Lecture 2: Safety I

Professor Katie Driggs-Campbell

January 17, 2024

ECE484: Principles of Safe Autonomy

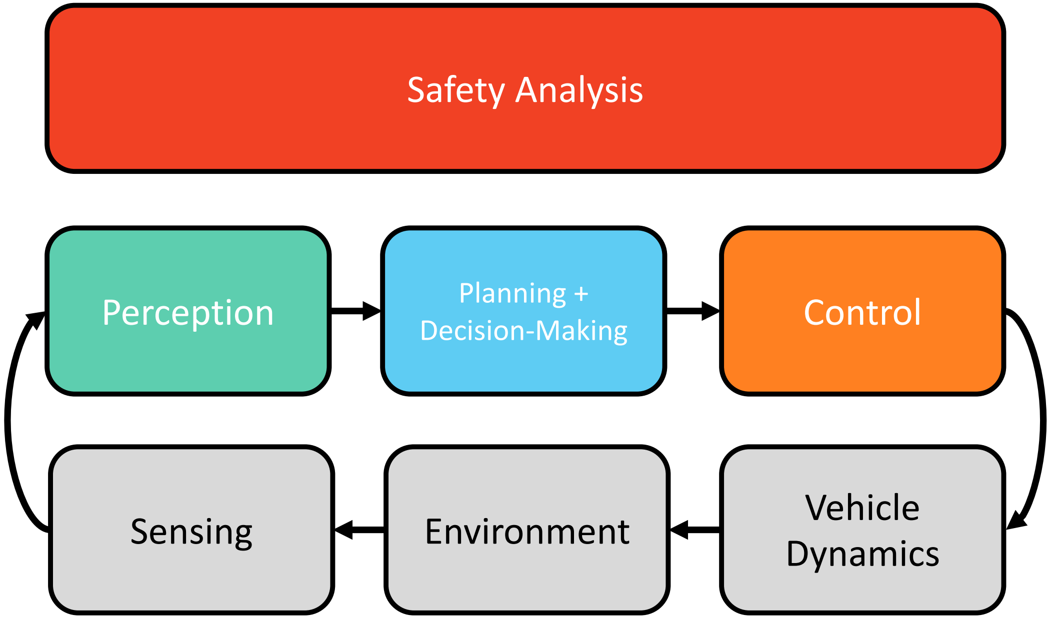
Administrivia

- Re course registration: Please sign up for lab sections!
- Canvas and campuswire setup – let me know if you do not have access
- We will have pop quizzes throughout the semester and one exam (4/18)

Today's Lecture

- Quick background on models
- Introducing automata and safety verification

Autonomous GEM Vehicle



How to assess safety?

1. Create a *model* of the autonomous system
   - What are the inputs and outputs to the system?
   - What are the expectations on behaviors?
   - No model is perfect – some models are useful!
   - What are the implicit and explicit biases in your system?

How to assess safety?

1. Create a *model* of the autonomous system
   - What are the inputs and outputs to the system?
   - What are the expectations on behaviors?
   - No model is perfect – some models are useful!
   - What are the implicit and explicit biases in your system?
2. Identify the *requirements* and *assumptions*
   - What parts of the model are available for observation/analysis?

How to assess safety?

