Lecture 2: Simple Safety

Professor Katie Driggs-Campbell

January 28, 2021

ECE484: Principles of Safe Autonomy

Administrivia

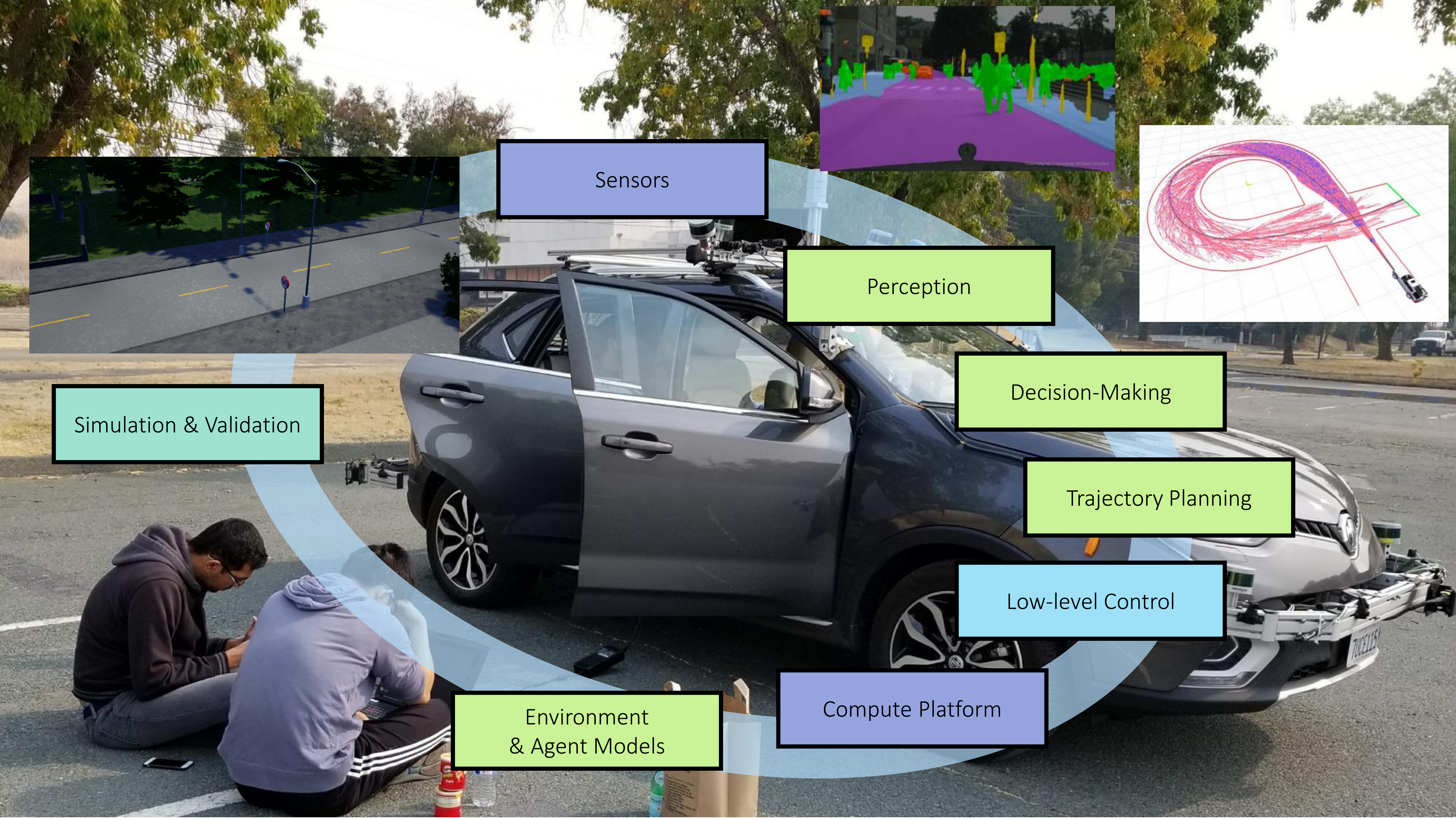
- Please form teams!
- Questions for the Guest Speakers
- Review COVID precautions for the GEM

