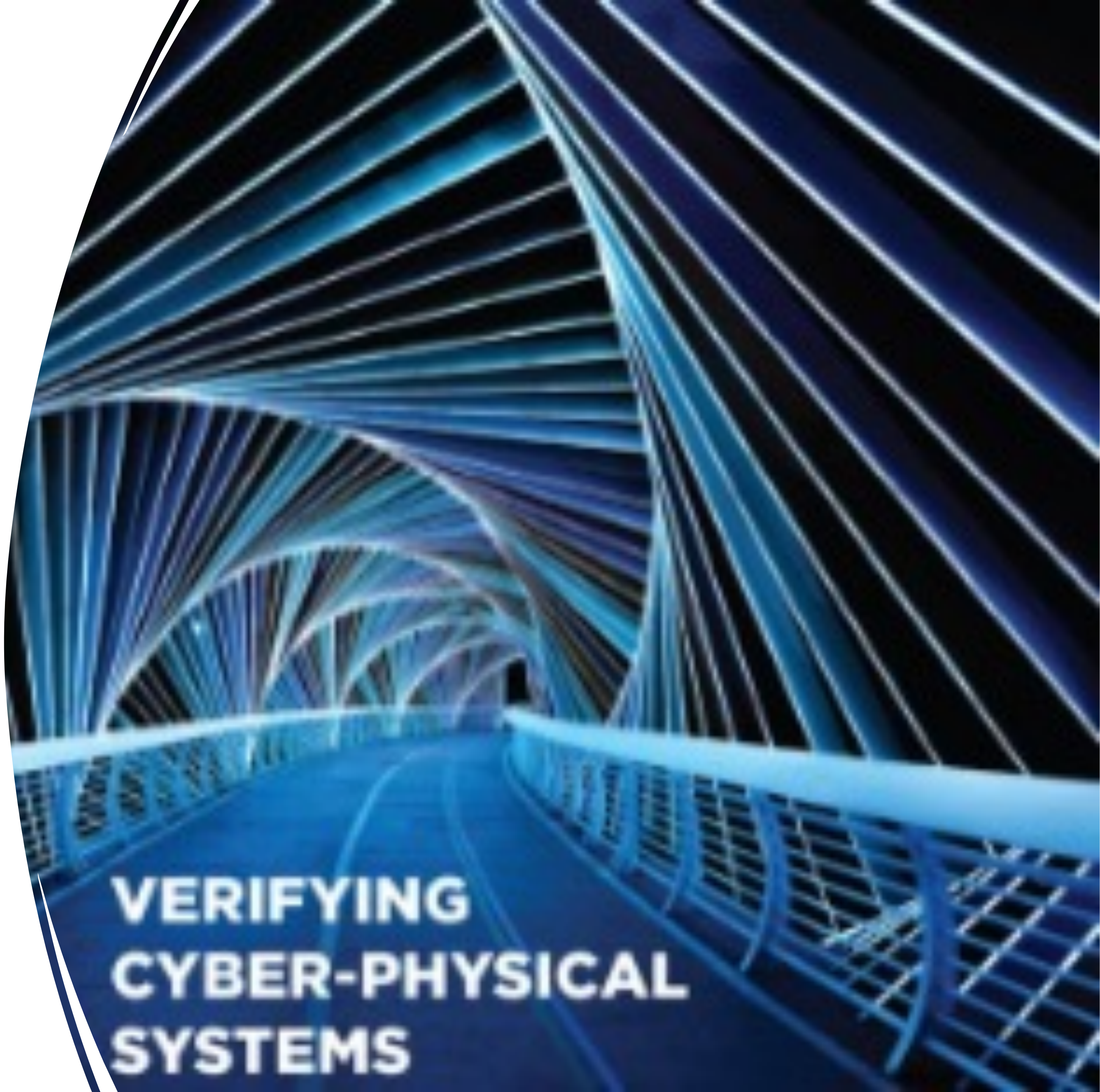
Sayan Mitra

CSL 266

mitras@illinois.edu

@Mitrasayn

# Welcome to Fall 21 edition!



What is this class about?

