Principles of Safe Autonomy
ECE 484 Fall 2026
Lecture 3

Professor: Huan Zhang

Fall 2026

https://publish.illinois.edu/safe-autonomy/

Review

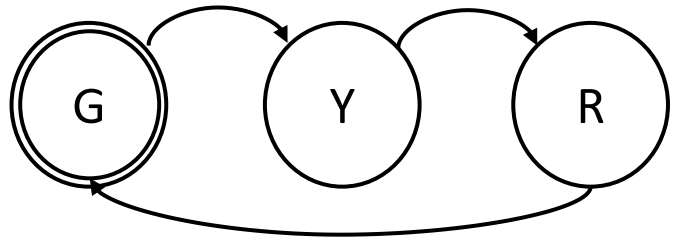
- Difference between Testing vs Verification?
- What is a model under the context of formal verification?
- What are the key components of an automaton?

Review: Automata or state machine: a **model** for computation

An **automaton** A is defined by a triple $\langle Q, Q_0, D \rangle$, where

- Q is a set of **states**
- $Q_0 \subseteq Q$ is a set of **initial states**
- $D \subseteq Q \times Q$ is a set of **transitions**

An execution of A is a finite or infinite sequence $q_0, q_1, \ldots$ such that $q_0 \in Q_0$ and, for every index i, $(q_i, q_{i+1}) \in D$



What is a requirement?

A requirement defines a set ***R*** of allowed executions

What is a *counter-example*?

An execution α that is not in the set ***R*** is a *counter-example*

Review: safety requirements

$R_{gap} = \{\alpha \mid \forall i\ \alpha_i.x_2 > \alpha_i.x_1\}$ positive gap $U_{gap} = \{\boldsymbol{q} | \boldsymbol{q}.x_2 - \boldsymbol{q}.x_1 \leq 0\}$

$R_{sp-lim} = \{\alpha \mid \forall i\ \alpha_i.v_1 \leq 70\}$ speed limit $U_{sp-lim} = \{\boldsymbol{q} | \boldsymbol{q}.v_1 > 70\}$

$R_{catch-up} = \{\alpha \mid \exists i\ \ 2 > \alpha_i.x_2 - \alpha_i.x_1 > 1\}$ catch eventually

A safety requirement says that ***every*** state along *every* execution should stay in **safe states**

Equivalently, ***no*** execution of A ever reaches an **unsafe state**

R_{gap} and R_{sp-lim} are safety requirements with U_{gap} and U_{sp-lim} as the unsafe sets

$R_{catch-up}$ is not a safety requirement

Review: Safety verification: Finite State Automata

Safety verification problem: Given an automaton A and an unsafe set U, check whether there exists any execution α of A that reaches U

$R_{gap} = \{\alpha \mid \forall i\ \alpha_i.x_2 > \alpha_i.x_1\}$ positive gap $U_{gap} = \{\boldsymbol{q} | \boldsymbol{q}.x_2 - \boldsymbol{q}.x_1 \leq 0\}$

$R_{sp-lim} = \{\alpha \mid \forall i\ \alpha_i.v_1 \leq 70\}$ speed limit $U_{sp-lim} = \{\boldsymbol{q} | \boldsymbol{q}.v_1 > 70\}$

Counter-examples of safety are finite executions ending in U

For finite automata, safety verification can be **solved using depth first search (DFS)** from Q_0

However, many problems have infinite state space. How to do safety verification?

Safety verification and Reachability: Infinite State Spaces

A state $q \in Q$ is reachable if there exists an execution α such that $\alpha_i = q$ for some index i

