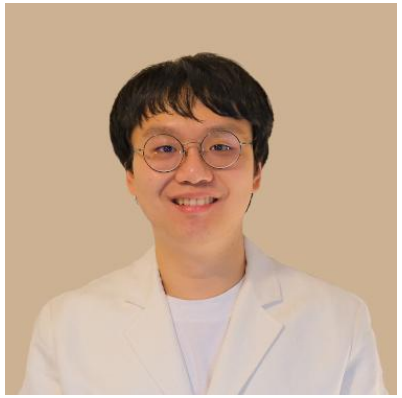
Principles of Safe Autonomy
ECE 484 Fall 2026
Lecture 2

Professor: Huan Zhang

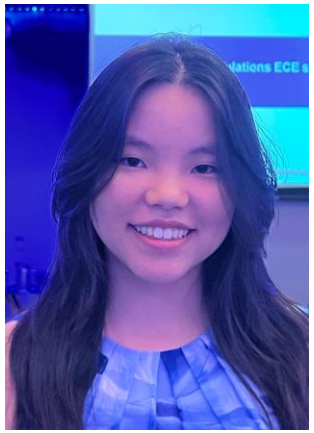
Fall 2026

https://publish.illinois.edu/safe-autonomy/

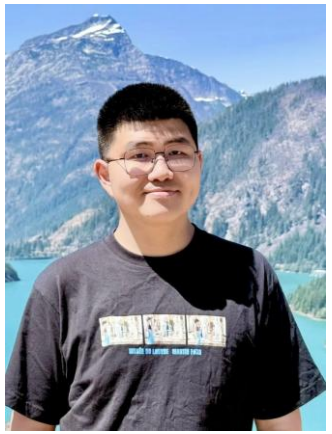
Office Hours and TA contacts



Prof. Huan Zhang (huanz)



Hanna Chen (**head TA**) (hannac4)



Yuxi Chen (yuxi5)

Junsheng Huang (jh103)

Taowei Huang (taoweih2)

Eric Ji (ericji3)

Vishesh Prasad (vprasad3)

Office Hours

TA	Day/Time	Location
Hanna	Thurs 4-5PM	ECEB 5072
Yuxi	Thurs 9:45-10:45AM	ECEB 5072
Junsheng	Mon 11AM-12PM	ECEB 5072
Taowei	Tues 10-11AM	ECEB 5072
Eric	Mon 1-2PM	ECEB 5072
Vishesh	Wed 11AM-12PM	ECEB 5072

Don’t forget the lab this Friday!

- MP0 and HW0 will be released on Friday
- Go to your registered lab session
- You can switch your lab section through the regular course registration portal (if the section you want to switch to is not full)
- Email the Head TA (Hanna Chen) if your section is full; we may help you switch if space is available
- Form your team! (3 – 4 people)
- All team members in the same lab session

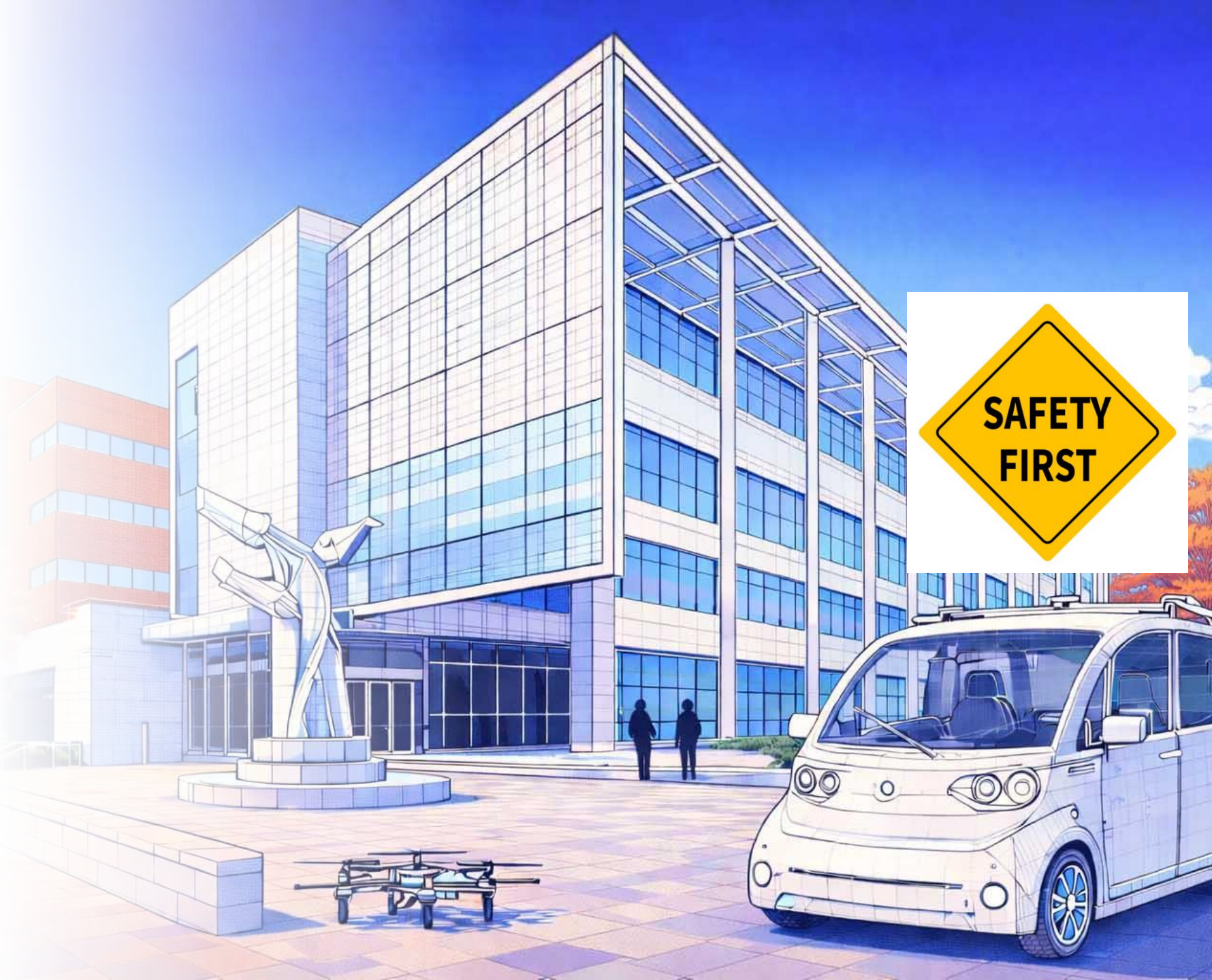
Lab Sessions

Session No.	TA
AB1 (10am-11am)	Eric Ji
AB2 (11am-12pm)	Yuxi Chen
AB3 (5pm-6pm)	Junsheng Huang
AB4 (6pm-7pm)	Hanna Chen
AB5 (7pm-8pm)	Taowei Huang