# INTRODUCTION

# What is verification?

**Definition.** ***Verification*** *is the action of demonstrating or proving some statement to be true by means of evidence. OED*

This class:

*some statement* = about cyber-physical systems

*evidence* = mathematical proof

# What are cyber-physical systems (CPS)?

A computer system monitoring or controlling a physical process.

- Examples: a drone for package delivery, control system for a smart electric grid, insulin pump for blood glucose control, …

The number of possible behaviors of such systems is usually *uncountably infinite*

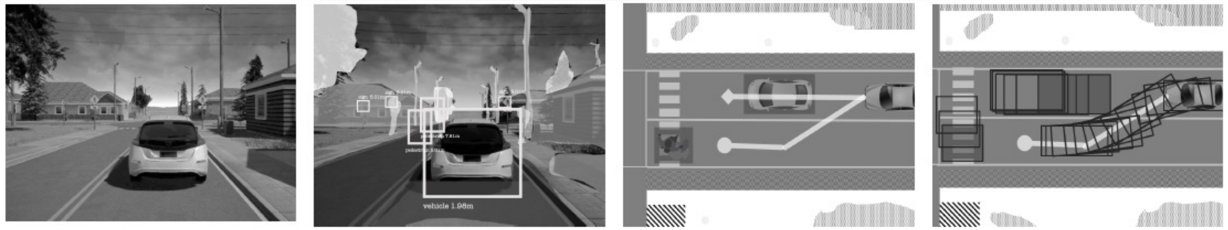
Requirements: Statements about all *behaviors*

- Drone visits waypoints while avoiding collisions
- Under all nominal conditions the vehicle stays within the lanes
- Insulin pump maintains blood glucose level to within the prescribed range

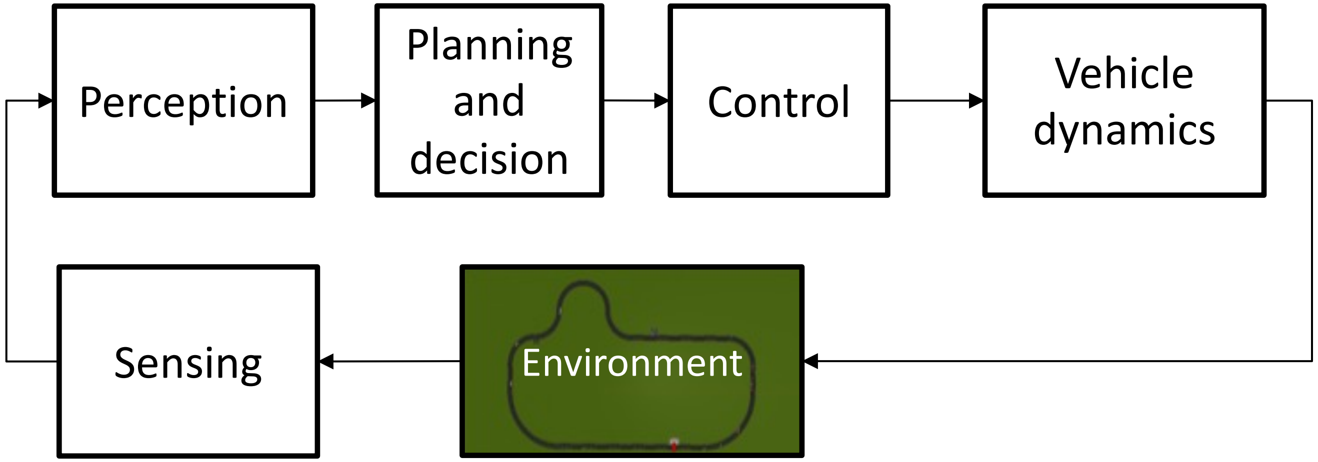
Testing: evaluates requirements on a finite number of behaviors

