ECE484 MP0

Daniel Zhuang, John Pohovey, Will Chen, Eric Ji, Guang Yin
Electrical and Computer Engineering
University of Illinois at Urbana-Champaign

Intro

- Welcome to ECE 484!

- Labs will include:
  - MP Walkthrough
  - MP Demo
  - OH

What is Verse?

- Tool to compute reachability created by Sayan Mitra's Reliable Autonomy lab
- Purpose: Verifying design of controller and transition logic
    - Do we meet all safety criteria?

- Continuous:
    - Dynamics of a vehicle (physics) Ex: Kinematics Bicycle Model
- Discrete modes:
    - Changing lanes, changing from flight to landing, etc
    - Dynamics change with each mode

- Reachability for Hybrid Systems
    - Verse Supports both continuous dynamics + discrete modes

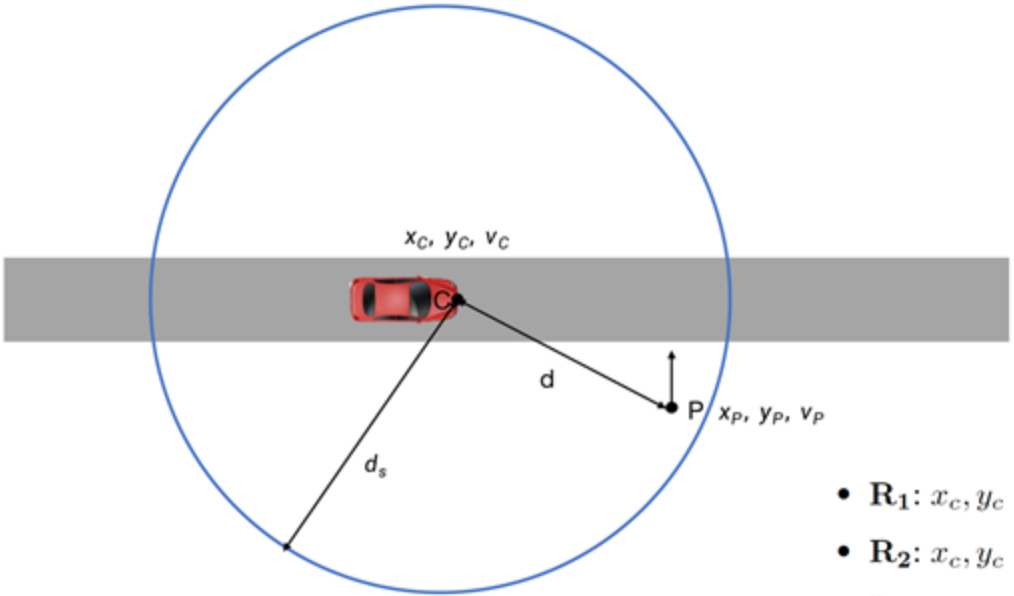
Design Problem 1

Design Problem 1

Design Problem 1

When do we brake? When do we accelerate?

More Details



- Dynamics already written
- Write logic to avoid collision
- Maximize average speed
- Time horizon: 50 secs
- Three ranges (R1, R2 and R3) for the initial states of the car and pedestrian
- Increasing difficulty from R1 to R3

- $\mathbf{R_1}$: $x_c, y_c \in [-5, 5]; v_c = 8; x_p = 175; y_p = -55; v_p = 3.$
- $\mathbf{R_2}$: $x_c, y_c \in [-5, 5]; v_c \in [5, 10]; x_p = 175; y_P = -55; v_P = 3.$
- $\mathbf{R_3}$: $x_c, y_c \in [-5, 5]; v_c \in [5, 10]; x_p \in [165, 175]; y_p \in [-55, -50]; v_P = 3.$

Design a Decision Logic (DL)

```
def decisionLogic(ego: State, other: State):
    output = copy.deepcopy(ego)

    # TODO: Edit this part of decision logic

    if ego.agent_mode == VehicleMode.Normal and other.dist < 12:
        output.agent_mode = VehicleMode.Brake

    ##########################################

    assert other.dist > 2.0 → Safety Requirement: C has to be at least 2 m away from P

    return output
```

