Principles of Safe Autonomy
ECE 484 Lecture 2: System Safety

Professor: Sayan Mitra

Why is it difficult to reason about safety purely using data

"Testing can be used to show the presence of bugs, but never to show their absence!'' --- Edsger W. Dijkstra

Because there are infinitely many *executions* and we can only test finitely many of those in any testing algorithm

In a probabilistic sense also, purely using data to gain safety assurance is not practical

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

To learn or extrapolate about all---infinitely many---executions from a finite sampling of executions, we need to make some assumptions about the system. A collection of these assumptions defines a ***model***

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

Why is it difficult to reason about safety purely using data

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

To learn or extrapolate about all---infinitely many---executions from a finite sampling of executions, we need to make some assumptions about the system. A collection of these assumptions defines a ***model***

Different types of model (and data) for sensing, control, planning, and we need to understand how to analyze and compose them

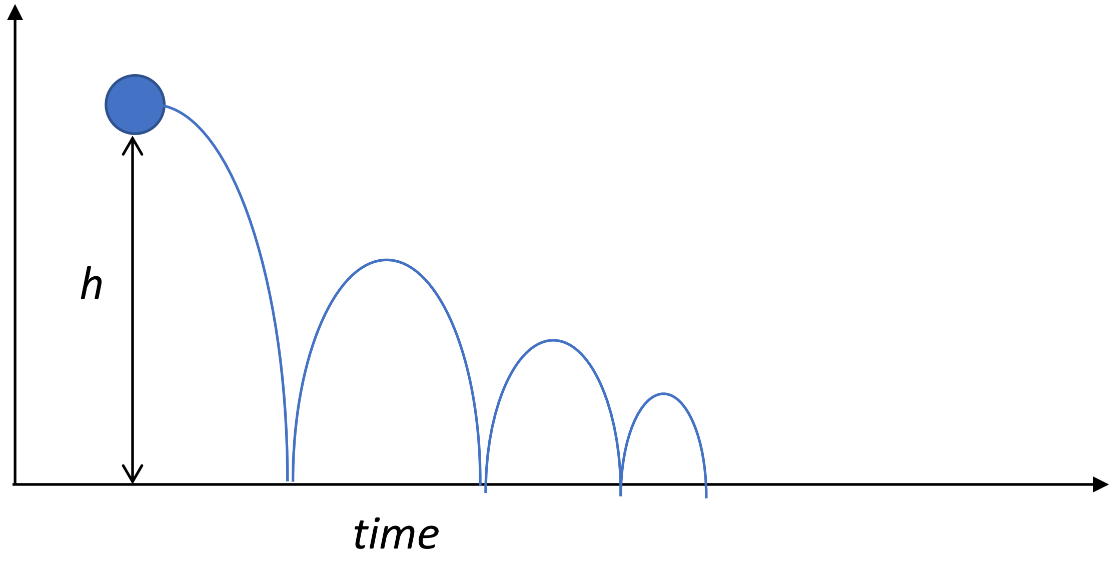
On a Formal Model of Safe and Scalable Self-driving Cars by Shai Shalev-Shwartz, Shaked Shammah, Amnon Shashua, 2017 (Responsibility Sensitive Safety)

Roadmap

- A simple class of models: *automata*
- What are executions of automata: sequence of states
- What are *requirements*?
- *Reachable states*, why we care to compute and why that can be hard
- *Invariants* as approximations of reachable states

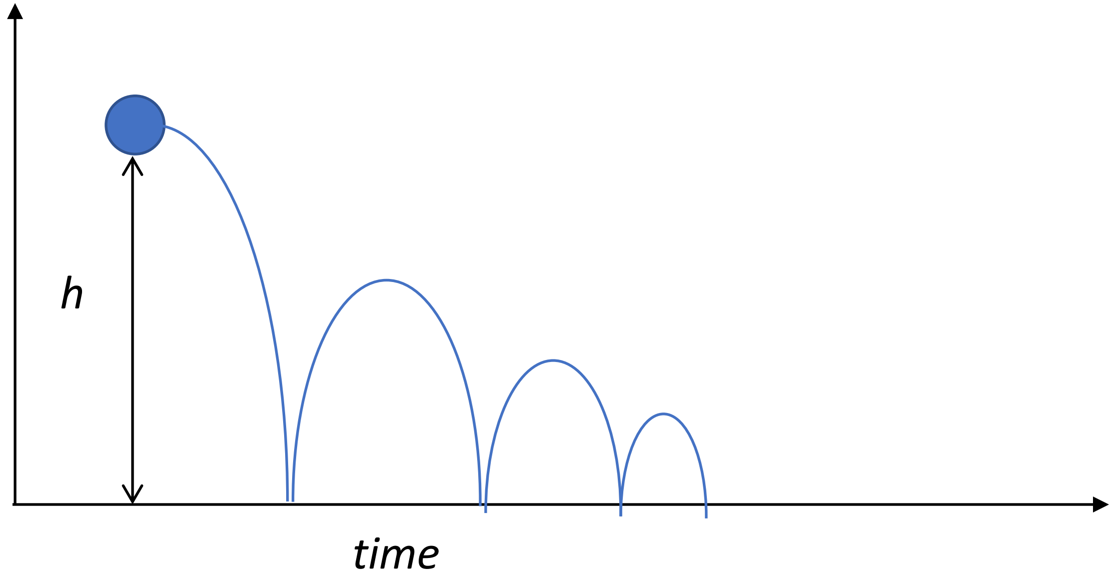
Example model of a bouncing ball

- Write the model of a ball dropped from height *h*



Example model of a bouncing ball

1. Define **states**---the *attributes* of the ball that completely define its motion: height x and velocity v
2. Define **state transitions**---how the state changes