Today’s Lecture

- Project Review
- Simple example of ensuring safety

Projects: explore, inspire, and impress

- We will provide a fully equipped Polaris GEM e2 vehicle (test vehicle and simulation) and basic autonomy modules
- Action Items:
  - If interested in working with hardware, become an IRL member asap!
    - Safety Driver training in the next few weeks
  - Team formation form due this week!
  - Project Pitch (in-class, likely parallel track) in one month
  - Two milestones due through out the semester
  - Project Presentations second to last week of class
  - Final report (and video demo) in place of final


Sensors
Perception
Decision-Making
Trajectory Planning
Low-level Control
Compute Platform
Environment
& Agent Models
Simulation & Validation

An honest scientific approach

1. Create detailed mathematical models of the autonomous systems and its environment
2. Enumerate the precise requirements of the system and the conditions on the environment under which it is supposed to work
3. Analyze the system to either
   - prove that all behaviors meet the requirement (perhaps with high probability)
   - find counter-examples, corner cases, etc., debug and repeat

- Currently, there are fundamental flaws in making this work for autonomous systems
- **Why study this approach?**
  - Careful reasoning can expose flawed assumptions and potentially bad design choices
  - Found success in other industries: microprocessors, aviation, cloud computing, nuclear, …
  - Working deliberately towards a <u>more perfect understanding</u> is a worthwhile intellectual struggle

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!
2. Identify the *requirements* and *assumptions*
3. Analyze model to show that it meets the requirements under the assumptions

Emergency Braking for Pedestrians

Image Credit: Bosch


LiveLeak

What are the design considerations and tradeoffs?

Modeling the scenario

- What is a model of a system?
- A *mathematical model* describes how a system behaves.
  - What are the key parameters and states?
  - How are the parameters selected by nature?
  - What are the initial conditions of the state?
  - How do the state change over time? …
  - What parts of the model are available for observation/analysis?
- Models include the implicit and explicit assumptions (biases) we are making about the system

A simple model

A simple model as a program

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

initially: $x_1 = x_{10}, v_1 = v_0, x_2 = x_{20}, v_2 = 0$

    $s = 0, timer = 0$

if $d \leq D_{sense}$

    $s = 1$

    if $v_1 \geq a_b$

        $v_1 = v_1 - a_b$

        $timer = timer + 1$

    else

        $v_1 = 0$

$x_1 = x_1 + v_1$

Image Credit: Bosch

Model Behavior

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

initially: $x_1 = x_{10}, v_1 = v_0, x_2 = x_{20}, v_2 = 0$

  $s = 0, timer = 0$

if $d \leq D_{sense}$

  $s = 1$

  if $v_1 \geq a_b$

    $v_1 = v_1 - a_b$

    $timer = timer + 1$

  else

    $v_1 = 0$

$x_1 = x_1 + v_1$

More explicit program

SimpleCar$(D_{sense}, v_0, x_{10}, x_{20}, a_b)$, $x_{20} > x_{10}$
initially: $x_1 = x_{10}, v_1 = v_0, x_2 = x_{20}, v_2 = 0$
  $s = 0, \mathit{timer} = 0$
if $d \leq D_{sense}$
  $s = 1$
  if $v_1 \geq a_b$
    $v_1 = v_1 - a_b$
    $\mathit{timer} = \mathit{timer} + 1$
  else
    $v_1 = 0$
$x_1 = x_1 + v_1$

SimpleCar$(D_{sense}, v_0, x_{10}, x_{20}, a_b)$, $x_{20} > x_{10}$
initially: $x_1(0) = x_{10}, v_1(0) = v_0, x_2(0) = x_{20}, v_2(0) = 0$
  $s(0) = 0, \mathit{timer}(0) = 0$
$d(t) = x_2(t) - x_1(t)$
if $d(t) \leq D_{sense}$
  $s(t+1) = 1$
  if $v_1(t) \geq a_b$
    $v_1(t+1) = v_1(t) - a_b$
    $\mathit{timer}(t+1) = \mathit{timer}(t) + 1$
  else
    $v_1(t+1) = 0$
    $\mathit{timer}(t+1) = \mathit{timer}(t)$
else
  $s(t+1) = 0$
  $v_1(t+1) = v_1(t)$
  $\mathit{timer}(t+1) = \mathit{timer}(t)$
$x_1(t+1) = x_1(t) + v_1(t)$

Identifying requirements: Define safety

- A *requirement* is a precise statement about what the behaviors of the system should and should not do

- An *invariant* is a requirement that something *always* holds. Examples:
  - "Car always remains far from the pedestrian"
  - "Drones never cross over to above 400ft in the airspace"
  - "A fully attentive safety driver should always be present during autonomy experiments"

Does our model satisfy the requirement?