Outline

Introduction to Safety

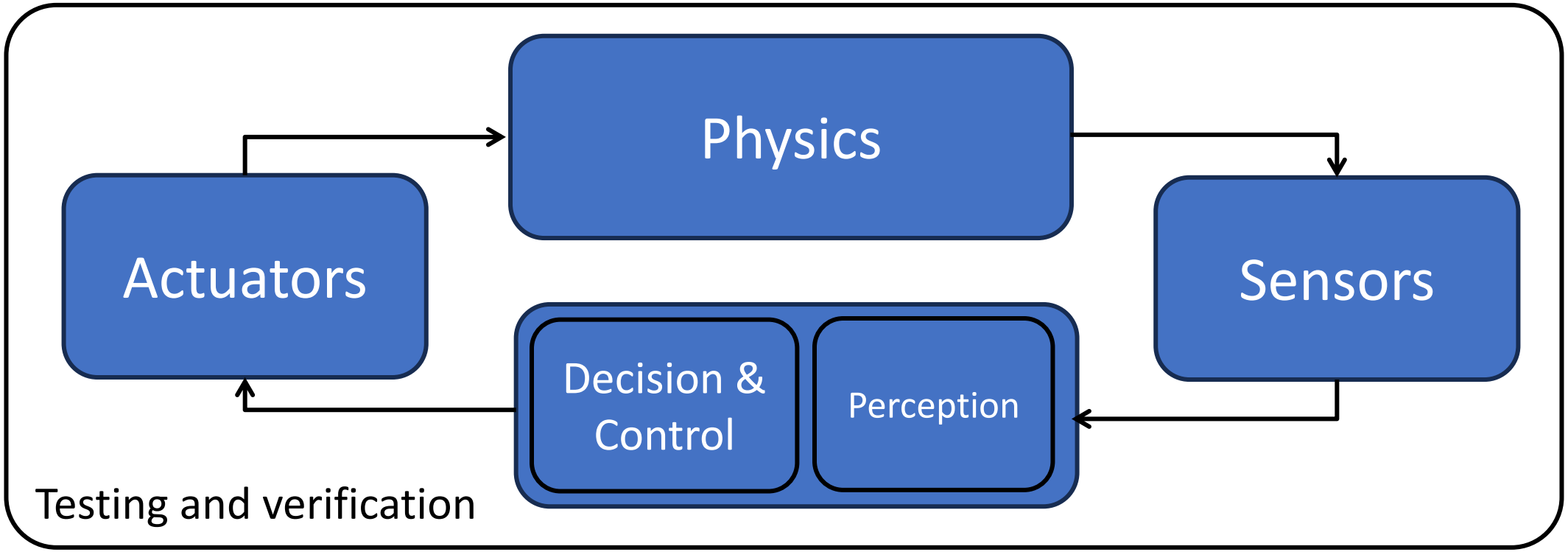
- Motivations
- Models
- Requirements
- Reachability



Principles of Safe Autonomy ECE 484

Autonomy is a frontier where perception, learning, control, and verification meet the real world and ECE 484 is where you begin to shape it.

ECE 484 develops the foundational principles behind autonomy



Example: Automatic Emergency Braking (AEB)

A car moving down a straight road has to detect any pedestrian (or another car) in front and stop before it collides.

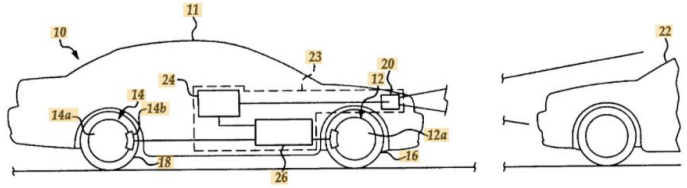


Figure 1

Non-working AEB will violate safety and reliability

www.google.com › patents

US20110168504A1 - Emergency braking system - Google ...

Jump to **Patent** citations (18) - US4053026A * 1975-12-09 1977-10-11 Nissan Motor Co., Ltd. Logic circuit for an **automatic braking** system for a motor ...

www.google.com › patents

US5170858A - Automatic braking apparatus with ultrasonic ...

An automatic braking apparatus includes: an ultrasonic wave emitter provided in a ... Info: **Patent** citations (13); Cited by (7); Legal events; Similar documents; Priority and ... US6523912B1 2003-02-25 **Autonomous emergency braking** system.

www.google.com › patents

DE102004030994A1 - Brake assistant for motor vehicles ...

B60T7/22 **Brake**-action initiating means for **automatic** initiation; for initiation not ... Info: **Patent** citations (3); Cited by (9); Legal events; Similar documents ... data from the environment sensor and then **automatically** initiates **emergency braking**.

www.google.com.pg › patents

Braking control system for vehicle - Google Patents

An **automatic emergency braking** system for a vehicle includes a forward viewing camera and a control. At least in part responsive to processing of captured ...

www.automotiveworld.com › news-releases › toyota-ip...

Toyota IP Solutions and IUPUI issue first commercial license ...

Jul 22, 2020 - ... and validation of automotive **automatic emergency braking** (AEB) ... and Director of **Patent** Licensing for Toyota Motor North America. "We are ...

insurancenewsnet.com › oarticle › patent-application-tit...

Patent Application Titled "Multiple-Stage Collision Avoidance ...

Apr 3, 2019 - No assignee for this **patent** application has been made. ... **Automatic emergency braking** systems will similarly, also, soon be required for tractor ...

Checking truthfulness of statements about reliability and safety

- A popular method for checking truth: Statistical **testing**
- "Testing can be used to show the presence of bugs, but never to show their absence!"
- --- Edsger W. Dijkstra
- Amount of testing required for autonomous systems can be prohibitive
  - Probability of a fatality caused by an **accident per one hour of human driving** is known to be 10^{-6}
  - Assume that for AV this has to be 10^{-9}
  - Data required to guarantee a probability of 10^{-9} fatality per hour of driving is proportional to its inverse, **10^9 hours, 30 billion miles**
  - Multi-agent, open system, with human interactions => cannot be simulated offline to generate data
  - Any change is software means tests have to be rerun

On a Formal Model of Safe and Scalable Self-driving Cars by *Shai Shalev-Shwartz, Shaked Shammah, Amnon Shashua, 2017 (Responsibility Sensitive Safety)*

Checking truthfulness of statements

The ultimate standard for truth: A theorem with a proof

Formal verification: The science of **proving** or disproving **truth of statements** asserting correctness of systems

Proofs are being used at scale in Amazon, Meta, Microsoft, NASA, …

“In 2017 alone the security team used deductive theorem provers or model checking tools to reason about cryptographic protocols/systems, hypervisors, boot-loaders/BIOS/firmware, garbage collectors, and network designs.” Byron Cook, Amazon

Byron Cook’s talk at FLoC 2018
https://www.youtube.com/watch?v=JfjLKBO27nw

Formal Verification vs Testing

Can we trust this “clever implementation” of the same function?