$Reach_A(Q_0) \subseteq Q$ the set of reachable states of *A* from Q_0

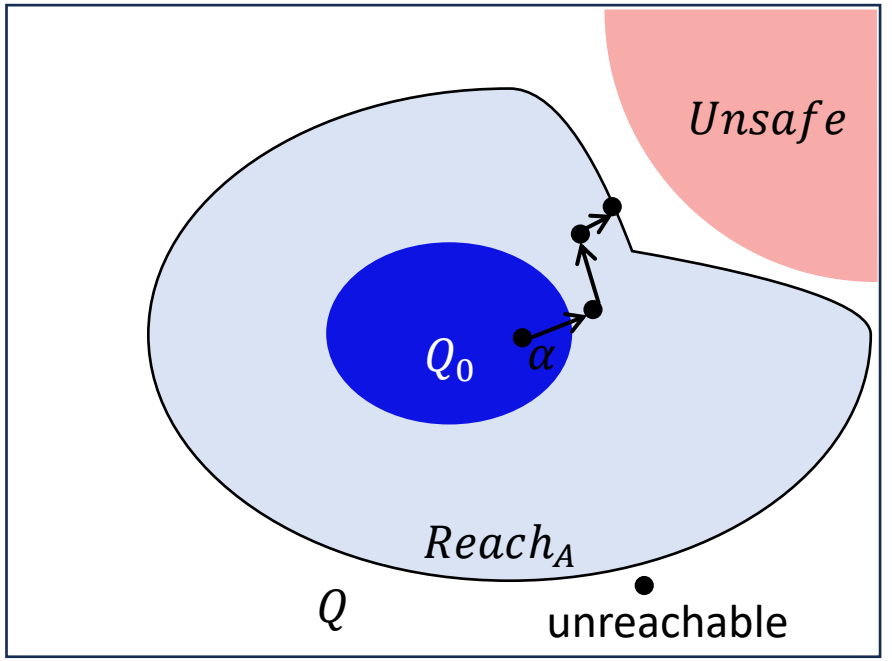
Safety verification problem is equivalent to checking $Reach_A \cap U = \emptyset$?

That is, if we can compute $Reach_A$ then we can verify safety

For finite-state systems, DFS computes $Reach_A(Q_0)$

For infinite state systems, we need:

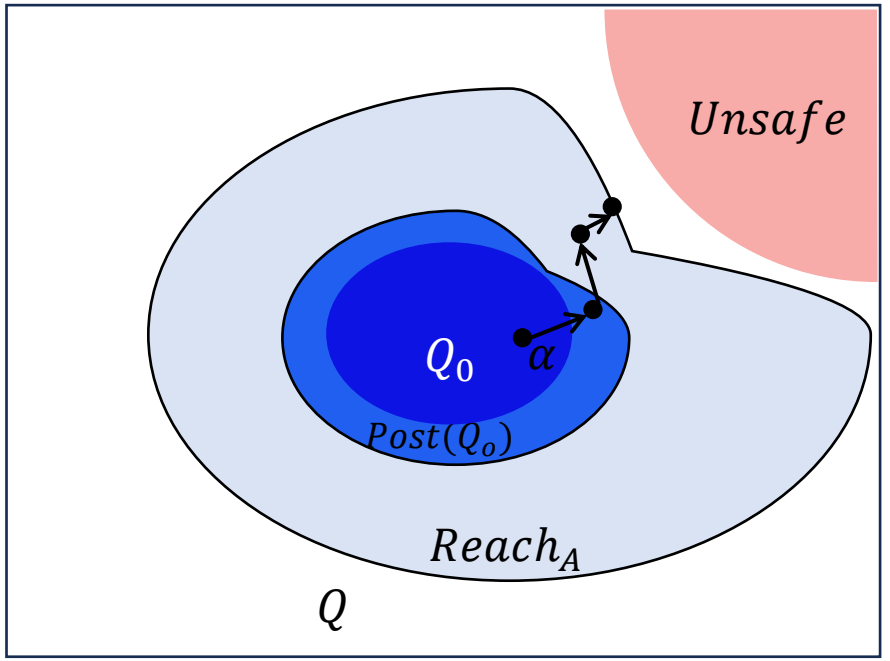
- Representation of infinite sets of states
- Iteratively computing $Reach_A$



Computing reachable sets and over-approximations

Define $Post(R) = \{q' \mid \exists\, q \in R, (q, q') \in D\}$ that gives all the states that can be reached in one step from the set of states R

- For a deterministic system $Post(\{q\}) = \{q'\}$ for $(q, q') \in D$
- For any R, $Post(R) = \cup_{q \in R} Post(\{q\})$
- $Post(Q_0) = \{q \mid \exists\, q_0 \in Q_0, (q_0, q) \in D\}$ states reachable in 1 step from Q_0
- $Post\big(Post(Q_0)\big) = ?$