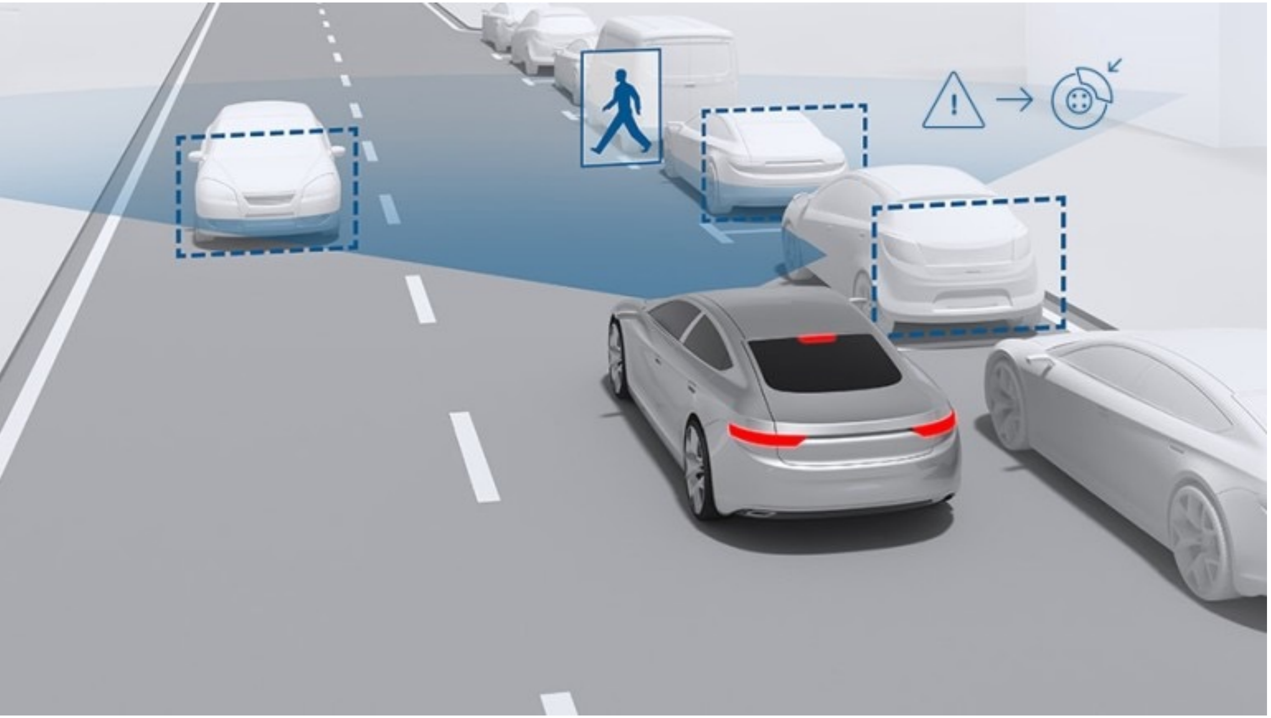
1. Create a *model* of the autonomous system
   - What are the inputs and outputs to the system?
   - What are the expectations on behaviors?
   - No model is perfect – some models are useful!
   - What are the implicit and explicit biases in your system?
2. Identify the *requirements* and *assumptions*
   - What parts of the model are available for observation/analysis?
3. Analyze model to show that it meets the requirements under the assumptions

What are some safety requirements for AVs? What are your design considerations?

- don't collide w/ ped that step in front of AV
- stay within lanes → within reason
- redundancy + back up safety
  + partial failures (system + hardware)
- stay under speed limit
- yield to emergency vehicles
- smoothness + comfort (mechanics + people)
- enforcing internal safety

- adverse weather
- maintaining speed + distance (ACC)

Emergency Braking for Pedestrians




LiveLeak

Simple State Machines and Automaton

- Definition: A state machine or an automaton is defined as: A

① a set of states Q
② a set of start/initial states $Q_0 \subseteq Q$
③ a set of transitions $D \subseteq Q \times Q$

→ nondeterministic
- from some state, A can go to multiple states

Cruise Control Example

ex)

$Q = \{C, B, A\}$, $Q_0 = \{C\}$

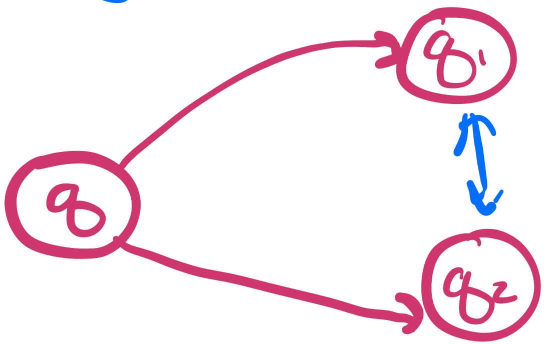
$D = \{\langle C, B \rangle, \langle B, C \rangle, \langle C, A \rangle, \ldots\}$