Example model of a bouncing ball

State variables

$x: \mathbb{R}$

$v: \mathbb{R}$

State transitions

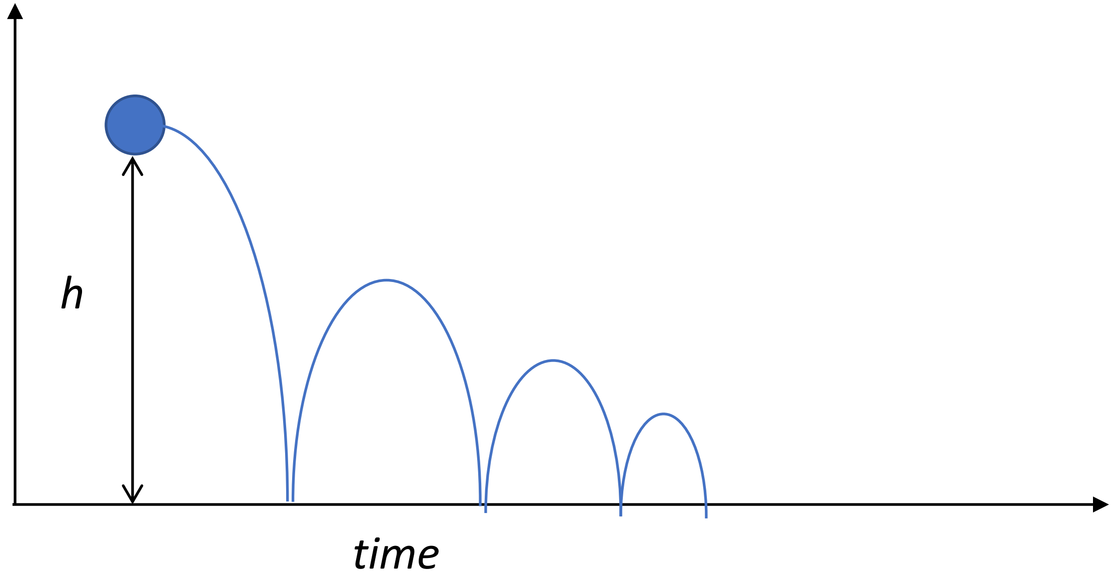
if $x \leq 0 \; \&\& \; v \leq 0$

$$v' = -c * v$$

else $v' = v$

$v' = v - g * delta$

$x' = x + delta * v$



Parameters

h, g, c, delta

Jupyter notebook https://github.com/PoPGRI/CodeACar22/blob/main/jupyter/control_notebook/main.ipynb

Automata or State Machines

Definition. An ***automaton*** ***(****also called* ***state machine****)* is a mathematical model define by:

- A set Q called the set of ***states***
- A set $Q_0 \subseteq Q$ of ***initial states***
- A set D $\subseteq Q \times Q$ called the set of transitions

1. Not necessarily finite state
2. Not necessarily deterministic

For the bouncing ball

$Q = \mathbb{R}^2$

$Q_0 = \{\langle h, 0 \rangle\}$

$D = ?$

Is it deterministic?

Model for AEB

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

AEB: Automatic Emergency Brakir

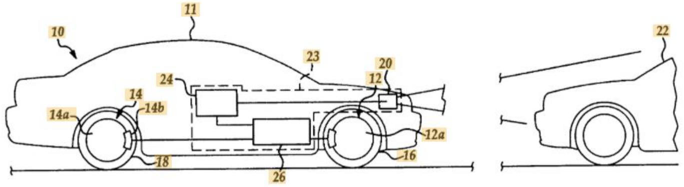


Figure 1

Not trivial

Today there is no enforced standard for testing AEB

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**.