Verification: aims to prove requirements over all behaviors

# Autonomous vehicle: An example CPS

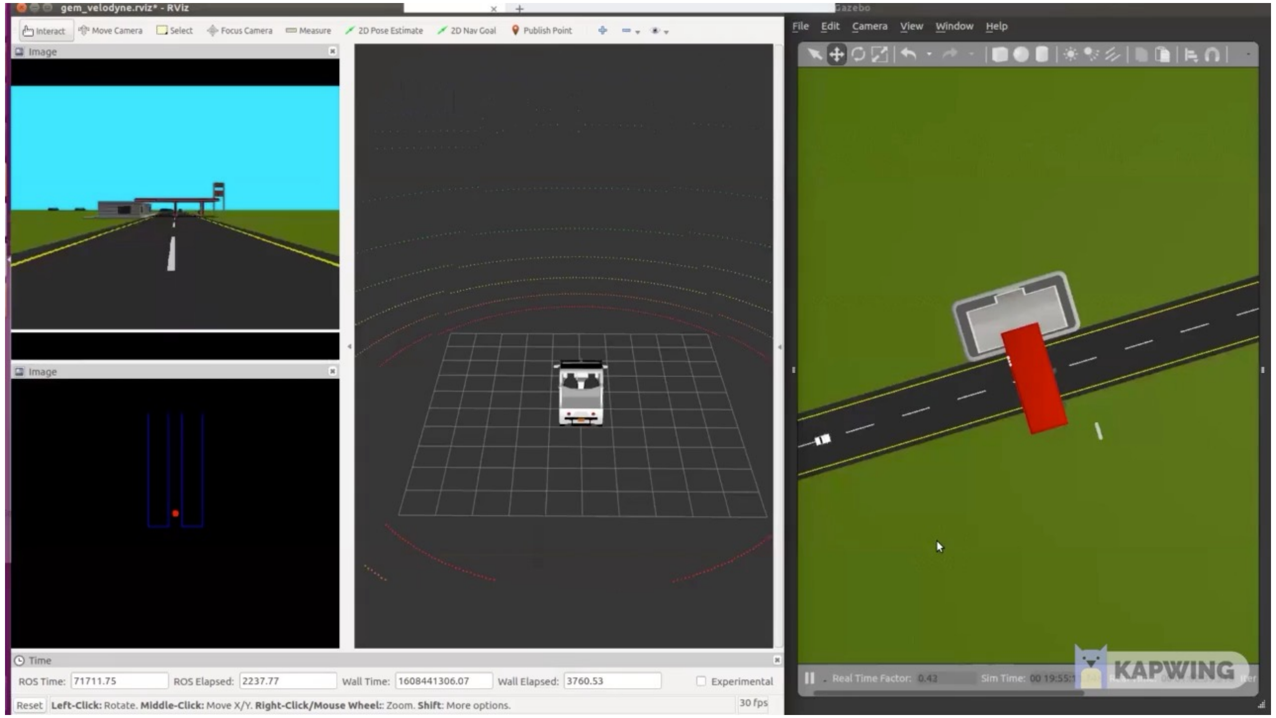
| **Sensing** | **Perception** | **Decisions and planning** | **Control** |
|---|---|---|---|
| Physics-based models of cameras, LIDAR, radar, GPS, and so on. | Programs for object tracking, scene understanding, and so on. | Programs and multi-agent models of pedestrians, cars, and so on. | Dynamical models of vehicle engine, powertrain, steering, tires, and so on. |




