Lecture 3: Safety II

Professor Katie Driggs-Campbell

January 23, 2024

ECE484: Principles of Safe Autonomy

Administrivia

- Schedule is now online
  - Slides are posted before (blank) and after lecture (filled) 🤞
  - TBDs will likely be guest lectures
- Office Hours and HW party info posted on website
  - No OH today!
- Lab starts this week – will introduce MP0
  - Attendance may be taken!
- If you have DRES accommodations, please send me your letter

Example: Emergency Braking System

$q \in Q$, Q_0

$\langle q, q' \rangle \in D$

$$
\begin{aligned}
&\text{if } x_2 - x_1 < d_s \\
&\quad v_1 := \max(0, v_1 - a_b) \\
&\text{else } v_1 := v_1 \\
&x_1 := x_1 + v_1 \\
&x_2 := x_2 + v_2
\end{aligned}
$$

$x_2 > x_1 > 0$

$v_1 \geq 0$

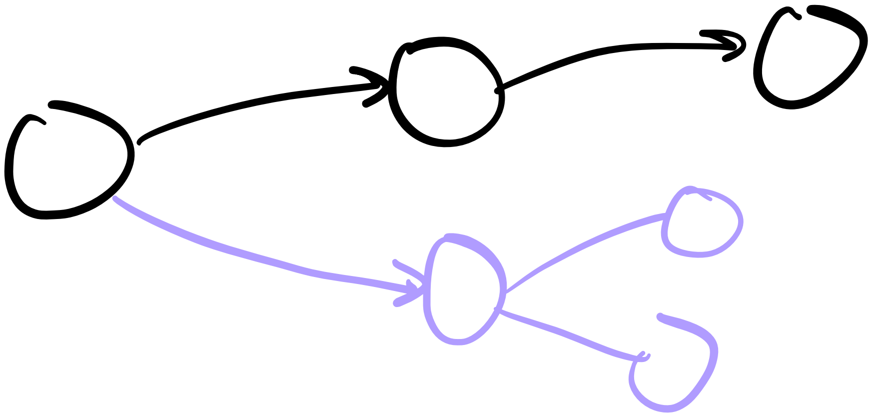
Executions and Behaviors

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

$$\alpha = q_0 q_1 q_2 \ldots$$

such that:

1. $q_0 \in Q_0$
2. $(q_i, q_{i+1}) \in D, \forall i$

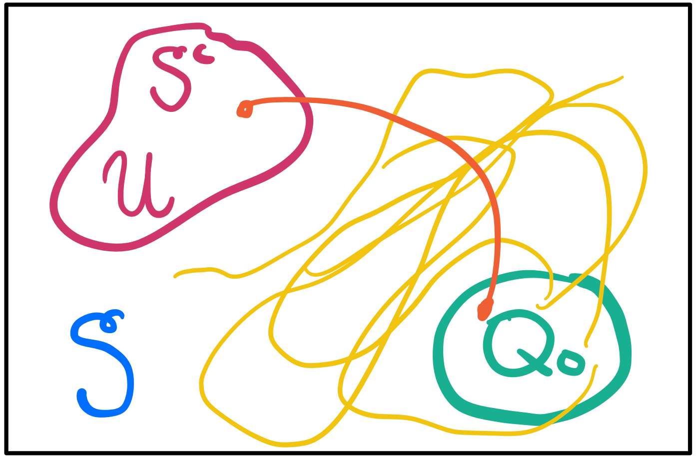


Note that nondeterministic A will have *many* executions!

Safety Requirements

We want to express our safety requirements as:

1. A formula involving state variables $x_2 - x_1 < \varepsilon$ ✓
2. A subset of Q



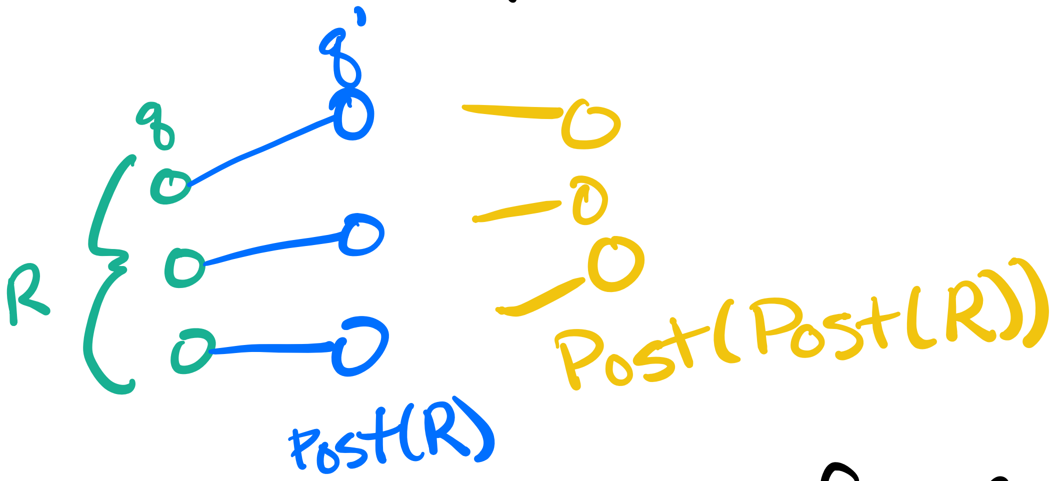
The Safety Verification Problem

Does there exist any execution $\alpha = q_0 \ldots q_k$ of A s.t. $q_k \in S^c = \mathcal{U}$?

if for every finite execution α of A and for every q_i in α, $q_i \in S$, then we say that A is w.r.t. S

Reachability and the Post operator

for $R \subseteq Q$, $Post(R) := \{q' \in Q \mid \exists q \in R \text{ and } \langle q, q' \rangle \in D\}$



→ provides set of reachable states

Partial Summary

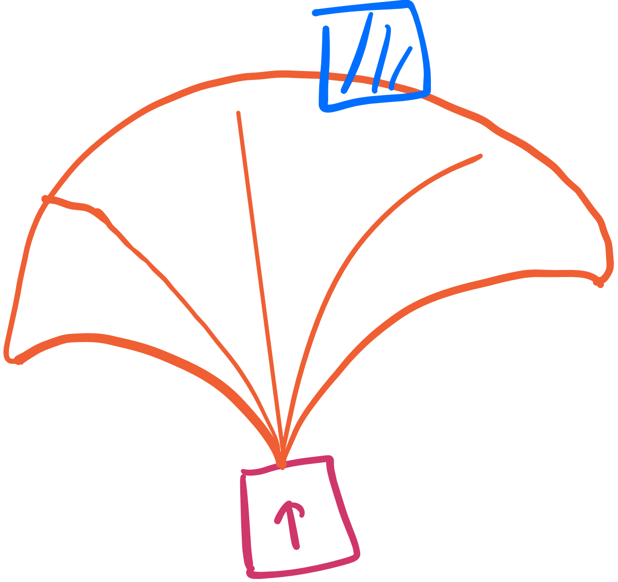
- Absolute safety checking boils down to showing that none of the executions of the automaton reaches an unsafe set U
  - To reason about all executions, we must work with infinite sets of states ☹

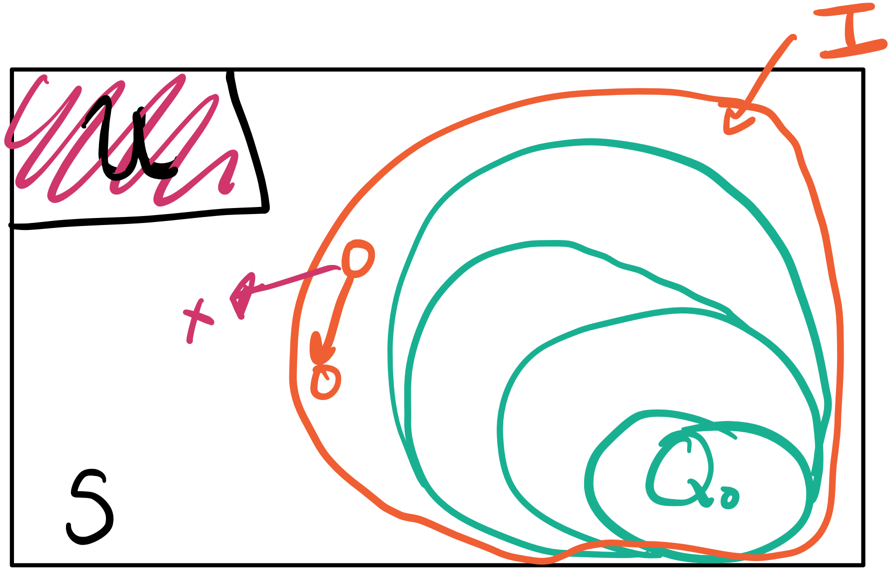
Partial Summary

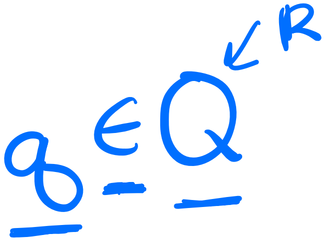
- Absolute safety checking boils down to showing that none of the executions of the automaton reaches an unsafe set U
  - To reason about all executions, we must work with infinite sets of states ☹
- One way to compute infinite sets is using the Post operator
  - However: computing all executions for unbounded time can be ***hard***