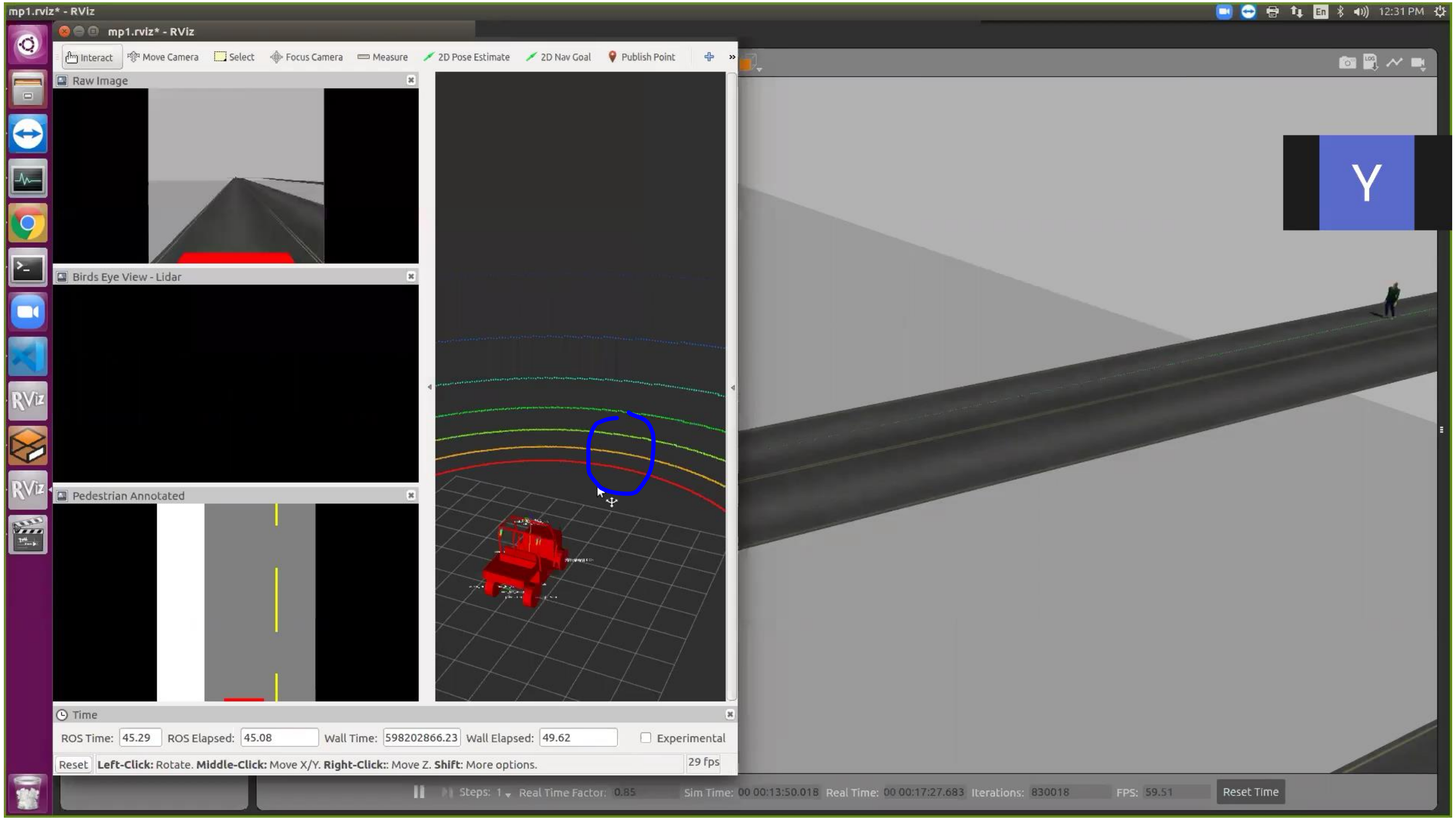
www.google.com.pg › patents
Braking control system for vehicle - Google Patents
An **automatic emergency braking** system for a vehicle includes a forward viewing camera and a control. At least in part responsive to processing of captured ...

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 ...

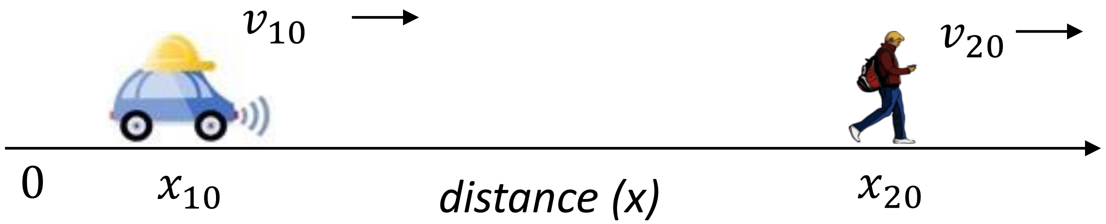
insurancenewsnet.com › oarticle › patent-application-tit...
Patent Application Titled "Multiple-Stage Collision Avoidance ...
Apr 3, 2019 - No assignee for this **patent** application has been made. ... **Automatic emergency braking** systems will similarly, also, soon be required for tractor ...

“simple”≠ Easy

MP0: Simulate model for testing



Model of Automatic Emergency Braking



State variables

$x_1, x_2 : \mathbb{R} = x_{10}, x_{20}$

$v_1, v_2 : \mathbb{R}$

State transitions

If $x_2 - x_1 \leq d_{safe}$

$v_1' = \max(0, v_1 - a_{brake})$

else $v_1' = v_1 * c$

$x_1' = x_1 + v_1$

$x_2' = x_2 + v_2$

Automaton model for AEB

$Q = \mathbb{R}^4$

$Q_0 = \{\langle x_{10}, x_{20}, v_{10}, v_{20} \rangle\}$

$D = ?$

Generalize model by adding uncertainty

- Range of initial conditions $x_1 \colon \mathbb{R} \in [x_{10} - 0.5, x_{10} + 0.5]$
- Range of braking force
  - $a_{brake} = choose\ [a_1, a_2]$
  - $v_1' = \max(0, v_1 - a_{brake})$
- Noise in sensing distances …
- Frequency of updates

Behaviors of automata

Definition. Given an automaton $A = \langle Q, Q_0, D \rangle$ an **execution** is a sequence of states $\alpha = q_0, q_1, q_2, \ldots$ such that (1) $q_0 \in Q_0$ and (2) for each i, $(q_i, q_{i+1}) \in D$.

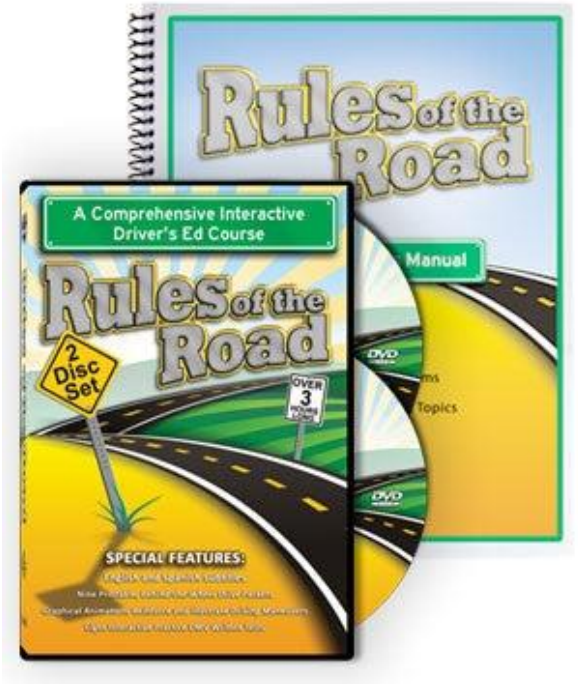
For execution α, we denote the k^{th} state as $\alpha(k)$

An automaton is deterministic if it has (essentially) a single execution

-- Not very interesting because has no uncertainty

Generally, the set of executions of A is uncountably infinite.