$env$: Lidar / vision

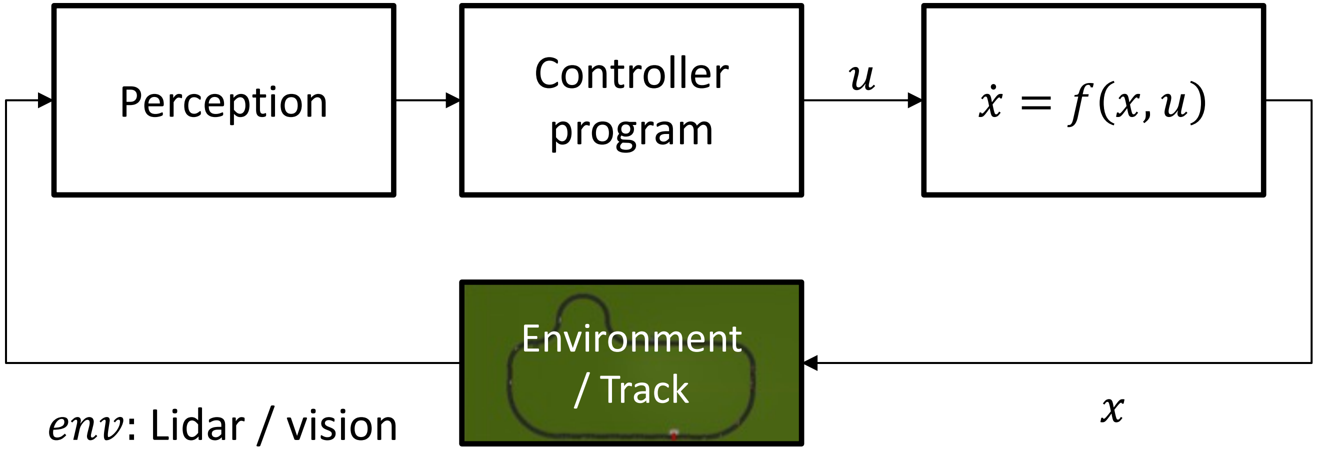
# Open problem

Simulated race car following a track with Lidar-based perception and control.

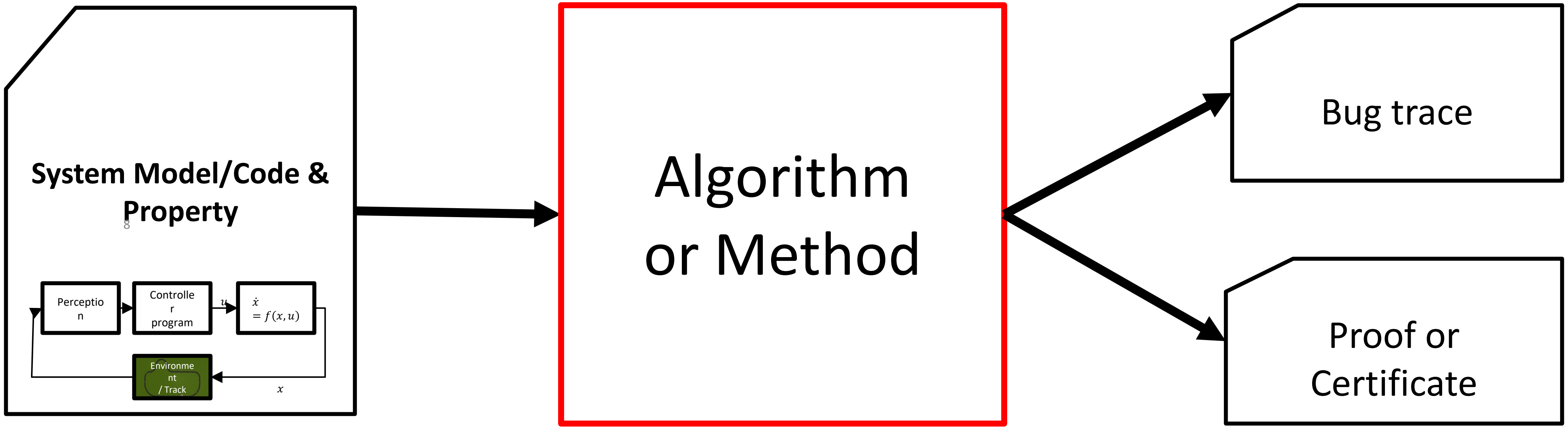



**Problem:** For a given track and initial conditions check that the *trajectory* of the car does not collide and stays in lane.

Can we check *efficiently*?
Can we *generalize* to *similar* tracks?
What should we assume about perception, accuracy of the vehicle model?
What should we assume about the execution of the controller?