Deterministic Automaton

$|Q_0| = 1$ and $\forall q \in Q$, $q_1, q_2 \in Q$

if $\langle q, q_1 \rangle \in D$ and $\langle q, q_2 \rangle \in D$

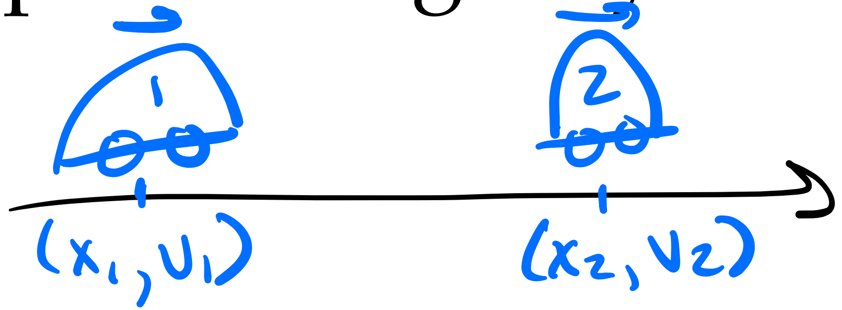
then $q_1 = q_2$



X not okay

here, $D: Q \rightarrow Q$ is a transition fn

Example: Emergency Braking System (1)



$Q = \mathbb{R}^4$

$Q_0 = \{ \overset{x_1[0]}{x_{10}}, v_{10}, x_{20}, v_{20} \}$

$x_{20} > x_{10} > 0$

$D \subseteq \mathbb{R}^4 \times \mathbb{R}^4$

↳ described by physical model, or some program

Example: Emergency Braking System (2)

if $x_2 - x_1 < d_s$

$\quad v_1 := \max(0, v_1 - a_b)$ // brake!

else $v_1 := v_1$

$x_1 := x_1 + v_1$

$x_2 := x_2 + v_2$

Executions and Behaviors

- Definition: an execution is a particular behavior or trajectory of an automaton

$\alpha = q_0 q_1 q_2 \ldots$

such that

① $q_0 \in Q$

② $(q_i, q_{i+1}) \in D \;\; \forall i$

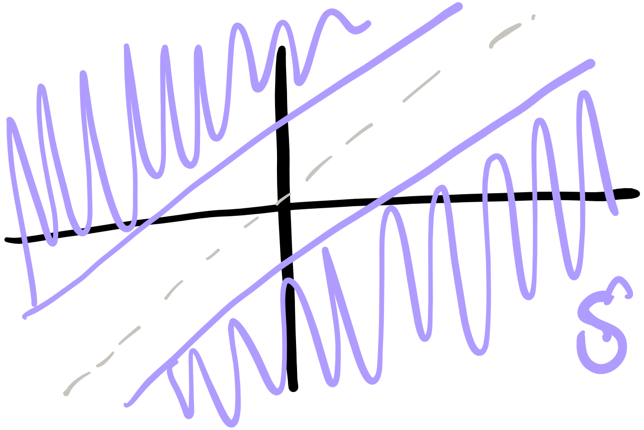
→ nondeterministic A have many executions

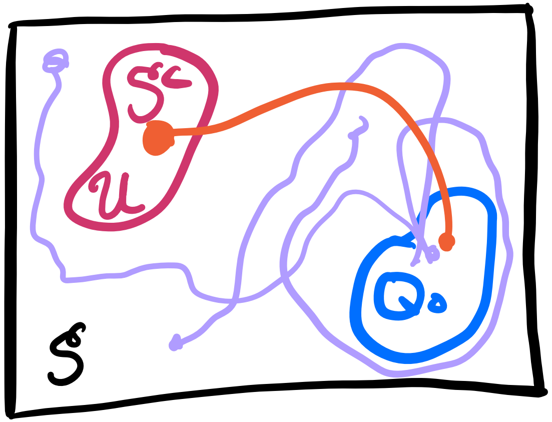
Safety Requirements

We want to express our safety requirements as:

1. A formula involving state variables $x_2 - x_1 > \tau$
2. A subset of Q

$S \subseteq Q = \mathbb{R}^4 = \{(x_1, v_1, x_2, v_2) \mid x_2 - x_1 \geq \tau\}$





The Safety Verification Problem

Reachability and the Post operator

Summary

- Start thinking about forming your team and decide project track
  - Sign-up to be member of IRL if interested in hardware projects
- All models are wrong, but some are useful!
- Think about how to define your safety requirements formally
- Automata provide us simple models for safety verification
- Next time: inductive invariants to prove safety for simple programs!