Naive implementation **?=** Clever implementation

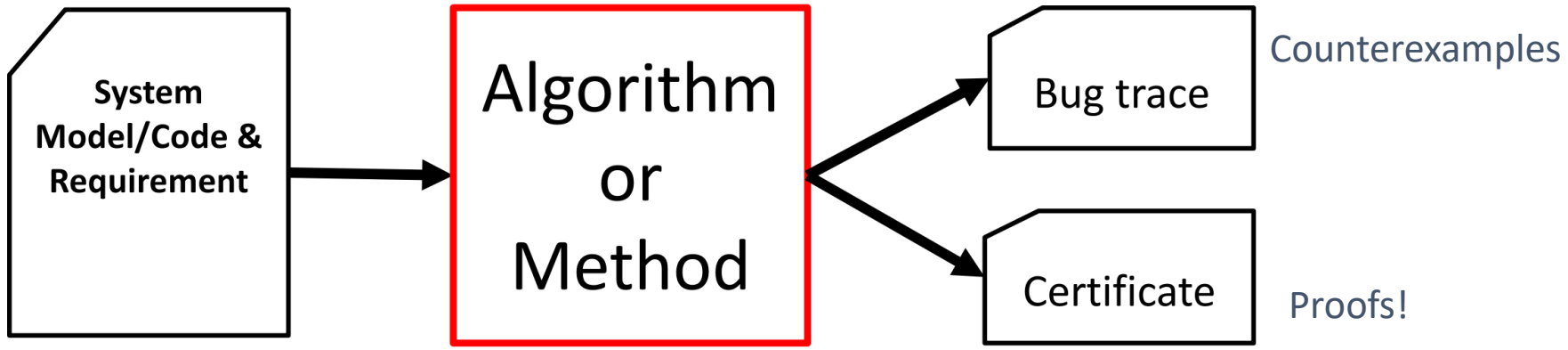
```
int popcount(uint32_t x) {
  int c = 0;
  for (int i = 0; i < 32; i++) {
    c += x & 1;
    x >>= 1;
  }
  return c;
}
```

```
int popcount (uint32_t x) {
    x = x - ((x >> 1) & 0x55555555);
    x = (x & 0x33333333) + ((x >> 2) & 0x33333333);
    x = ((x + (x >> 4) & 0xf0f0f0f) * 0x1010101) >> 24;
    return x;
}
```

Testing: evaluates requirements on a finite number of behaviors

Verification: aims to prove requirements over all behaviors

Formal verification:
model + **requirement** + algorithm to find **proofs**



Outline for this module

- Math **models**: automata, executions
- **Requirements**: statements about correctness
- **Proofs**: Reachable states, Invariants for safety guarantees (next lecture)

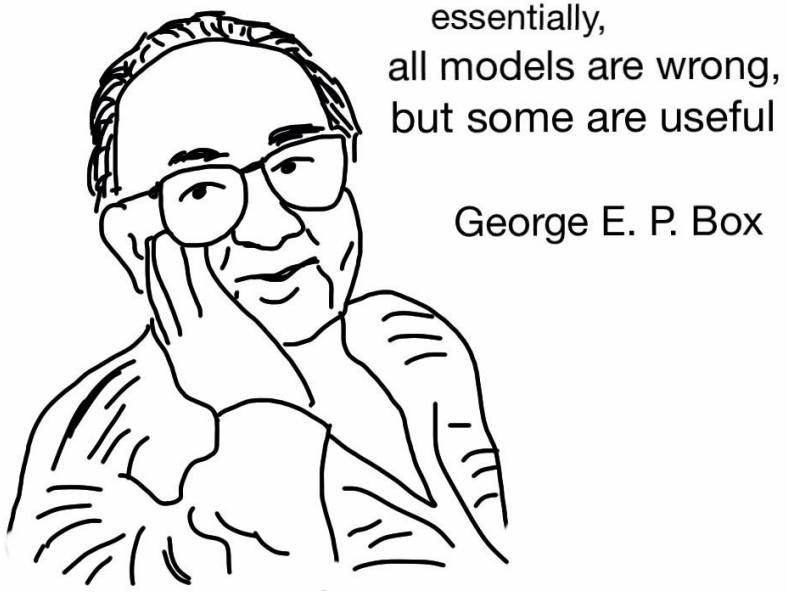
Start with: modeling

To prove anything, first we have to start with **assumptions** for the mathematical description of the system

Assumptions are captured in the *models* (of the system under study, e.g., an autonomous vehicle)

- Programs, state machines, or differential equations, block diagrams
- Discrete or continuous time, state or both -- hybrid
- Deterministic or nondeterministic or probabilistic
- Composition and interfaces, abstraction
- Deal with machine learning, deep neural networks

In this class, we will introduce some simple models (**automata/state machine**)



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

Automata or state machine: a **model** for computation

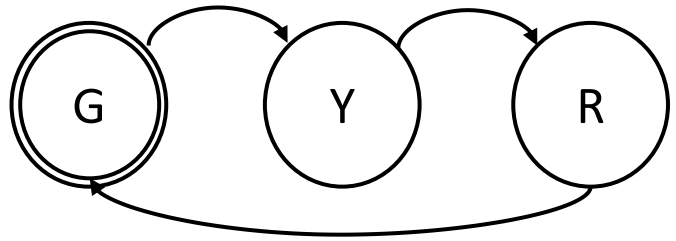
An automaton A is defined by a triple $\langle Q, Q_0, D \rangle$, where

- Q is a set of states
- $Q_0 \subseteq Q$ is a set of initial states
- $D \subseteq Q \times Q$ is a set of transitions

An execution of A is a finite or infinite sequence $q_0, q_1, \ldots$ such that $q_0 \in Q_0$ and, for every index i, $(q_i, q_{i+1}) \in D$

Example 1: Traffic light automaton

- $Q = \{G, Y, R\}$, $Q_0 = \{G\}$
- $D = \{(G, Y), (Y, R), (R, G)\}$



Execution of traffic light $G, Y, R, G, Y, R \ldots$ infinite even though finite state

Infinite State Automata: Collatz Automata example

Example 2:

- $Q = \mathbb{N} = \{1,2,3, \dots\}\ Q_0 = \{n_0\}$
- $\left(n, \frac{n}{2}\right) \in D$ if n is even; $(n, 3n+1) \in D$ if n is odd

Deterministic automaton because $|Q_0| = 1$ and each state $q \in Q$ has a unique next state

Deterministic automata have a single maximal execution (all others are its prefixes)

Execution from $n_0 = 6$ is 6, 3, 10, 5, 16, 8, 4, 2, 1, 4, 2, 1 …

Execution from $n_0 = 7$ is 7, 22, 11, 34, 17, 52, 26, 13, 40, 20, 10, 5, 16, 8, 4, 2, 1, . . .