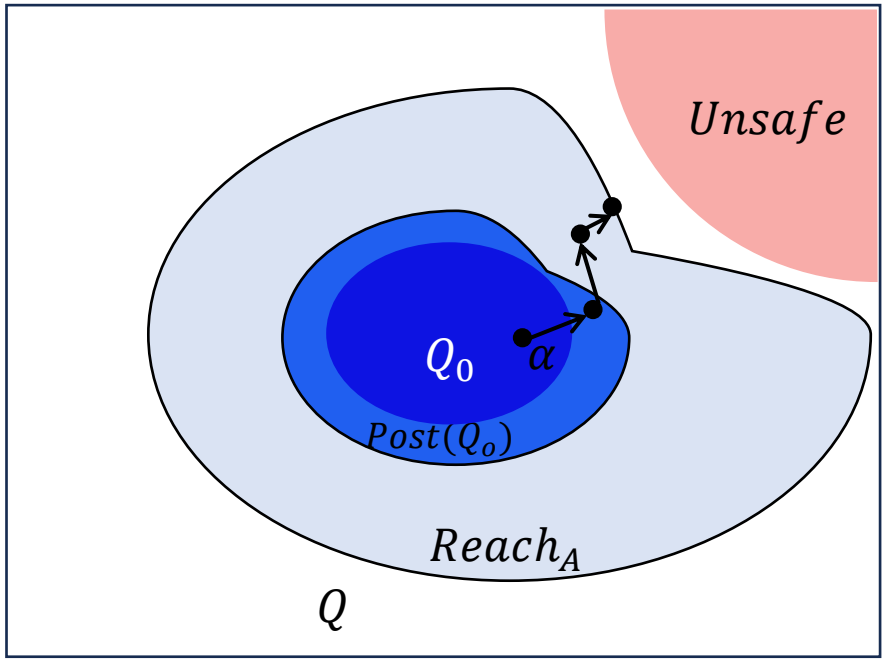


Exercise. Show that Post is monotonic, i.e., if $S_1 \subseteq S_2$ then $Post(S_1) \subseteq Post(S_2)$.

Exercise. Show that the set of states reachable in exactly k steps is $Post^k(Q_0)$.

Computing reachable sets and over-approximations

Define $Post(R) = \{q' \mid \exists\, q \in R, (q, q') \in D\}$ that gives all the states that can be reached in one step from the set of states R



Infinite sets & nondeterministic $Post(R)$ requires some representation of **sets**

Example:

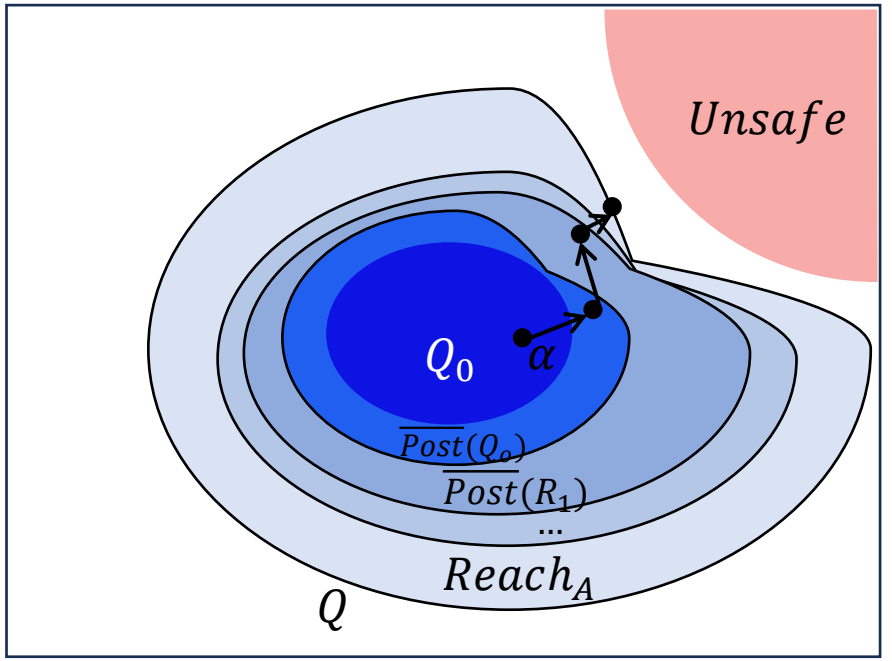
- $Q = [x_1 : \mathbb{R}]$, $D : x_1 = x_1 + v_1$ then $R = [a, b]$, $Post(R) = [a + v_1, b + v_1]$