# The verification problem



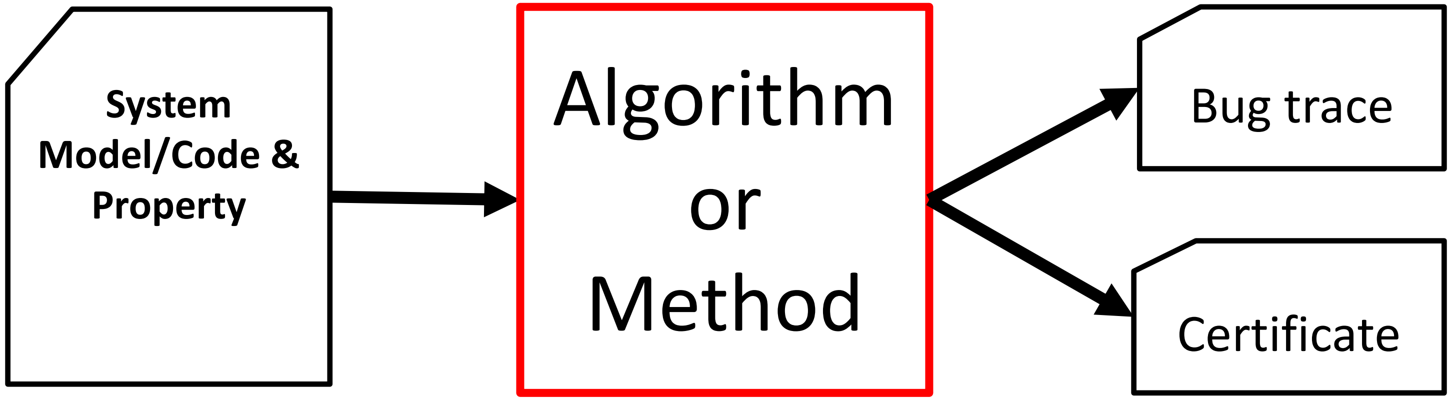

**Verification.** *The action of demonstrating or proving to be true by means of evidence; formal assertion of truth. (OED)*

# Program verification

**System.** A subroutine
`sort(int a[])`
for returning a sorted array of integers in some programming language, e.g. C

A model M for execution of programs in C

**Requirement.** Output of
`sort(int a[])`
is the sorted version of the input array `a[]`




**counterexample.** A particular input array `a` and initialization of `sort` that produces wrong output

A mathematical proof that establishes that `sort(int a[])` works for all inputs in the given model M of C

**Verifying compiler.** Checks that `sort` meets the requirement

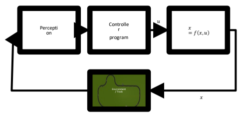
# A *cyber-physical* example

**System.** A program/system for *lane keeping control* for vehicles

Model/assumptions for executing such programs including the effects on the physical vehicle

**Requirement.** The vehicle does not go outside the lane boundaries

**System Model/Code & Property**



Algorithm or Method

Bug trace

Proof Certificate

**counterexample.** A particular environment situation (lane geometry, sensor failure, computer configuration) that makes the vehicle go outside lanes

A mathematical proof that establishes that for all *allowed* inputs and environments the vehicle stays with the lane

**Verification tool**

When can we build such a tool? How expensive is it? How well is it going to work? Under what assumptions?

Algorithm or Method

# Our goals in this course

*Write programs (tools) that prove correctness of CPS*

- *Understand limits of such programs*
- *Learn models of CPS at different levels of abstractions*

# Successes of Verification

Hardware verification now standard in EDA tools from Synopsys, Cadence, etc.

SLAM tool from MSR routinely used for verification of Device Drivers at Microsoft:

AMAZON AWS developers write proofs using CBMC and other Automated reasoning tools

Goolge runs static analysis tools on their entire codebase

Formal modeling and analysis is becoming part of certification process for avionics (e.g., ASTREE); DO-333 supplement of DO-178C identifies aspects of airworthiness certification that pertains to of software using *formal methods*

Coverity, Galois, SRI, and others

Automotive and manufacturing ... coming soon.

*"Things like even software verification, this has been the Holy Grail of computer science for many decades but now in some very key areas, for example, driver verification we're building tools that can do actual proof about the software and how it works in order to guarantee the reliability."* **Bill Gates, April 18, 2002. Keynote address at WinHec 2002**

# Intellectual vibrancy

Covers and connects some of the brightest ideas in CS and control

Vibrant research community:

Conferences: CAV, TACAS, PLDI (programming languages),

HSCC, EMSoft, ICCPS (hybrid and cyber-physical systems)

Robotics, automatic control

AI and machine learning

Turing Awards: Lamport (2014), Clarke, Sifakis & Emerson (2008), Pnueli (1997), Lampson (1992), Milner (1991), Hoare (1980), Dijkstra (1972) …