Question: For any n_0 does the execution end in the 4-2-1 loop?

This is a well-known open problem in mathematics called the Collatz conjecture

Testing and verification can be hard for simple deterministic automata

Requirements and **Counter-examples**

Requirements define what the system must and must not do

- Example: “Car stays within speed limit”
- Autonomous car: “Ego should not collide with lead car”
- Collatz: “Every number eventually ends in the 4-2-1 cycle”

A requirement defines a set ***R*** of allowed executions

An execution α that is not in the set ***R*** is a *counter-example*

$R_{\text{eventually-1}} = \{\alpha \mid \exists k \; \alpha_k = 1\}$

An automaton A satisfies a requirement R if *all* executions of A are in R

Whether the Collatz automaton satisfies the requirement $R_{\text{eventually-1}}$ for all initial conditions remains an open problem, although no counter-example has been found up to 2^{70}
This is an example of a verification problem

Verification problem

Verification problem: Given an automaton A and a requirement R, check whether all executions of A satisfy R or find a counter-example

Testing or checking individual executions can help find counter-examples but cannot show that there is no counter-example

Verification can be hard because

- |Q| is finite but large and testing may require visiting all the states
- |Q| is small but the number of executions is very large
- |Q| may be infinite (e.g., Collatz, with Q = {1, 2, 3, …}) and D may be nondeterministic --- typical for autonomous systems

Example: Automatic Emergency Braking (AEB)

Car must brake to maintain safe gap with lead vehicle/pedestrian

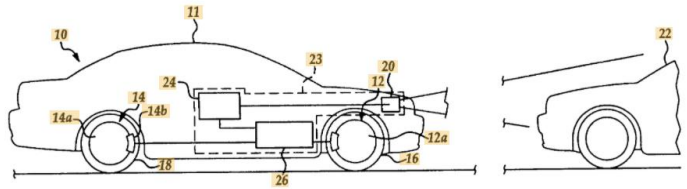


Figure 1

There is no standard for checking correctness of AEB systems

Future: every time an AEB engineer commits code to GitHub, a theorem **proves** that A satisfies R_{gap} under stated assumptions — or a counter-example execution is found

www.google.com › patents
US20110168504A1 - Emergency braking system - Google ...
Jump to **Patent** citations (18) - US4053026A * 1975-12-09 1977-10-11 Nissan Motor Co., Ltd. Logic circuit for an **automatic braking** system for a motor ...

www.google.com › patents
US5170858A - Automatic braking apparatus with ultrasonic ...
An automatic braking apparatus includes: an ultrasonic wave emitter provided in a ... Info: **Patent** citations (13); Cited by (7); Legal events; Similar documents; Priority and ... US6523912B1 2003-02-25 **Autonomous emergency braking** system.

www.google.com › patents
DE102004030994A1 - Brake assistant for motor vehicles ...
B60T7/22 **Brake**-action initiating means for **automatic** initiation; for initiation not ... Info: **Patent** citations (3); Cited by (9); Legal events; Similar documents ... data from the environment sensor and then **automatically** initiates **emergency braking**.

www.google.com.pg › patents
Braking control system for vehicle - Google Patents
An **automatic emergency braking** system for a vehicle includes a forward viewing camera and a control. At least in part responsive to processing of captured ...

www.automotiveworld.com › news-releases › toyota-ip...
Toyota IP Solutions and IUPUI issue first commercial license ...
Jul 22, 2020 - ... and validation of automotive **automatic emergency braking** (AEB) ... and Director of **Patent** Licensing for Toyota Motor North America. "We are ...

insurancenewsnet.com › oarticle › patent-application-tit...
Patent Application Titled "Multiple-Stage Collision Avoidance ...
Apr 3, 2019 - No assignee for this **patent** application has been made. ... **Automatic emergency braking** systems will similarly, also, soon be required for tractor ...

Automaton **model** of AEB

Automaton $A = \langle Q, Q_0, D \rangle$

- Q: $[x_1, x_2, v_1] \in \mathbb{R}^3$
- $Q_0 = \{[x_1 = x_{10}, x_2 = x_{20}, v_1 = v_{10}]\}$
- $D \subseteq Q \times Q$ written as a **program**:

  If $x_2 - x_1 \leq d_s$

  $v_1 \in [v_1 - a_{1b}, v_1 - a_{2b}]$ (break)