Generally, for nonconvex R or nonlinear D exact $Post(R)$ may be infeasible

We use over-approximation $\overline{Post}(R)$ such that $Post(R) \subseteq \overline{Post}(R)$

Reachable sets and over-approximations

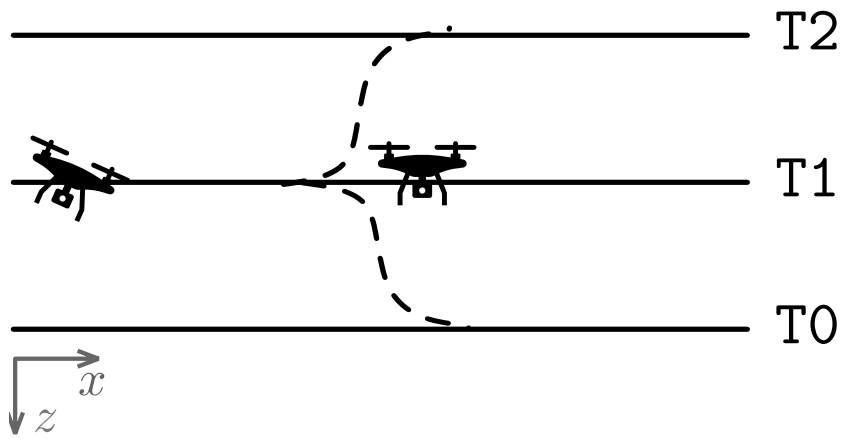
Reachability($A = \langle Q, Q_0, D \rangle$)
$R_0 = Q_0$
$i = 0$
do
  $R_{i+1} = \overline{Post}\,(R_i) \cup R_i$
  $i = i + 1$
Until $R_i = R_{i-1}$
Return R_i

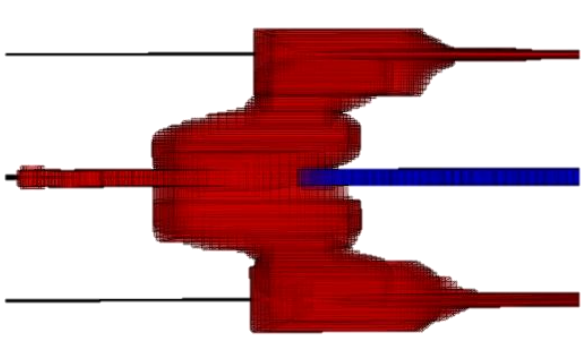


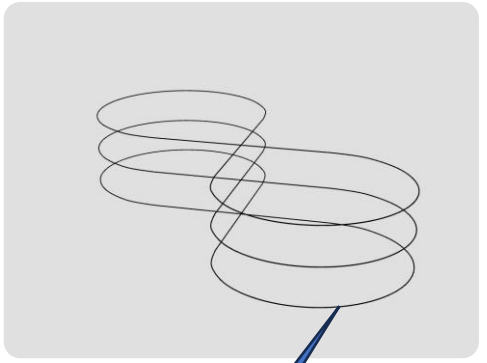
Exercise. Show that if this algorithm terminates and returns R then $Reach_A(Q_0) \subseteq R$, i.e.,
R **over-approximates** the reachable set of A. (denoted as $\overline{Reach_A}(Q_0)$)
$R \cap \text{Unsafe} = \emptyset$ proves safety, but $R \cap \text{Unsafe} \neq \emptyset$ does **not** imply that there is a real counterexample (sound, but not complete)

Verse: Python library for reachability analysis (MP0)

```
class Mode(Enum):
  Normal = auto()
  Up = auto()
  …
class Track(Enum):
  T0 = auto()
  T1 = auto()
  …
class State:
  x: float
  y: float
  …
  mode: Mode
  track: Track
```