Requirements



Definition. A *requirement* is a statement about a system's behaviors.

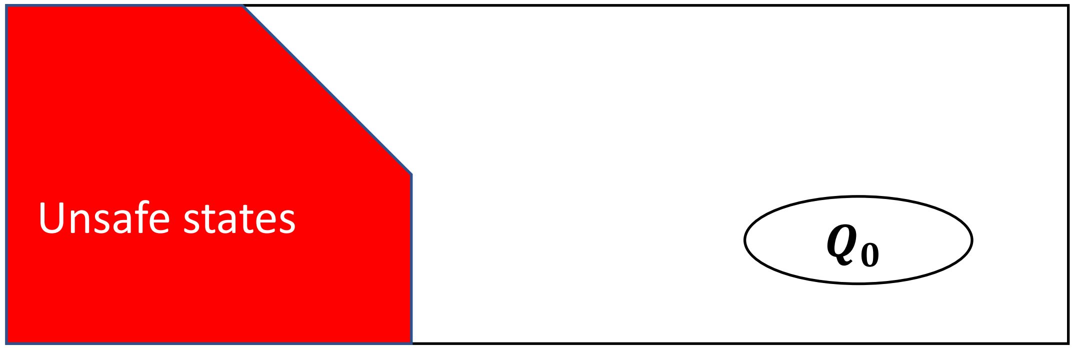
- Examples. "Ball <u>never</u> reaches a height above h"
  $\forall\ t, x(t) \leq h$
- "Ball eventually sits on the ground at x = 0" $\exists t, x(t) = 0$
- "Car <u>always</u> maintains safe distance to pedestrian"
  $\forall t, x_2(t) - x_1(t) > 2\ m$
- "Car <u>never</u> exceeds speed limit" …

Safety requirements are statements that must always hold (or never be violated)

Rules of the road ++

A picture for safety requirements

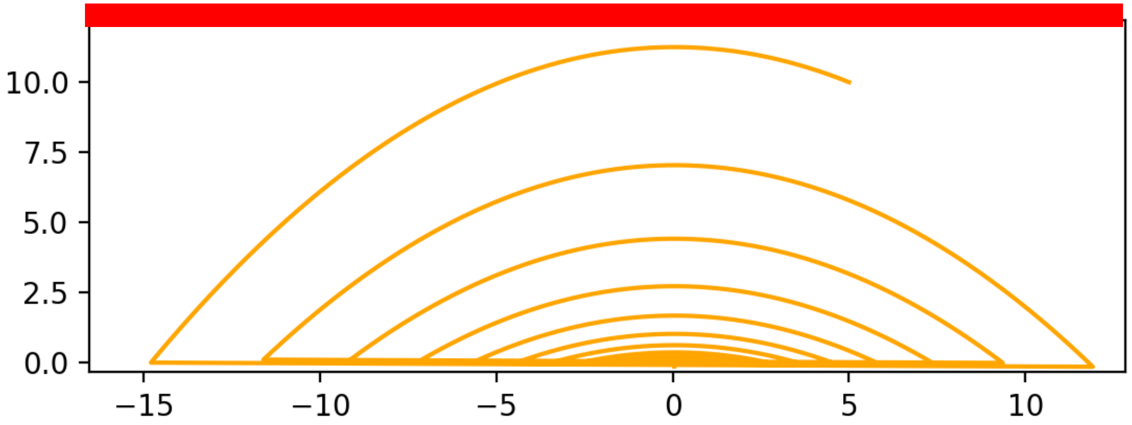
Safety requirements can be equivalently seen as a set of **unsafe states** that must always be avoided



"Ball never reaches a height above h"
$\forall\, t, x(t) \leq h$

corresponding unsafe set

$U = \{\langle x, v\rangle | x > h\} \subseteq \mathbb{R}^2$



Exercise. Try to plot projections of the unsafe states for AEB example.