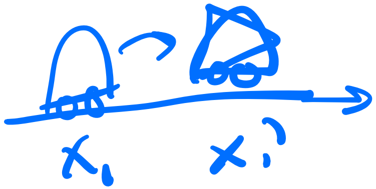
Partial Summary

- Absolute safety checking boils down to showing that none of the executions of the automaton reaches an unsafe set U
  - To reason about all executions, we must work with infinite sets of states ☹
- One way to compute infinite sets is using the Post operator
  - However: computing all executions for unbounded time can be ***hard***
- We will now introduce a potential shortcut: invariants!

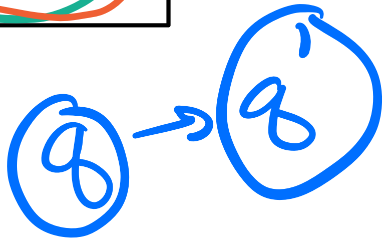








Inductive Invariants!



Inductive Invariants to Prove Safety

Thm if there exists $I \subseteq Q$ s.t.

(1) $Q_0 \subseteq I$ (2) $Post(I) \subseteq I$

then all executions A stay in I

Further, if $I \subseteq S$, then A is safe wrt S

$Q = \{ x_1, v_1, x_2, v_2 \}$

$q \in Q$

$q.x_1 \longrightarrow q'.x_1$

pre state post state

Proof by Induction

- For any execution of A, $\alpha = q_0, q_1, \ldots, q_k$, we will prove by induction on k that $\forall i \; q_i \in I$

1. Base case: $k = 0$, $\alpha = q_0 \in Q_0 \subseteq I$ by (1)
2. Inductive Step: Given $\alpha = q_0, q_1, \ldots, q_{k-1}, q_k$ and $q_{k-1} \in I$, show that $q_k \in I$

By (2), $Post(I) \leq I$

since we have $q_{k-1} \in I \Rightarrow q_k \in I$

$\therefore \forall i \; q_i \in I$

Simple requirement and candidate invariant (1)

$S_1 := v_1 \geq 0$ // $I_1 = [\![S_1]\!] := v_1 \geq 0$

(1) $Q_0 \subseteq I \triangleq \{q \mid q.v_1 \geq 0\}$ ✓

$\forall q_0 \in Q_0$ show that $q_0 \in I$

by def $q_0.v_{10} \geq 0 \Rightarrow q_0 \in I$ ✓

Simple requirement and candidate invariant (2)

(2) $Post(I) \subseteq I$

$Post(I) := \{q' \mid q \in I \text{ and } \langle q, q' \rangle \in D\}$

For any state $q \in I$ if $q.v_1 \geq 0$ and $(q, q') \in D$, show $q'.v_1 \geq 0$

if $q.x_2 - q.x_1 < ds$
$q'.v_1 = \max(0, q.v_1 - a_b)$ | $q'.v_1 \geq 0$

else
$q'.v_1 = q.v_1$ | $q'.v_1 = q.v_1 \geq 0$

Another requirement

$S_2: x_1 < x_2$ // Is S_2 an inductive invariant?

① $Q_0 \subseteq S_2$ $0 < x_{10} < x_{20}$ ✓

② $Post(S_2) \subseteq S_2$? ✗

if $q.v_1 >> q.x_2 - q.x_1$

then $q'.x_1$ max exceed $q'.x_2$

Adding more information

timer := 0

if $x_2 - x_1 < d_s$

    if $v_1 > a_b$

        $v_1 := v_1 - a_b$ (A)

        timer := timer + 1

    else $v_1 = 0$ (B)

else $v_1 := v_1$ (C)

$x_1 := x_1 + v_1$

$x_2 := x_2 + v_2$

I_3: $\text{timer} \le \frac{v_{10} - v_1}{a_b}$

(1) $q_0.\text{timer} = 0 \le \frac{v_{10} - q_0.v_1}{a_b}$ ✓

(2) $q \in I_3 \Rightarrow q' \in I_3$

check A, B, C

Three Cases to Consider: (1)

Three Cases to Consider: (2)