Requirement: “Car always remains far from the pedestrian”

Invariant 1. For all $x_{10}, x_{20}, v_0, D_{sense}, a_b$ and for all t, $x(t).d > 0$

→ Does this invariant hold for our model?

Another Invariant!

Invariant 2. $timer + \frac{v_1}{a_b} \leq \frac{v_0}{a_b}$

Invariant 2. For all $x_{10}, x_{20}, v_0, D_{sense}, a_b$ and for all t, $x(t).timer + \frac{\mathbf{x}(t).v_1}{a_b} \leq \frac{v_0}{a_b}$

Proof by Induction (1)

Invariant 2. For all $x_{10}, x_{20}, v_0, D_{sense}, a_b$ and for all t,

$$\boldsymbol{x}(t).timer + \frac{\boldsymbol{x}(t).v_1}{a_b} \leq v_0/a_b$$

1. Base case. $P(\boldsymbol{x}(0))$

$$\boldsymbol{x}(0).timer + \frac{\boldsymbol{x}(0).v_1}{a_b}$$

2. Induction. $P(\boldsymbol{x}(t)) \Rightarrow P(\text{SimpleCar}(\boldsymbol{x}(t)))$

   Three cases to consider

   1. $d > D_{sense}$
   2. $d \leq D_{sense} \wedge v_1 \geq a_b$
   3. $d \leq D_{sense} \wedge v_1 < a_b$

SimpleCar$(D_{sense}, v_0, x_{10}, x_{20}, a_b)$, $x_{20} > x_{10}$
initially: $x_1(0) = x_{10}, v_1(0) = v_0, x_2(0) = x_{20}, v_2(0) = 0$
  $s(0) = 0$, $timer(0) = 0$
$d(t) = x_2(t) - x_1(t)$
if $d(t) \leq D_{sense}$
  $s(t+1) = 1$
  if $v_1(t) \geq a_b$
    $v_1(t+1) = v_1(t) - a_b$
    $timer(t+1) = timer(t) + 1$
  else
    $v_1(t+1) = 0$
    $timer(t+1) = timer(t)$
else
  $s(t+1) = 0$
  $v_1(t+1) = v_1(t)$
  $timer(t+1) = timer(t)$
$x_1(t+1) = x_1(t) + v_1(t)$

Proof by Induction (2)

Invariant 2. For all $x_{10}, x_{20}, v_0, D_{sense}, a_b$ and for all t,

$$\boldsymbol{x}(t).timer + \frac{\boldsymbol{x}(t).v_1}{a_b} \leq v_0/a_b$$

1. $d > D_{sense}$

   $$\boldsymbol{x}(t+1).timer + \frac{\boldsymbol{x}(t+1).v_1}{a_b} = \boldsymbol{x}(t).timer + \frac{\boldsymbol{x}(t).v_1}{a_b} \leq v_0/a_b$$

2. $d \leq D_{sense} \wedge v_1 \geq a_b$

   $$\boldsymbol{x}(t+1).timer + \frac{\boldsymbol{x}(t+1).v_1}{a_b} =$$

   $$\boldsymbol{x}(t).timer + 1 + \frac{\boldsymbol{x}(t).v_1 - a_b}{a_b} \leq v_0/a_b$$