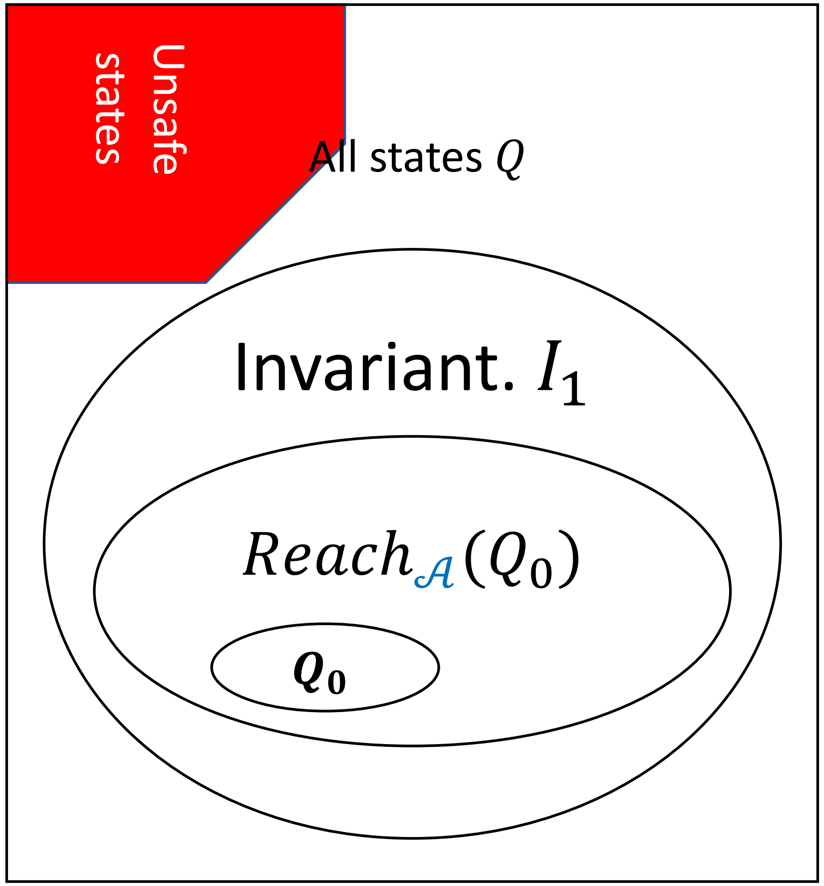
Safety verification problem

Definition. Given an automaton $A = \langle Q, Q_0, D \rangle$ and a safety requirement $\mathrm{U} \subseteq Q$, we have to decide whether $\forall$ execution α of $A, \forall k, \alpha(k) \notin U$?

That is, does automaton A ever ***reach*** U ?

How will you show that the ball never crosses h?

For any automaton



Reachable states

Definition. Given an automaton $A = \langle Q, Q_0, D \rangle$ the set of **reachable states** of A is defined as $\mathrm{Reach}_A = \{q \in Q \mid \exists\ \alpha, k, such\ that\ \alpha(k) = q\}$.

A state is **reachable** if there is some execution that reaches it.

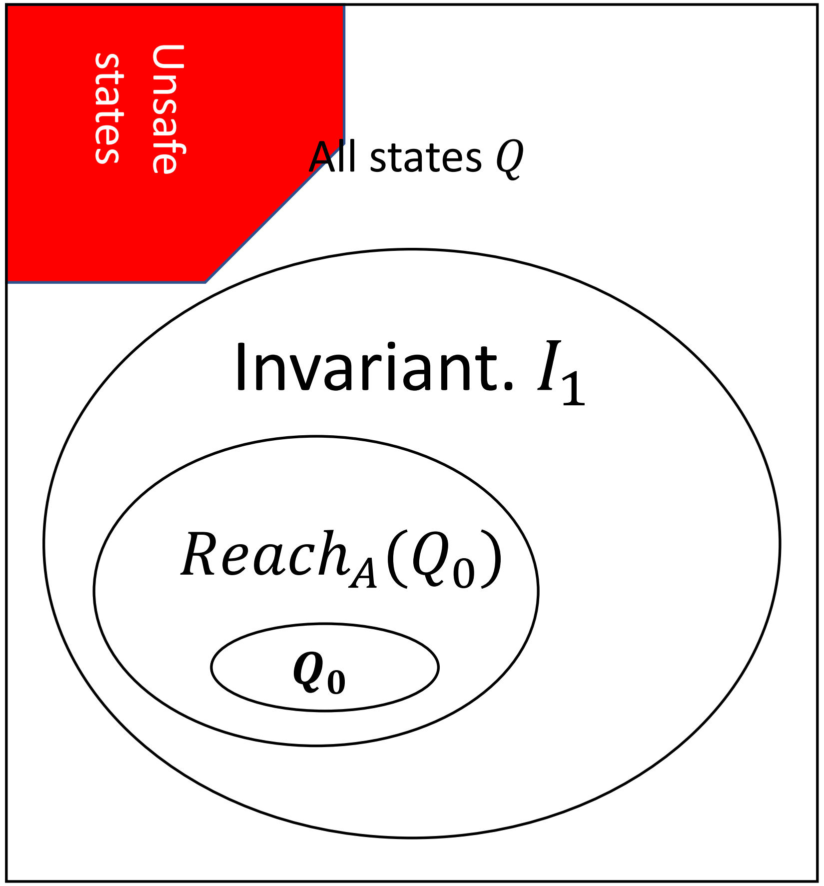
The safety verification problem can be restated as checking $\mathrm{Reach}_A \cap U = \emptyset$?

For general automata, computing $Reach_A$ is hard (undecidable)

Notice, even if we can over-approximate Reach_A that can be adequate.

Definition. An **invariant** for A is any set of states that over-approximates Reach_A. That is, $\mathrm{Reach}_A \subseteq I$.

Q is an invariant, but it is not particularly useful.



A strategy for computing $Reach_A$

Definition. $Post_A(S) = \{q' \in Q \mid \exists\, q \in S, (q', q) \in D\}$

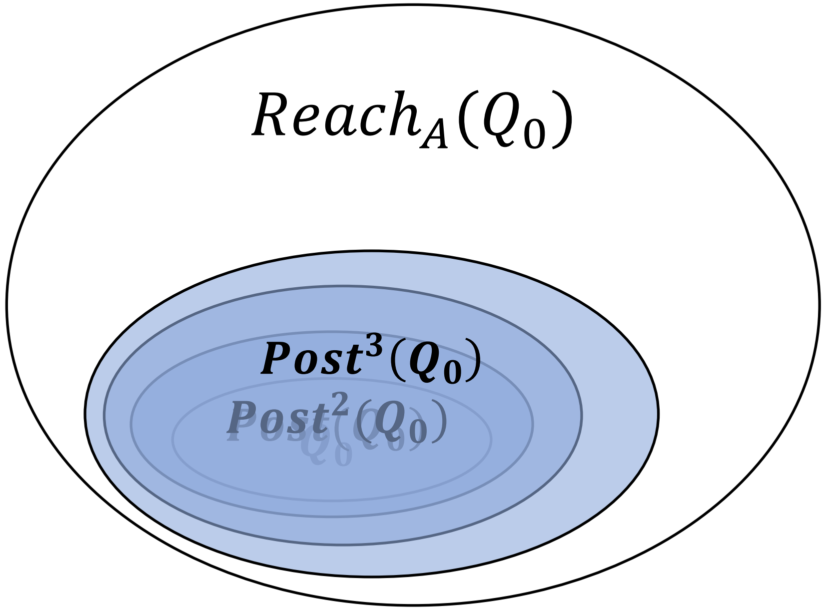
Set of all the states that can be reached from S in a single transition