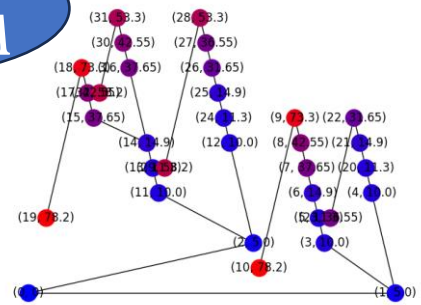
Q

D

```
def decisionLogic(ego: State, others: List[State], map):
  if ego.mode == Normal:
    if any(isClose(ego, other) for other in others):
      if map.exist(ego.track, ego.mode, Up):
        next.mode = Up
        next.track = map.h(ego.track, ego.mode, Up)
      if map.exist(ego.track, ego.mode, Down):
        next.mode = Down
        …
```

```
q1 = QuadrotorAgent(”q1", …)
q1.set_initial([…], (Mode.Normal, Track.T1))
scenario.add_agent(q1)
q2 = …
scenario.set_map(M5())
scenario.simulate(…)
scenario.verify(…)
```

$Reach_A$



$Unsafe$

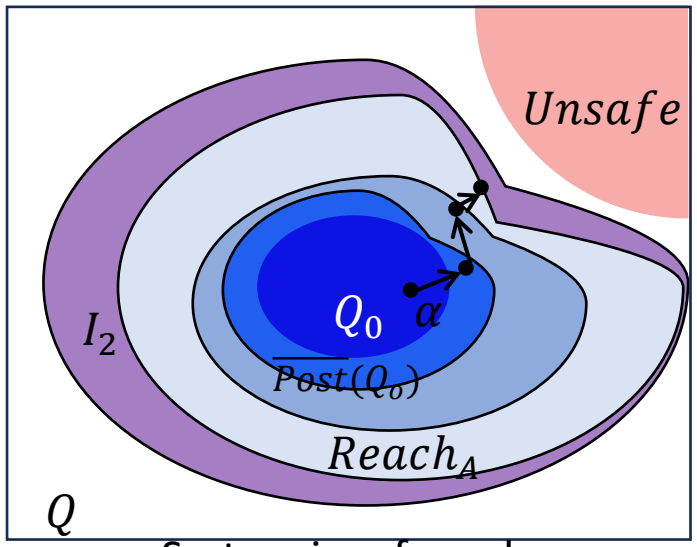
```
assert not any(isVeryClose(ego, other) for other in others), "Separation"
```

Invariants and safety verification

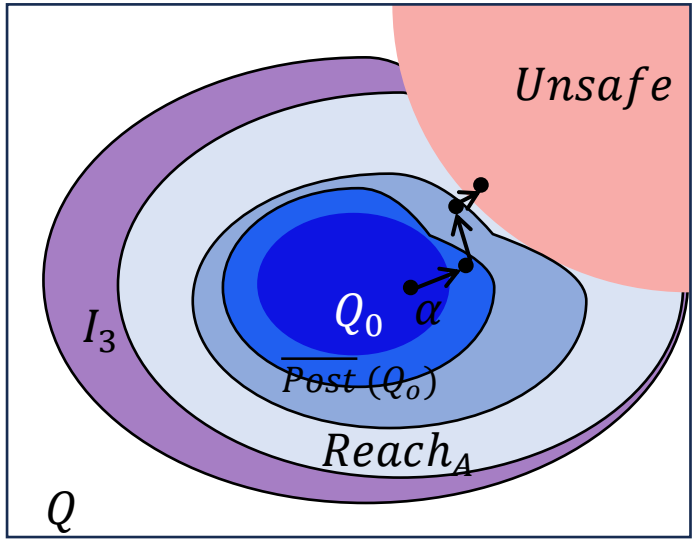
A set $I \subseteq Q$ is an invariant if $Reach_A(Q_0) \subseteq I$

Over-approximates the reachable states, not unique, and contains everything that *can* happen