- Determines how to transition between discrete modes

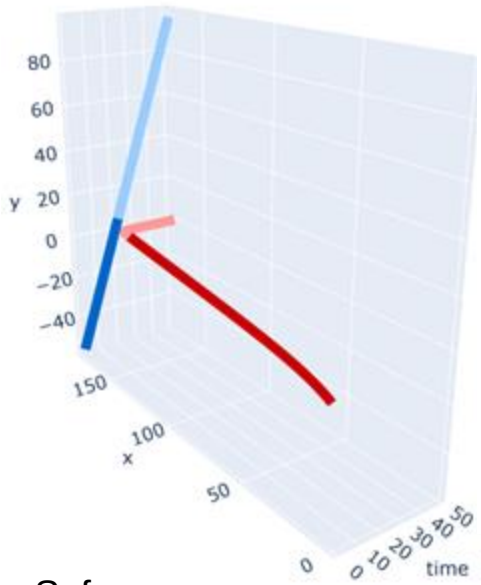
- Design with if statements
- Must be written a specific way
  - Linear inequalities/equalities and logical operators in the if condition
  - In the if body, assign the next mode (Normal, Brake, HardBrake, or Accel)

- When running reachability:
  - DL Looks like code but IT'S NOT ACTUALLY CODE
  - Parsed and analyzed - not run in the traditional sense.

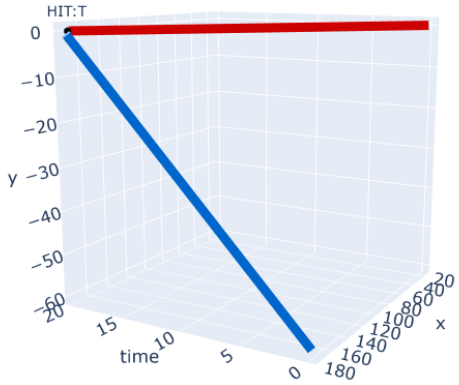
Simulation

Car

Pedestrian



Safe



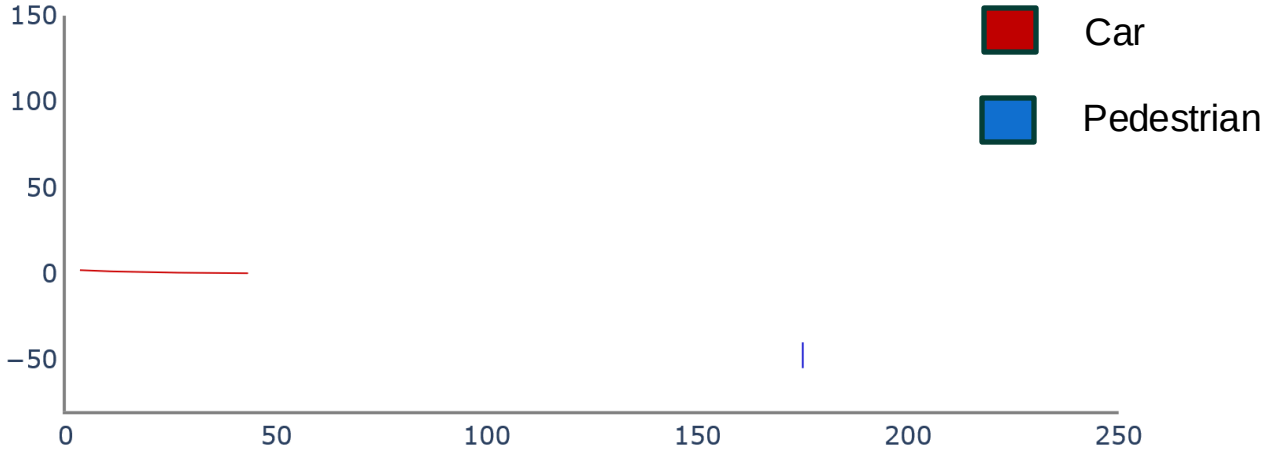
Unsafe

- Randomly sampled initial states for the car and pedestrian from initial range

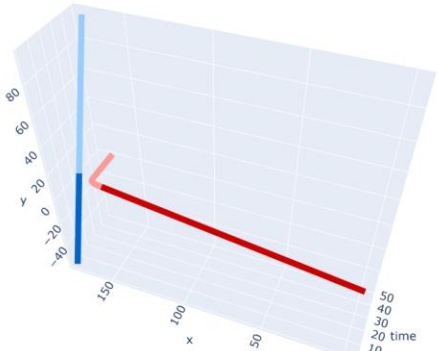
- Run for 50 secs
- Plot trajectories of both
- No safety violation→ Safety Achieved
- Safety violation → Safety Not Achieved