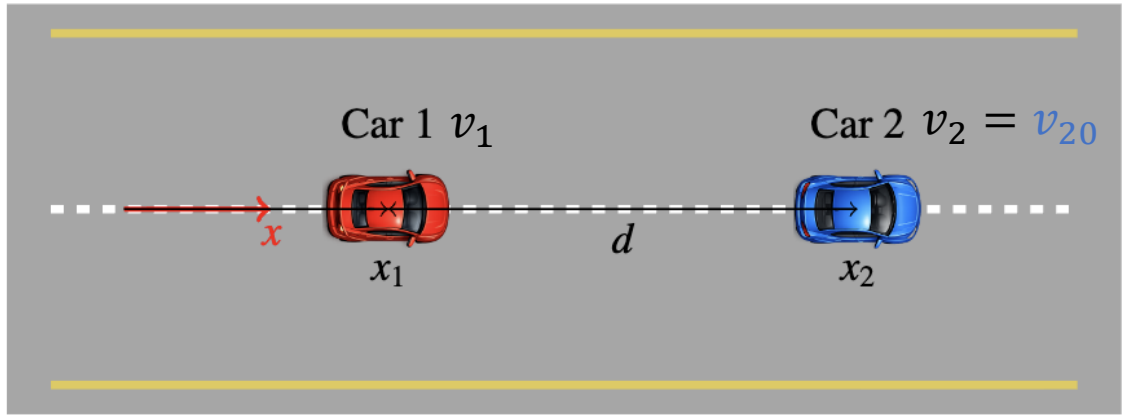
  else

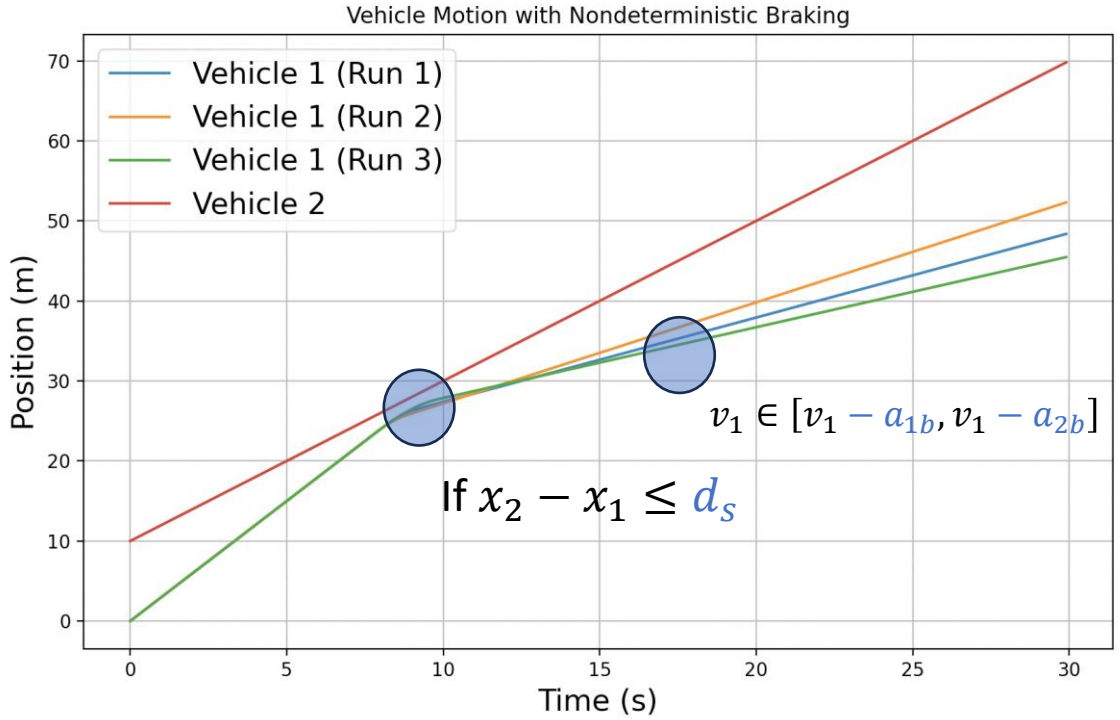
  $v_1 = v_1$ (unchanged, for clarity)

  $x_2 = x_2 + v_{20}$

  $x_1 = x_1 + v_1$

  $0 < a_{1b} \leq a_{2b}$





Automaton model of AEB

Automaton $A = \langle Q, Q_0, D \rangle$

- $Q: \mathbb{R}^3; \boldsymbol{q} \in Q\ \boldsymbol{q}.x_1, \boldsymbol{q}.x_2, \boldsymbol{q}.v_1 \in \mathbb{R}$
- $Q_0 = \{\boldsymbol{q} \mid [\boldsymbol{q}.x_1 = x_{10}, \boldsymbol{q}.x_2 = x_{20}, \boldsymbol{q}.v_1 = v_{10}]\}$
- Writing the program as state transactions:
- $(\boldsymbol{q}, \boldsymbol{q}') \in D$ iff

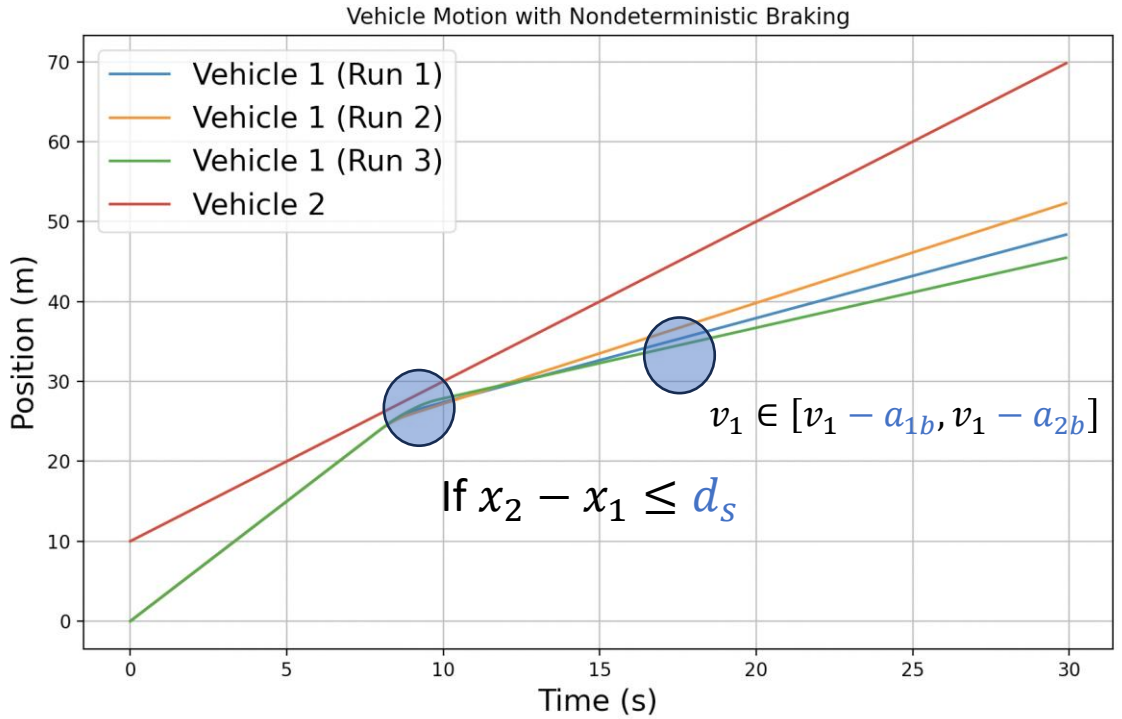
  If $\boldsymbol{q}.x_2 - \boldsymbol{q}.x_1 \leq d_s$

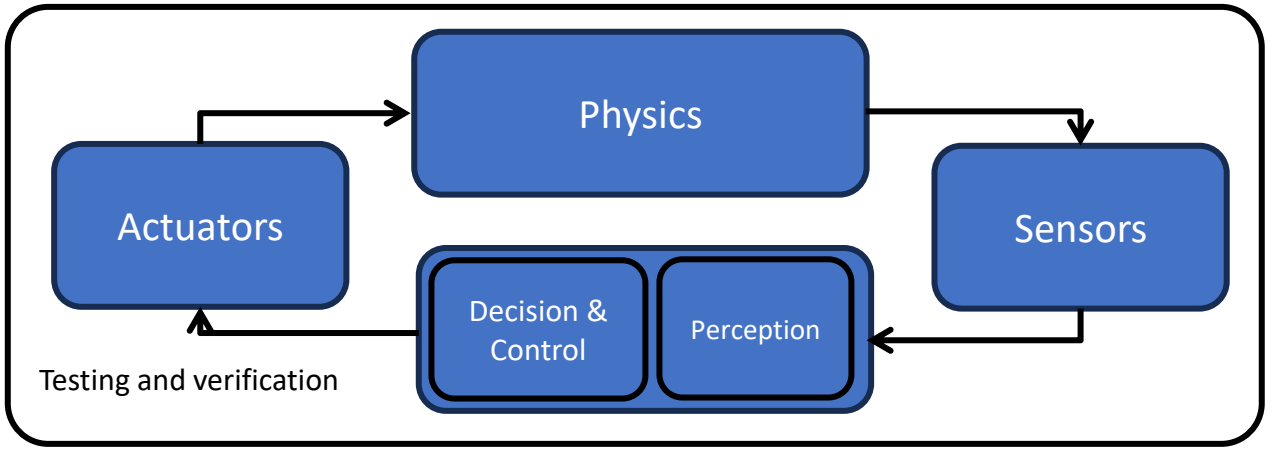
  $\boldsymbol{q}'.v_1 \in [\boldsymbol{q}.v_1 - a_{1b}, \boldsymbol{q}.v_1 - a_{2b}]$