ACM Doctoral Dissertation Award: Chuchu Fan (2020) alumni of this class

Faculty and research positions: Alumni of this course are professors at Vanderbilt, UNC Chapel Hill, MIT, Kansas, Stoney Brook, and researchers at Waymo, Toyota, Boeing

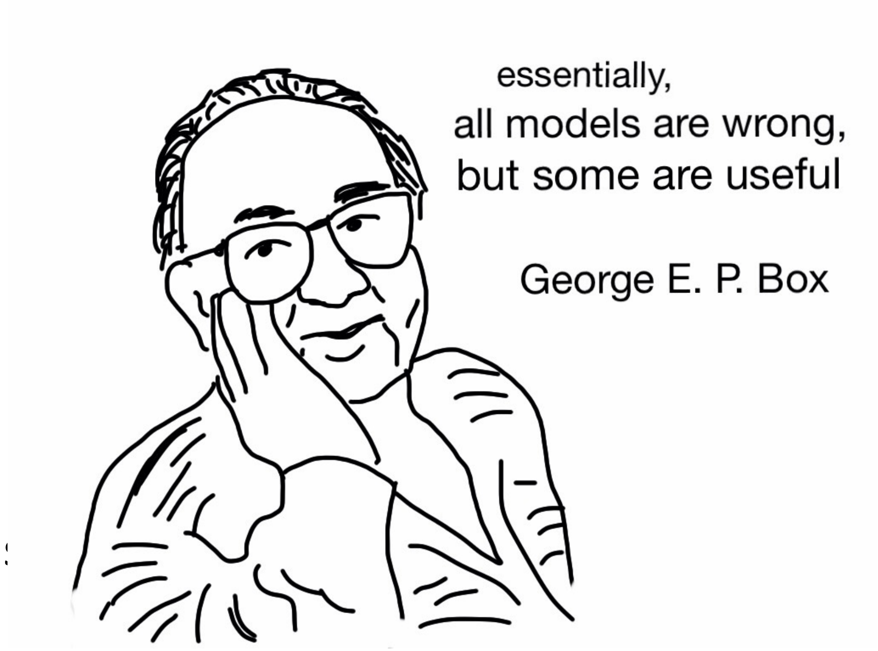
# Challenge 1: Models

To prove anything, first we have to start with assumptions

Assumptions are captures in the *models* (of cyberphysical systems)

1/3 of this class is about models

- Programs, state machines, or differential equations, block diagram:
- Discrete or continuous time, state or both -- hybrid
- Deterministic or nondeterministic or probabilistic
- Composition and interfaces, abstraction
- Modeling languages, tools

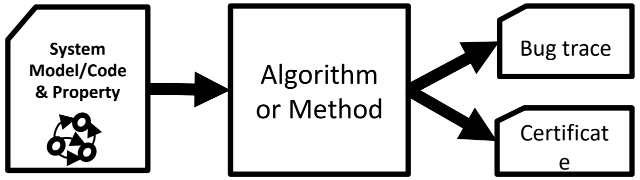



https://tribalsimplicity.com/2014/07/28/george-box-models-wrong-useful/

# Challenge 2: Scalability

Verification of hybrid automaton is *undecidable*

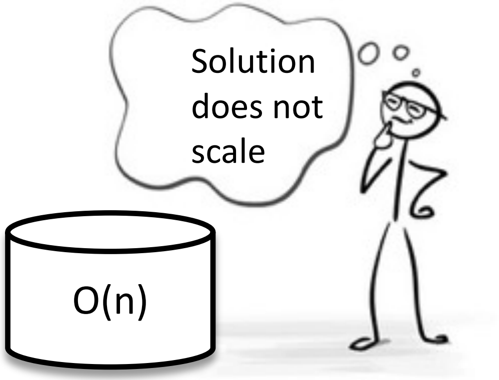
– *No one* can find the  is Algorithm of that type



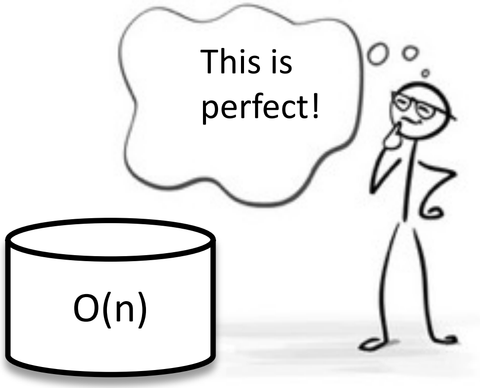

Approximate and bounded time versions of the problem can be solved algorithmically

Often the algorithms do not *scale* with the size of the model, number of agents, time horizon, etc.