Example

Simulate from this initial state

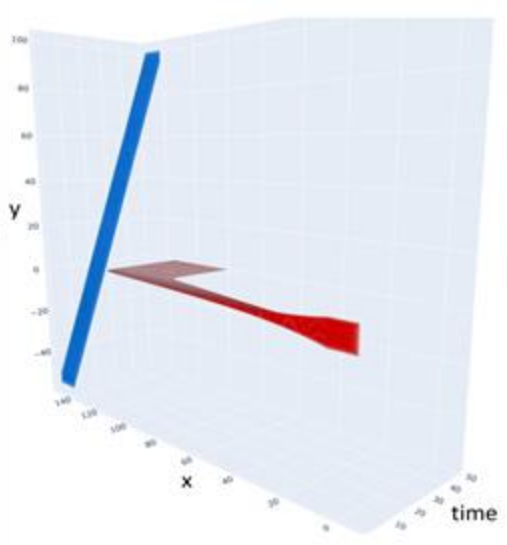


- Initial ranges for Vehicle:
  - x: [-5,5]
  - y: [-5,5]
  - vx : [0,10]
- Initial ranges for Pedestrian:
  - x: [120,200]
  - y: [-50,-25]
  - vy: [0,5]



Choose (x= 0,y =0,vx = 8) as initial state for vehicle
Choose (x = 175, y = -55 ,vy = 3) as initial state for pedestrian

Reachability Analysis

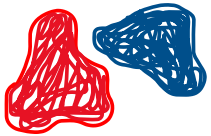


- What if we want to test the entire initial set?
- Simulate every possibility in the initial conditions.
  - Computationally infeasible
- Solution: Approximate using Verse's reachability analysis

- Take entire initial set and analyze all possible states that car+ pedestrian can reach
- At timestep t, reachable set represents region where the car+pedestrian could be at t
- Combine all timesteps: reachtube

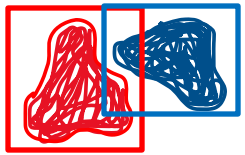
Reachability: Pt 2

- No safety assertion violation → Safety Guaranteed
- Safety assertion violation→ Nothing is Guaranteed
  - Ex: We somehow calculate the exact reachable set for car + pedestrian at some time t



No intersection!

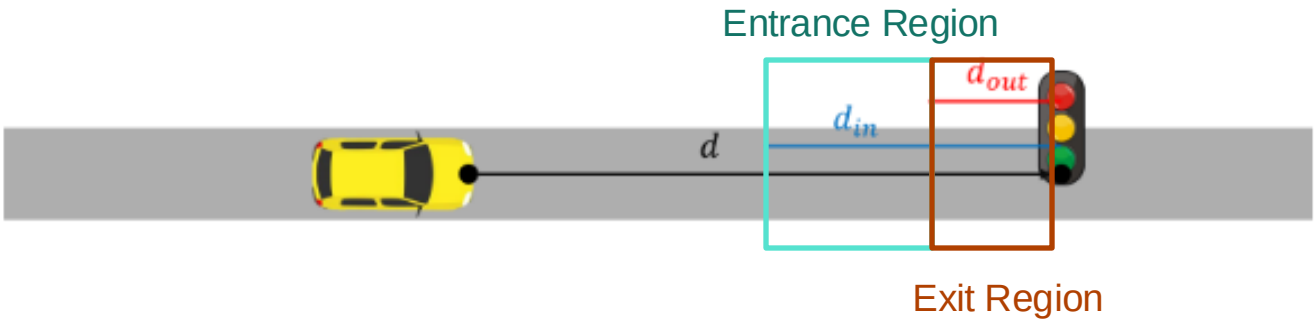
  - Reachable set means car and pedestrian is somewhere in this region.
  - Verse over-approximates with rectangles: rectangle contains actual reachable set