Exercise. if $S_1 \subseteq S_2, Post_A(S_1) \subseteq Post_A(S_2)$ [Monotonicity]

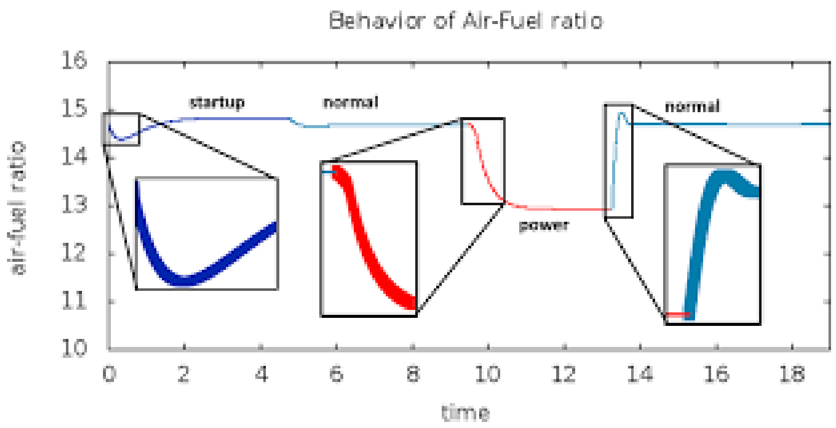
Define $Post_A^0(S) = S$ $Post_A^k(S) = Post_A(Post_A^{k-1}(S))$

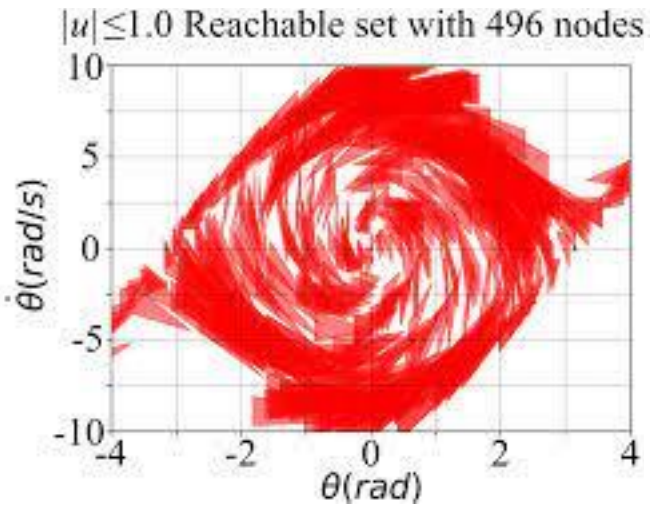
Exercise*. $Post_A^k(Q_0)$ = set of states that are reachable after k steps

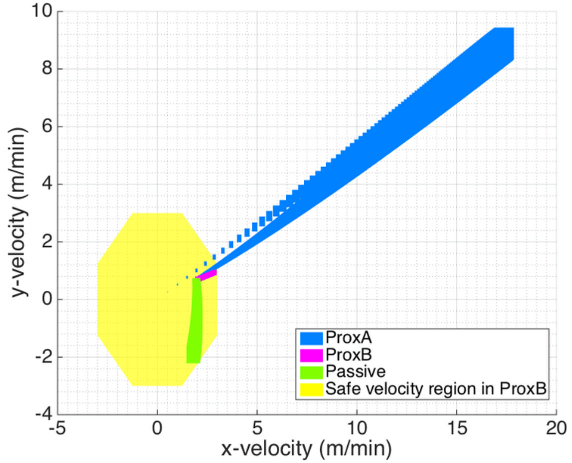
If this process converges, then we could compute $Reach_A$

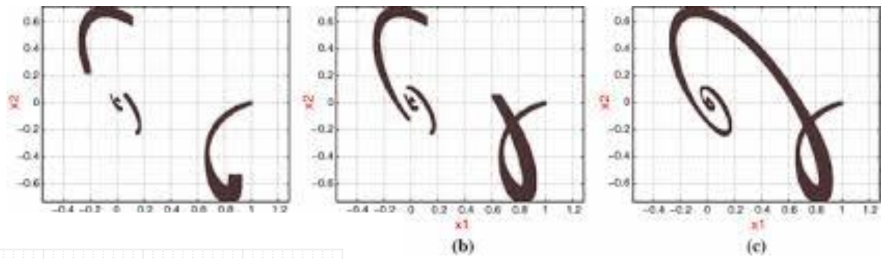


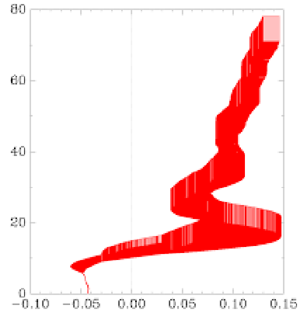
For general automata, computing $Reach_A$ is hard (undecidable)

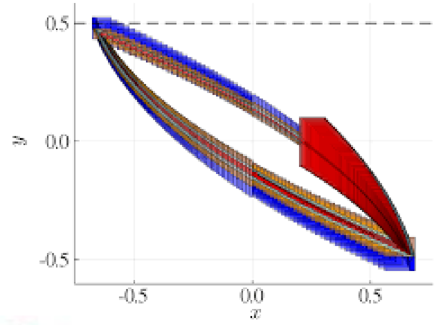


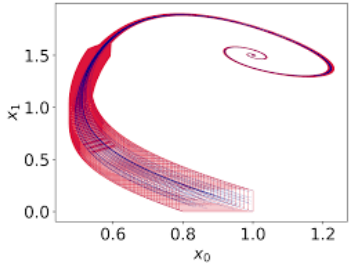












Our strategy for safety verification

- Find an invariant set of states $I \subseteq Q$ of A such that $I \cap U = \emptyset$
- How to check that a $I \subseteq Q$ is an invariant of A?