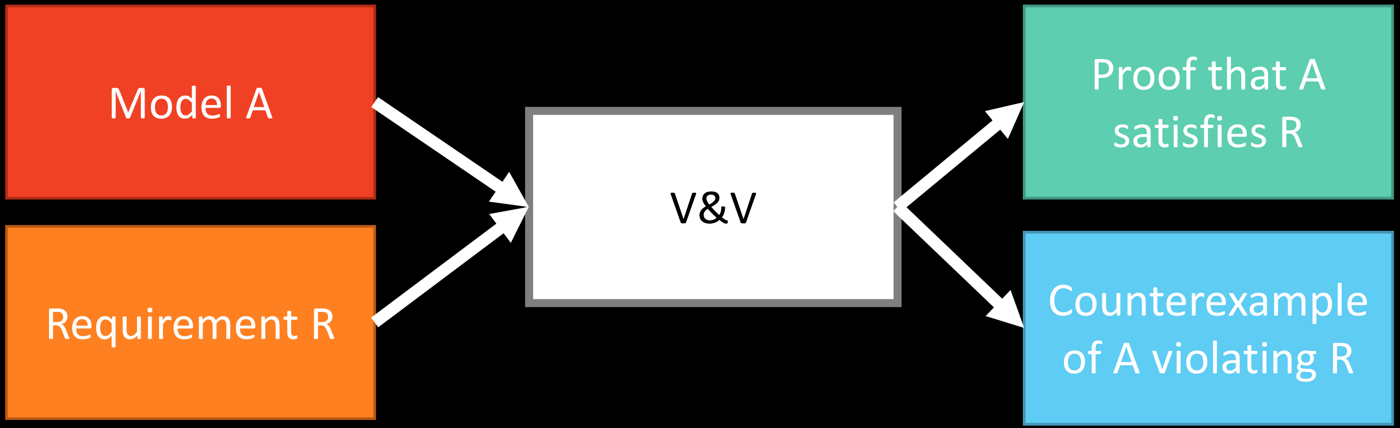
3. $d \leq D_{sense} \wedge v_1 < a_b$

   $$\boldsymbol{x}(t+1).timer + \frac{\boldsymbol{x}(t+1).v_1}{a_b} = \boldsymbol{x}(t).timer + 0 \leq v_0/a_b$$

```
SimpleCar(D_sense, v_0, x_10, x_20, a_b), x_20 > x_10
initially: x_1(0) = x_10, v_1(0) = v_0, x_2(0) = x_20, v_2(0) = 0
  s(0) = 0, timer(0) = 0
d(t) = x_2(t) − x_1(t)
if d(t) ≤ D_sense
  s(t+1) = 1
  if v_1(t) ≥ a_b
    v_1(t+1) = v_1(t) − a_b
    timer(t+1) = timer(t) + 1
  else
    v_1(t+1) = 0
    timer(t+1) = timer(t)
else
  s(t+1) = 0
  v_1(t+1) = v_1(t)
  timer(t+1) = timer(t)
x_1(t+1) = x_1(t) + v_1(t)
```

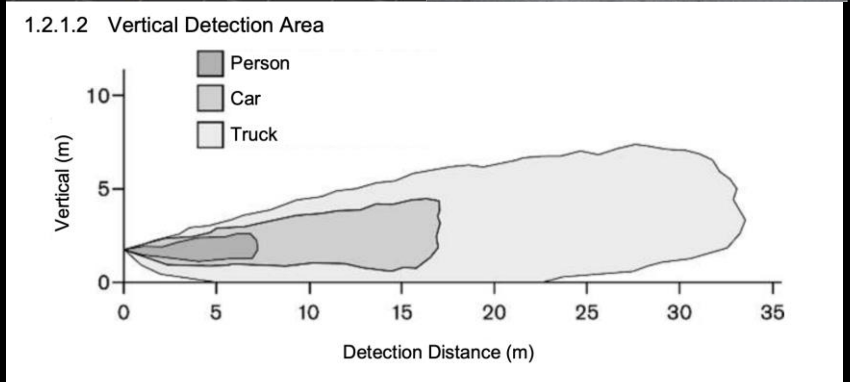
Remarks and takeaway messages from the exercise

- **Invariant 2** takes us close to proving safety of our model (**Invariant 1**)
- We will need to **add assumptions** on the model to complete the proof
- 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



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



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 Time. Each execution of the above function models advancement of time by 1 step. If 1 step = 1 second, then

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

    - We cannot talk about what happens between [t, 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

Summary

- Form your team. Decide project track.
  - Sign-up to be member of IRL.
- Careful modeling and reasoning can expose flawed assumptions, bad design bugs, and make the system *explainable.*
  - What are the baked-in assumptions and prescribed assumptions?
  - How to formalize requirements (e.g., safety, smoothness)?
- An example of an inductive proof for safety verification of a discrete time model
  - Discrete time model: states, initial states, transition function
  - Requirements, invariants, e.g., safety
  - Counter-examples
- Detailed discussion of baked-in *assumptions* and discovered assumptions