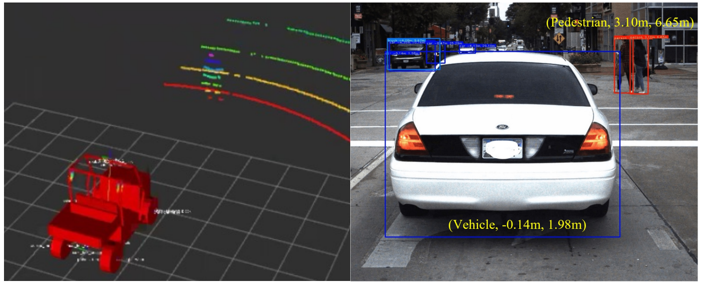
Three Cases to Consider: (3)

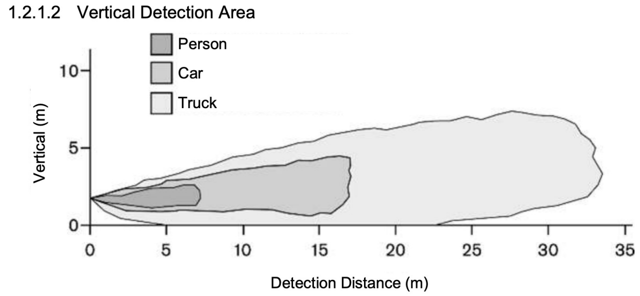
Showing Safety with a Timer

- Goal: show $x_2 - x_1 > 0$
- Maximum distance traveled by car 1 after detection:

Baked-in Assumptions (1)

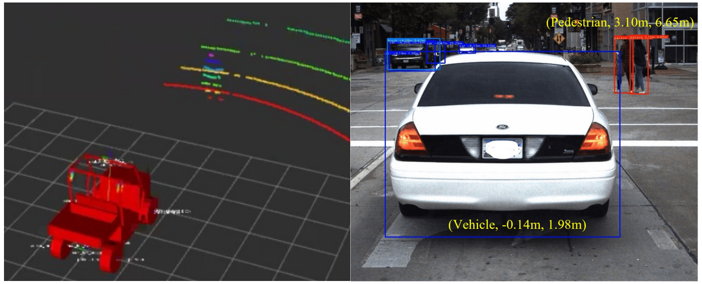
- Perception.
  - Sensor detects obstacle **iff** distance $d \leq D_{sense}$
  - How to model vision errors?



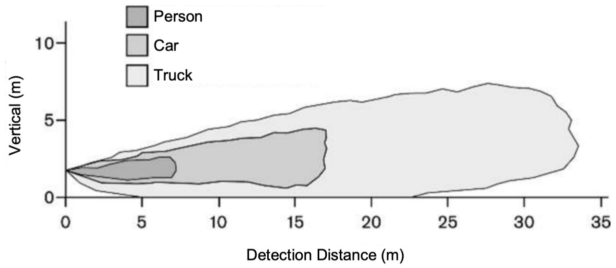


Baked-in Assumptions (1)

- Perception.
  - Sensor detects obstacle **iff** distance $d \leq D_{sense}$
  - How to model vision errors?
- Pedestrian Behaviors.
  - Pedestrian is assumed to be moving with constant velocity from initial position

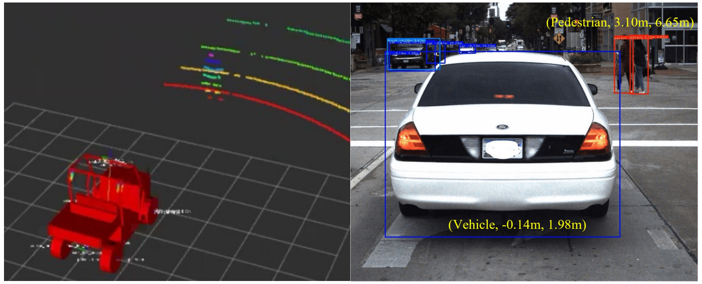


1.2.1.2 Vertical Detection Area

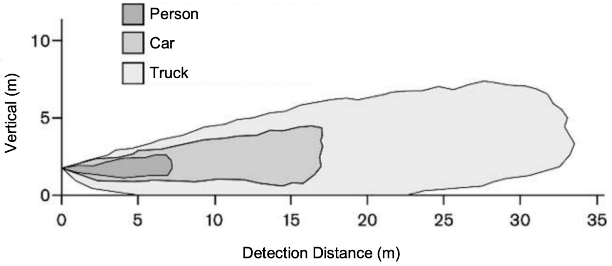


Baked-in Assumptions (1)