  else $\boldsymbol{q}'.v_1 = \boldsymbol{q}.v_1$

  $\boldsymbol{q}'.x_2 = \boldsymbol{q}.x_2 + v_{20}$

  $\boldsymbol{q}'.x_1 = \boldsymbol{q}.x_1 + \boldsymbol{q}'.v_1$





What is missing in the AEB model?

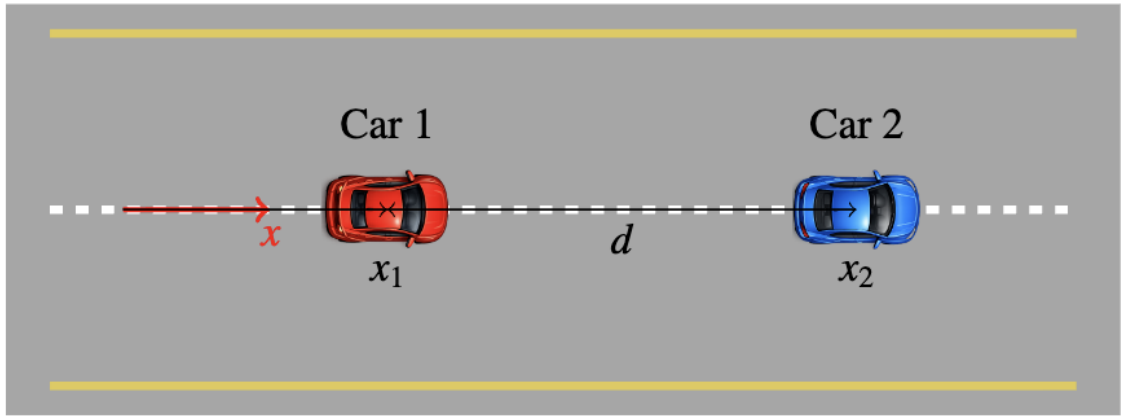
If $x_2 - x_1 \leq d_s$

$v_1 \in [v_1 - a_{1b}, v_1 - a_{2b}]$

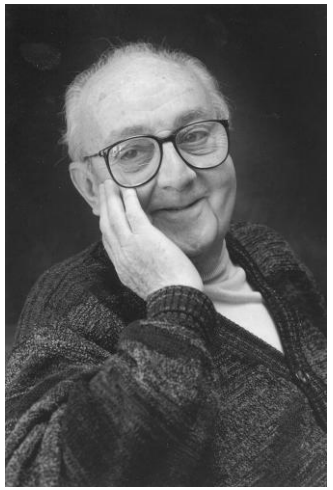
else $v_1 = v_1$

$x_2 = x_2 + v_{20}$

$x_1 = x_1 + v_1$



- Acceleration, friction in dynamics
- Uncertainty in sensing
- Uncertainty in lead vehicle behavior
- Timing of execution of control loop



"All models are wrong, but some are useful." — George E. P. Box

Safety and **liveness** requirements

$R_{gap} = \{\alpha \mid \forall i\ \alpha_i.x_2 > \alpha_i.x_1\}$ positive gap $U_{gap} = \{\boldsymbol{q} | \boldsymbol{q}.x_2 - \boldsymbol{q}.x_1 \leq 0\}$

$R_{sp\text{-}lim} = \{\alpha \mid \forall i\ \alpha_i.v_1 \leq 70\}$ speed limit $U_{sp\text{-}lim} = \{\boldsymbol{q} | \boldsymbol{q}.v_1 > 70\}$

$R_{catch\text{-}up} = \{\alpha \mid \exists i\ \ 2 > \alpha_i.x_2 - \alpha_i.x_1 > 1\}$ catch eventually

A safety requirement says that ***every*** state along *every* execution should stay in **safe states**

Equivalently, ***no*** execution of A ever reaches an **unsafe state**

R_{gap} and $R_{sp\text{-}lim}$ are safety requirements with U_{gap} and $U_{sp\text{-}lim}$ as the unsafe sets

$R_{catch\text{-}up}$ is not a safety requirement; it is an example of a liveness / progress requirement

A liveness requirement says that along every execution **eventually** some good state is reached

Safety verification: Finite State Automata

Safety verification problem: Given an automaton A and an unsafe set U, check whether there exists any execution α of A that reaches U

Counter-examples of safety are finite executions ending in U

For finite automata, safety verification can be **solved using depth first search (DFS)** from Q_0

- Consider $\langle Q, Q_0, D \rangle$ as a directed graph with $\langle Q, D \rangle$
- DFS visits every state reachable from Q_0
- If none of the reachable states is in U, there is no counter-example
- Absence of a counter-example **proves** that the automaton is safe

In practice, |Q| is often astronomically large (state-space explosion), so even a linear-time search may not scale

Safety verification and Reachability: Infinite State Spaces

A state $q \in Q$ is reachable if there exists an execution α such that $\alpha_i = q$ for some index i

$Reach_A(Q_0) \subseteq Q$ the set of reachable states of *A* from Q_0

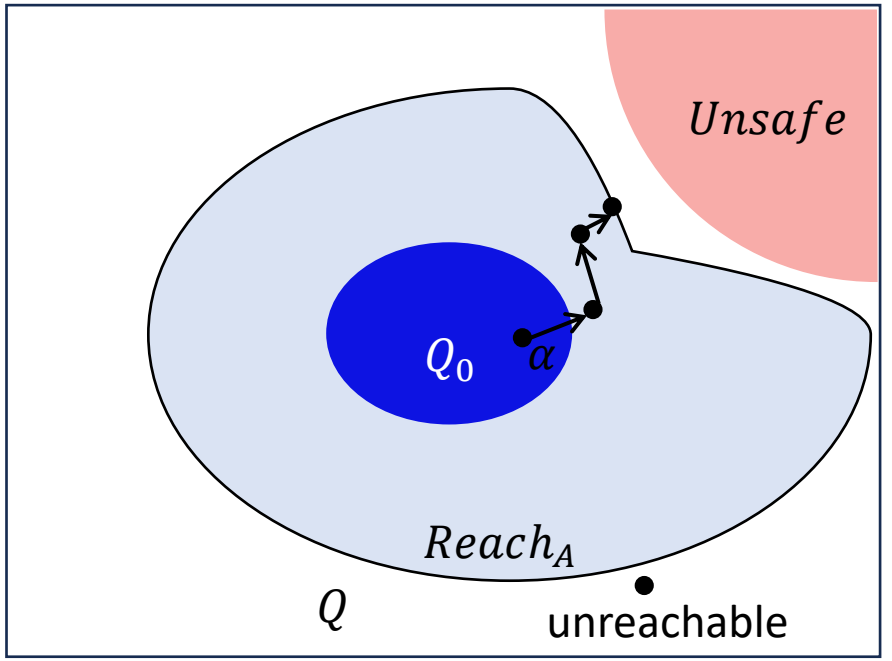
Safety verification problem is equivalent to checking $Reach_A \cap U = \emptyset$?