Theorem 1. Given automaton $A = \langle Q, Q_0, \mathcal{D} \rangle$ and a set of states $I \subseteq Q$ if:

- (Start condition) $\mathrm{Q}_0 \subseteq I$, and
- (Transition closure) $\mathrm{Post(I)} \subseteq I$

then I is an invariant of A. That is $Reach_{\mathcal{A}}(\Theta) \subseteq I$.

Theorem 1. Given automaton $A = \langle Q, Q_0, \mathcal{D} \rangle$ and a set of states $I \subseteq Q$ if:

- (Start condition) $\mathrm{Q}_0 \subseteq I$, and
- (Transition closure) $\mathrm{Post(I)} \subseteq I$

then I is an invariant of A. That is $Reach_{\mathcal{A}}(\Theta) \subseteq I$.

Proof. Consider any reachable state $q \in Reach_A$. We will have to show that q is also in I. By the definition of a reachable state, there exists an execution α of $\mathcal{A}$ such that $\alpha(k) = q$.

We proceed by induction on the length α

For the base case, α consists of a single starting state $\alpha = q \in \mathrm{Q}_0$, because executions always start at Q_0. And by the Start condition, $q \in I$.

For the inductive step, $\alpha = \alpha' q$ where α' is the prefix or a shorter execution. By the induction hypothesis, we know that the last state of $\alpha' say \ q' \in I$.

Invoking Transition condition on $q' \rightarrow q$ we obtain $q \in I$. QED

Back to the bouncing ball

- $I_1 = \{\langle x, v\rangle | x \leq h\}$
- Can we show that it is an invariant using the Theorem 1?
- We have to check
  - (Start condition) $\mathrm{Q}_0 \subseteq I_1$. Initially $x = h \leq h$ and $v = 0$ but does not matter \checks out
  - (Transition closure) $\mathbf{Post(I)} \subseteq I_1$
    - For any state with $x \leq h$, can we show that $x' \leq h$?
    - NO! If the velocity is positive then $x' > x$, and we cannot show the invariant
- Theorem 1 is a sufficient condition for proving invariance (not necessary)

State variables
x: $\mathbb{R}$
v: $\mathbb{R}$

State transitions

if $x \leq 0$ && $v \leq 0$

$v' = -c * v$

else $v' = v$
$v' = v - g * delta$
$x' = x + delta * v$

Back to the bouncing ball

- $I_1 = \{\langle x, v\rangle \mid v + x = h\}$
- Can we show that it is an invariant using the Theorem 1?
- We have to check
  - (Start condition) $Q_0 \subseteq I_1$. Initially $v + x = 0 + h = h$
  - (Transition closure) $\mathrm{Post(I)} \subseteq I_1$
    - For any state with $v + x = h$, can we show that $v' + x' = h$?
    - Two cases:
    - If $x > 0$ then $x' + v' = (x + v') + v - g$
- Theorem 1 is a sufficient condition for proving invariance (not necessary)

State variables
x: $\mathbb{R}$
v: $\mathbb{R}$

State transitions

if $x \leq 0$ && $v \leq 0$
$v' = -v$
$x' = v'$
else
$v' = v - g$
$x' = x + v'$

Roadmap

- A simple class of models: *automata*.
- What are executions of automata: sequence of states
- What are *requirements*?
- *Reachable states*, why we care to compute and why that can be hard
- *Invariants* as approximations of reachable states

"All models are wrong, some are useful."