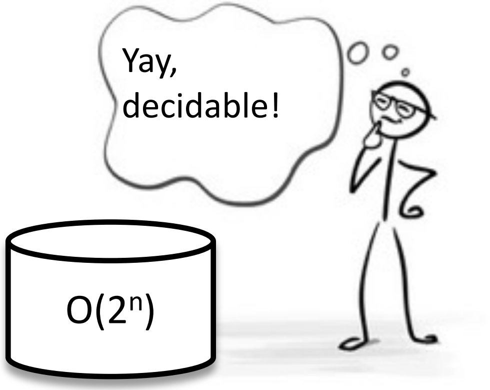
# Perspectives on scalability




data scientist



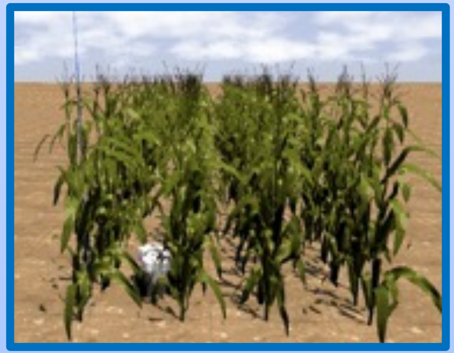

algorithmist

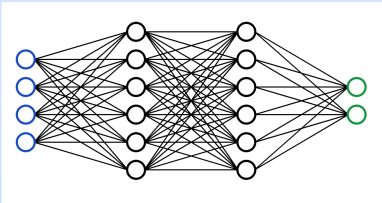



verification engineer

# Challenge 3: Perception

image





Perceived variables
heading, dist

Nnet-based
Lane detection

Environment
simulated

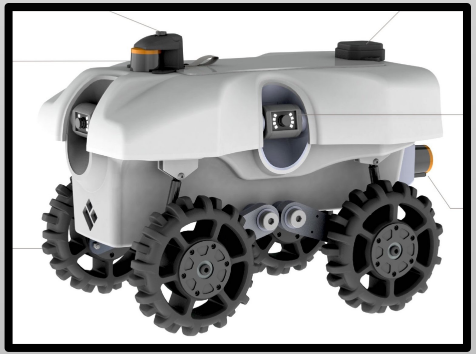
Uncontrolled variables
lighting, weather, etc.

Lateral
controller

Vehicle model

```
def control(heading, dist):
  error = heading + arctan(KP*dist, VEL)
# Calculate controller output
  ang_vel = error / CYCLE_TIME
  if ang_vel > VEL_MAX:
     ang_vel = VEL_MAX
  elif ang_vel < VEL_MIN:
     ang_vel = VEL_MIN
  return ang_vel
```

Controlled variables
angular velocity



New, underspecified, empirical

Well-understood

# Learning objectives

- Foundational connections between computer science and control theory
- Model anything
- Introduction to key concepts in formal methods and cyberphysical systems; exposure to some of the most influential ideas in CS and control theory
- Learn powerful algorithms and tools
- Jumpstart research

Invariant, barrier certificates, ranking functions, stability, self-stabilization, convergence, transition system

Programs, state machines, or differential equations, discrete or continuous state or both, Hybrid, switched, Deterministic or nondeterministic or both, composition, interfaces, abstraction, modeling languages, tools

satisfiability modulo theory, semantics, temporal logics, theorem provers, SAF solvers, ranking functions, data-driven verification, HYLAA, C2E2, SpaceEx, Flow*, Z3, …

semester-long project, feedback, presentation, hardware, software, and data resources

How the course works

# ADMINISTRIVIA

# Illinois 2021 Edition

- https://wiki.illinois.edu/wiki/pages/viewpage.action?pageId=642598908
- Lectures TR 12:30 – 1:50
- Textbook
- Homeworks: 4-5 sets. Analysis and some coding
- Project: Semester long research project, usually leads to a publication