Intersection

  - Verse can be too conservative – approximation too big
  - Most conservative reachable set: the entire state space
    - Reachable set will always intersect unsafe set
    - Verse's purpose: use assumptions to shrink reachable set

Design Problem 2:



- Lights cycle continuously
- Safety:
  - If the light is red:
    - The vehicle cannot be in the entrance region
    - The vehicle cannot be slow in the exit region (make fast exit)
- Maximize average speed when not stopping for light
- Three ranges for vehicle's initial states

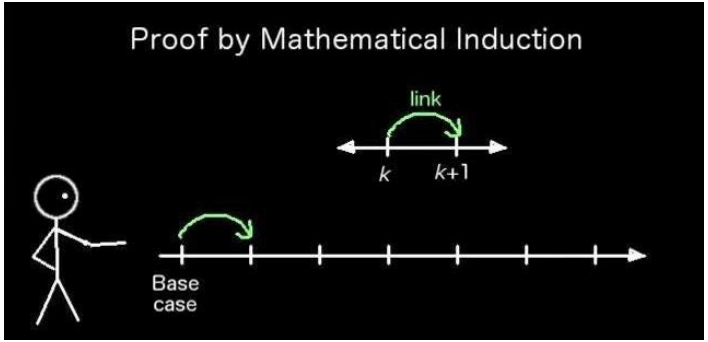
- You may use either one or both of simulation and reachability
  - **Read MP doc to determine which functions to use**

- If stuck: Look at Verse documentation and troubleshooting page

User Study

- Voluntary (highly encouraged)
- Fill out consent to participate
- Fill out feedback form after completing each component
- +1 point on Midterm 1 (capped)

HW 0: Review of Induction Method



Any statement P(n) which is for “n” natural number can be proved using the Principle of Mathematical Induction by following the below steps:

Base Step	Verify if the statement is true for trivial cases (n = 1) i.e. check if P(1) is true.
Assumption Step	Assume that the statement is true for n = k for some k ≥ 1 i.e. P(k) is true.
Induction Step	If the truth of P(k) implies the truth of P(k + 1), then the statement P(n) is true for all n ≥ 1.

HW 0: Testing Automatic Emergency Braking

SimpleCar$(D_{sense}, v_0, x_{10}, x_{20}, a_b)$, $x_{20} > x_{10}$

Initially: $x_1(0) = x_{10}$, $x_2(0) = x_{20}$, $v_1(0) = v_0$, $v_2(0) = 0$

$s(0) = 0$, $timer(0) = 0$, $timer2(0) = 0$

$d(t) = x_2(t) - x_1(t)$

if $d(t) \leq D_{sense}$

    $s(t+1) = 1$

    if $v_1(t) \geq a_b$

        $v_1(t+1) = v_1(t) - a_b$

        $timer(t+1) = timer(t) + 1$

        $timer2(t+1) = timer2(t)$

    else

        $v_1(t+1) = 0$

        $timer(t+1) = timer(t)$

        $timer2(t+1) = timer2(t)$

else

    $s(t+1) = 0$

    $v_1(t+1) = v_1(t)$

    $timer(t+1) = timer(t)$

    $timer2(t+1) = timer2(t) + 1$

$x_1(t+1) = x_1(t) + v_1(t)$

- Zero Reaction Delay
  - Choose different car velocity and sensing distance
  - Analyze the safety of the system

- Adding Reaction Delay
  - Choose different reaction delay time and car's deceleration
  - Analyze the safety of the system

Grading

- HW0 (100 pts)
- Specific point breakdown in documentation
- **Individual submission**
- Upload the individual work to Canvas. Include your name and netid in the pdf.
-
- MP0 (100pts)
- Report 90 pts
- Demo 10 pts
- **Individual submission**
- Upload the report and code to Canvas. Include your name and netid in the pdf.

Important Logistics

- MP0 and HW0 is due September 13th
  - You must show up for the demo!
- Think about your groups for future MPs (Groups must be in the same section)
- Next week, we will talk about Lane Detection (MP1)