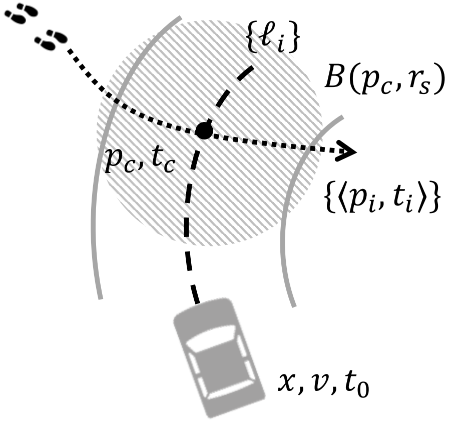
- Perception.
  - Sensor detects obstacle **iff** distance $d \leq D_{sense}$
  - How to model vision errors?
- Pedestrian Behaviors.
  - Pedestrian is assumed to be moving with constant velocity from initial position
- No sensing-computation-actuation delay.
  - The time step in which $d \leq D_{sense}$ is true is exactly when the velocity starts to decrease



1.2.1.2 Vertical Detection Area

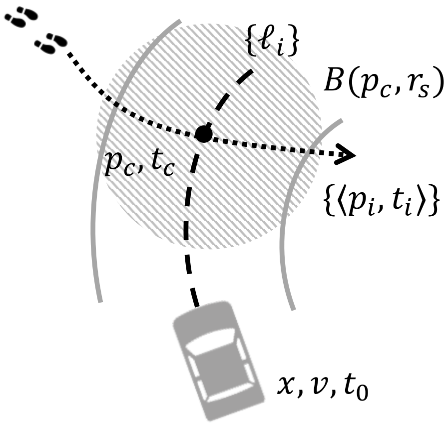


Baked-in Assumptions (2)



- Mechanical or Dynamical assumptions
  - Vehicle and pedestrian moving in 1-D lane.
  - Does not go backwards.
  - Perfect discrete kinematic model for velocity and acceleration.

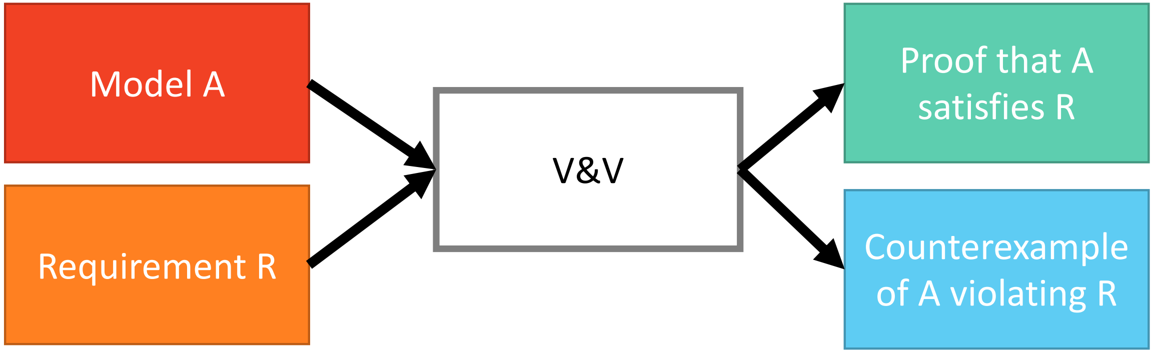
Baked-in Assumptions (2)



- Mechanical or Dynamical assumptions
  - Vehicle and pedestrian moving in 1-D lane.
  - Does not go backwards.
  - Perfect discrete kinematic model for velocity and acceleration.
- Nature of time
  - Discrete steps. Each execution of the above function models advancement of time by 1 step. If 1 step = 1 second, $x_1(t+1) = x_1(t) + v_1(t).1$
  - Atomic steps. 1 step = complete (atomic) execution of the program.
    - We cannot directly talk about the states visited after partial execution of program

Remarks and Takeaway

- The proof by induction shows a property of ***all behaviors of our model***
- The proof is conceptually simple, but can quickly get tedious and error prone
  - Verification and Validation tools like Z3, Dafny, PVS, CoQ, AST, MC2, automate this



Summary

- We must translate safety requirements into sets of states or formulas over state variables
- Reachability allows us to prove safety
- Invariant trick can give a shortcut for proving safety ☺
  - The invariant I may contain important information about conserved quantities and may also tell us why the system is safe
  - However, often requires guessing and checking and a lot of engineering effort
- Mind the gap between model and reality!
- Next: More safety (fun lecture)