That is, if we can compute $Reach_A$ then we can verify safety

For finite-state systems, DFS computes $Reach_A(Q_0)$

For infinite state systems, we need:

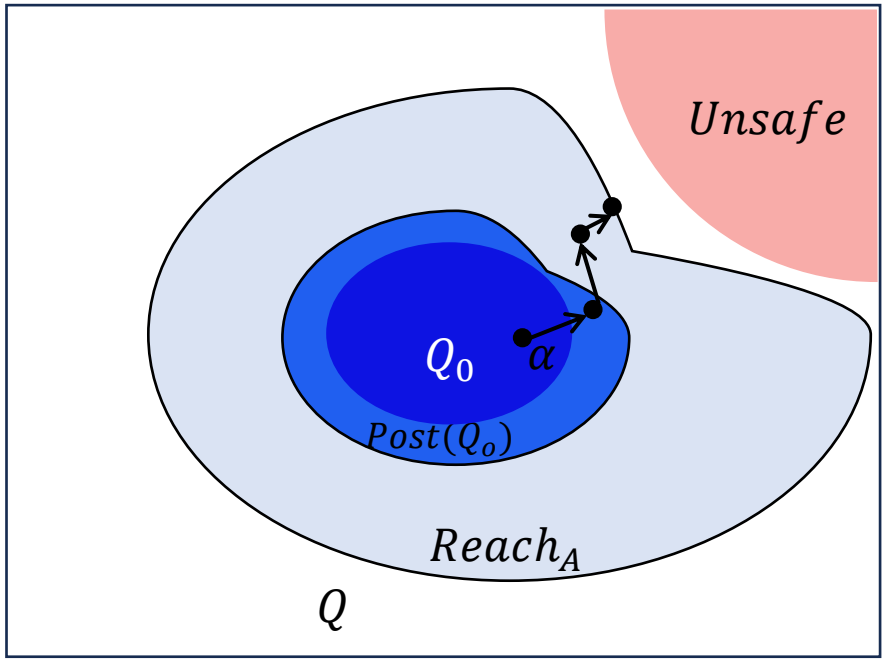
- Representation of infinite sets of states
- Iteratively computing $Reach_A$



Computing reachable sets and over-approximations

Define $Post(R) = \{q' \mid \exists\, q \in R, (q, q') \in D\}$ that gives all the states that can be reached in one step from the set of states R

- For a deterministic system $Post(\{q\}) = \{q'\}$ for $(q, q') \in D$
- For any R, $Post(R) = \cup_{q \in R} Post(\{q\})$
- $Post(Q_0) = \{q \mid \exists\, q_0 \in Q_0, (q_0, q) \in D\}$ states reachable in 1 step from Q_0
- $Post\big(Post(Q_0)\big) = ?$



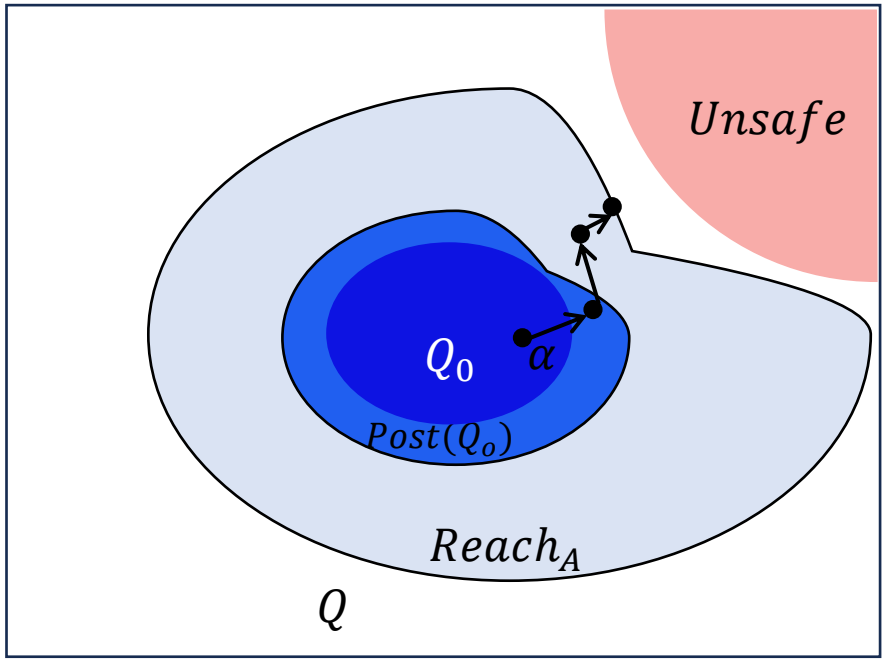
Exercise. Show that Post is monotonic, i.e., if $S_1 \subseteq S_2$ then $Post(S_1) \subseteq Post(S_2)$.

Exercise. Show that the set of states reachable in exactly k steps is $Post^k(Q_0)$.

Computing reachable sets and over-approximations

Define $Post(R) = \{q' \mid \exists\, q \in R, (q, q') \in D\}$ that gives all the states that can be reached in one step from the set of states R

- For a deterministic system $Post(\{q\}) = \{q'\}$ for $(q, q') \in D$
- For any R, $Post(R) = \cup_{q \in R} Post(\{q\})$
- $Post(Q_0) = \{q \mid \exists\, q_0 \in Q_0, (q_0, q) \in D\}$ states reachable in 1 step from Q_0
- $Post\big(Post(Q_0)\big) = ?$



Infinite sets & nondeterministic $Post(R)$ requires some representation of **sets**

Example:

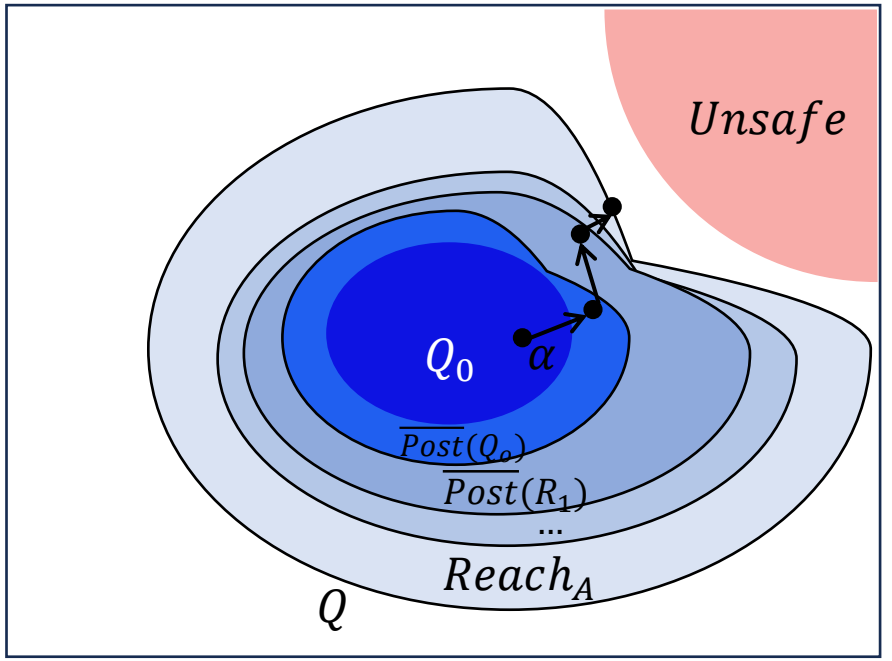
- $Q = [x_1 : \mathbb{R}]$, $D : x_1 = x_1 + v_1$ then $R = [a, b]$, $Post(R) = [a + v_1, b + v_1]$

Generally, for nonconvex R or nonlinear D exact $Post(R)$ may be infeasible

We use over-approximation $\overline{Post}(R)$ such that $Post(R) \subseteq \overline{Post}(R)$

Reachable sets and over-approximations

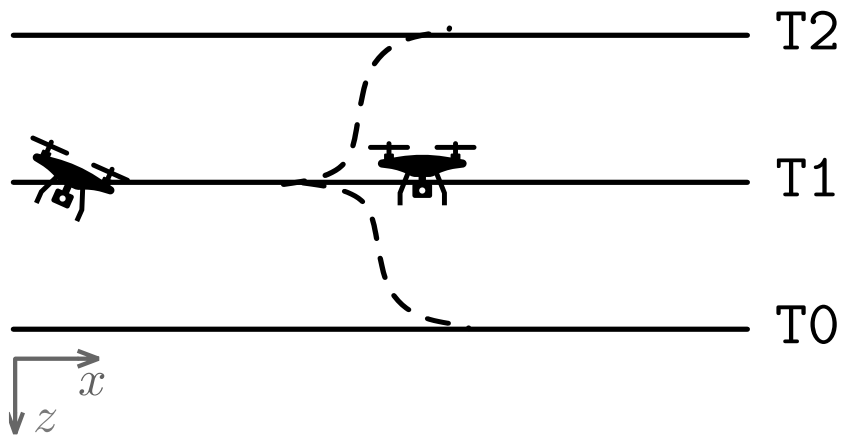
Reachability($A = \langle Q, Q_0, D \rangle$)
$R_0 = Q_0$
$i = 0$
do
  $R_{i+1} = \overline{Post}\,(R_i) \cup R_i$
  $i = i + 1$
Until $R_i = R_{i-1}$
Return R_i

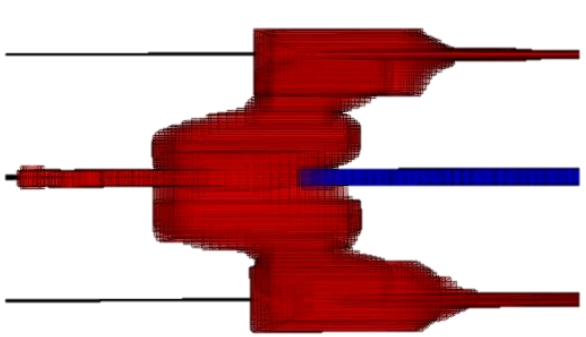


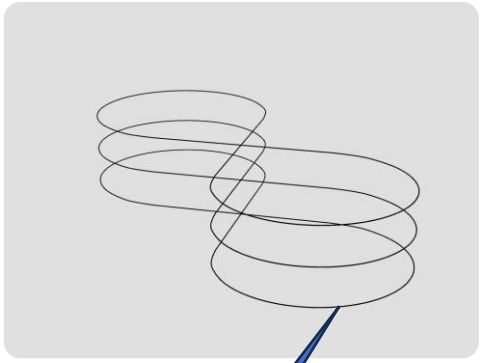
Exercise. Show that if this algorithm terminates and returns R then $Reach_A(Q_0) \subseteq R$, i.e.,
R **over-approximates** the reachable set of A. (denoted as $\overline{Reach_A}(Q_0)$)
$R \cap \text{Unsafe} = \emptyset$ proves safety, but $R \cap \text{Unsafe} \neq \emptyset$ does **not** imply that there is a real counterexample (sound, but not complete)

Verse: Python library for reachability analysis (MP0)

```
class Mode(Enum):
  Normal = auto()
  Up = auto()
  …
class Track(Enum):
  T0 = auto()
  T1 = auto()
  …
class State:
  x: float
  y: float
  …
  mode: Mode
  track: Track
```







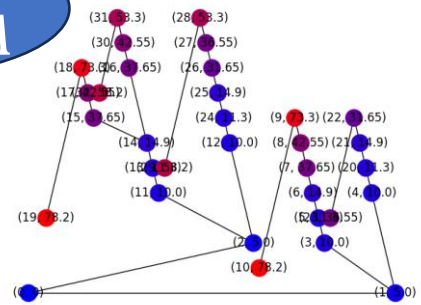
Q

D

```
def decisionLogic(ego: State, others: List[State], map):
  if ego.mode == Normal:
    if any(isClose(ego, other) for other in others):
      if map.exist(ego.track, ego.mode, Up):
        next.mode = Up
        next.track = map.h(ego.track, ego.mode, Up)
      if map.exist(ego.track, ego.mode, Down):
        next.mode = Down
        …
```

```
q1 = QuadrotorAgent(”q1", …)
q1.set_initial([…], (Mode.Normal, Track.T1))
scenario.add_agent(q1)
q2 = …
scenario.set_map(M5())
scenario.simulate(…)
scenario.verify(…)
```

$Reach_A$



$Unsafe$

```
assert not any(isVeryClose(ego, other) for other in others), "Separation"
```

Summary

- Testing vs Verification
- Automaton is a model that defines system behaviors
- Requirements define desired or correct behaviors
- Verification proves that automaton satisfies requirements or finds counter-examples
- Safety requirements say that *no execution ever* hits unsafe states
- Reachable sets and over-approximations