Wrong and useless models

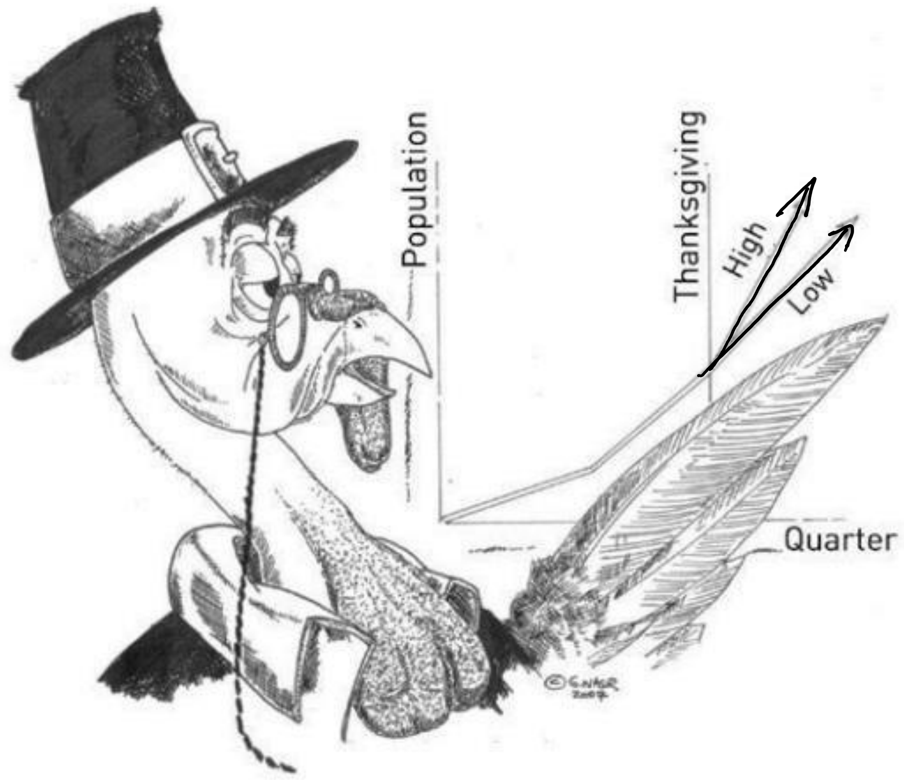
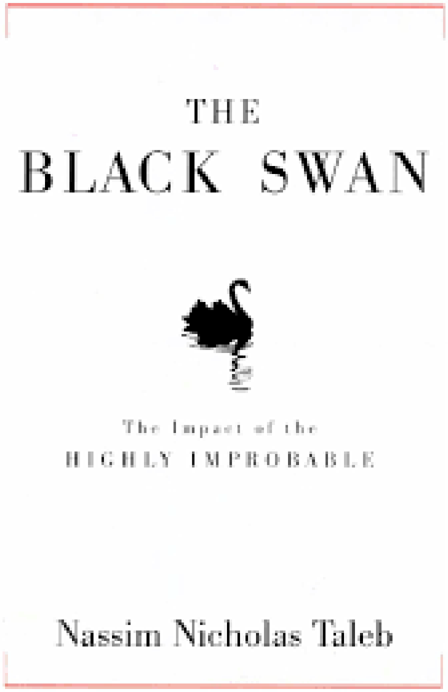
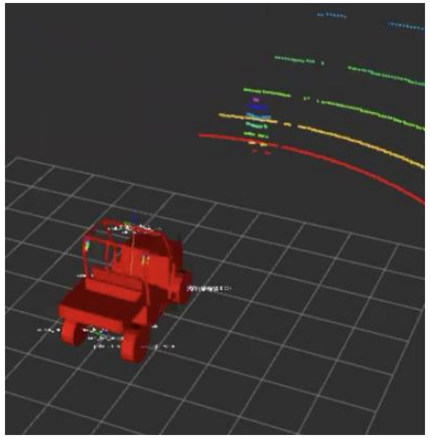


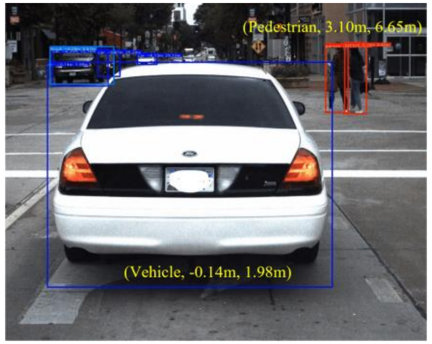
FIGURE 4. A turkey using "evidence"; unaware of Thanksgiving, it is making "rigorous" future projections based on the past. Credit: George Nasr

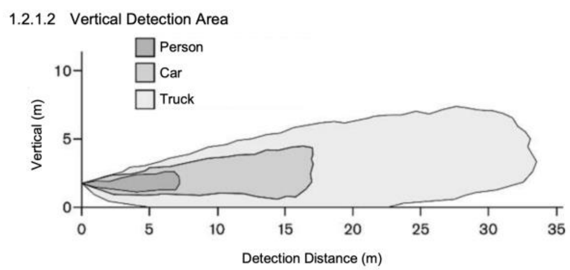


Baked-in Assumptions in our example



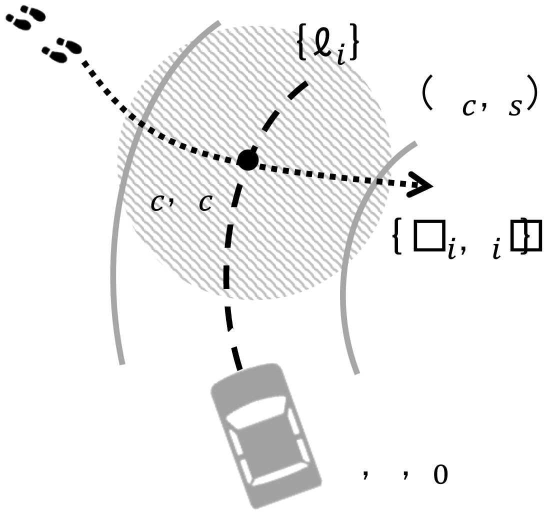
- Perception.
  - Sensor detects obstacle **iff** distance $d \leq D_{sense}$
  - No false positives, negatives, probabilities
  - Pedestrian is known to be moving with constant velocity from initial position. This will be used in the safety analysis, but not in the vehicle's automatic braking algorithm





- No sensing-computation-actuation delay.
  - The time step in which $d \leq D_{sense}$ becomes smaller is exactly when the velocity starts to decrease

Baked-in Assumptions (continued)



- 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, $x_1(t+1) = x_1(t) + v_1(t).1$
    - 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

- 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 of we have to work with infinite sets of states
- One way to compute infinite sets is using the Post operator
- But, computing all executions for unbounded time can be hard
- If we can guess an invariant satisfying conditions of Proposition 1.1, that can give a shortcut for proving safety
- The inavariant may contain important information about conserved quantities, and thus, may tell us why the system is safe, and not just that it is so
- Mind the gap between model and reality
- Next. Application of invariants in braking example