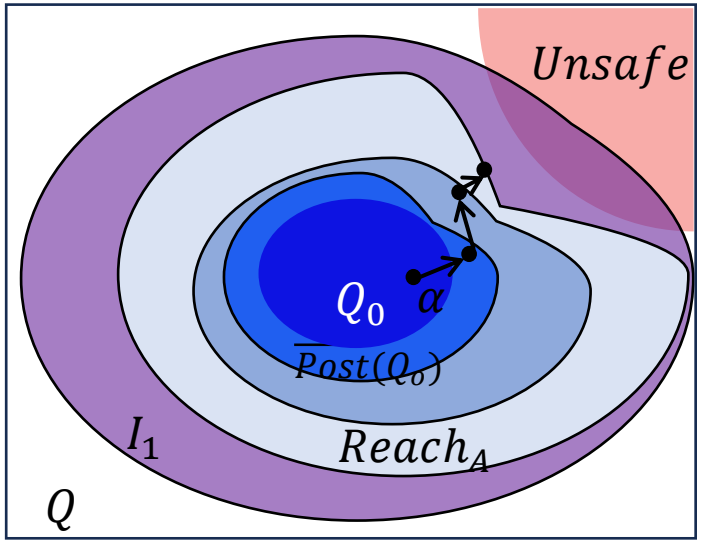
If the algorithm terminates, it returns *an* invariant which may or may not prove safety



System is safe and verified by invariant I_2



$I_3 \cap Unsafe \neq \emptyset$ and system is unsafe



$I_1 \cap Unsafe \neq \emptyset$ but system is safe

Inductive invariants

Proposition 1. If (i) $\mathrm{Q_0} \subseteq I$ and (ii) $Post(I) \subseteq I$ then I is an invariant, i.e., $\mathrm{Reach_A} \subseteq I$.

Such invariants are called **inductive** invariants

Proof. Consider any reachable state $\boldsymbol{q} \in Reach_A \subseteq Q$

By definition of reachable state, there is an execution α with $\alpha_k = \boldsymbol{q}$

By induction on k we will show that $\boldsymbol{q} \in I$

Base case, for k=0, $\alpha_0 = q_0 \in Q_0 \subseteq I$ [using definition of execution and (i)]

Induction. By inductive hypothesis, suppose $\alpha_k \in I$. We have to show $\boldsymbol{q} = \alpha_{k+1} \in I$.

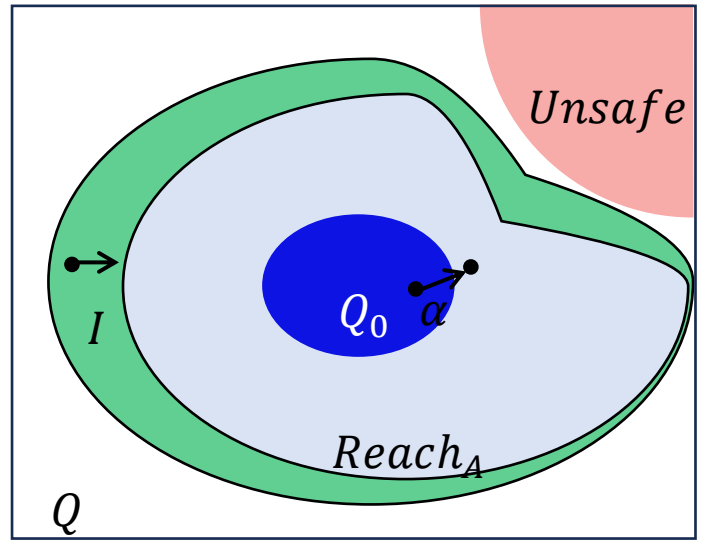
$\boldsymbol{q} \in Post(\alpha_k)$ [Definition of Post, $(\alpha_k, \boldsymbol{q}) \in D$]

$\boldsymbol{q} \in Post(I)$ [Monotonicity of Post. $\alpha_k \in I \Rightarrow Post(\alpha_k) \subseteq Post(I)$]

$\boldsymbol{q} \in I$ [By (ii)]

Using Inductive invariants to prove Safety

- Guess a candidate inductive invariant I
- If $\mathrm{I} \cap Unsafe = \emptyset$ and $\mathrm{Q}_0 \subseteq I$ and $Post(I) \subseteq I$ then by the Proposition 1, $\mathbf{Reach}_A \subseteq I$ and we have verified safety
- If the transition condition Post(I) ⊆ I fails, that does *not* imply that I is not an invariant
- It only implies that I cannot be checked *inductively* by Proposition 1.
- If the start condition fails, $\mathrm{Q}_0 \not\subset$ I, then I is certainly NOT an invariant, because $\mathrm{Q}_0 \subseteq \mathrm{Reach}_A$.



System is safe and verified by the inductive invariant

Revisiting AEB: how to verify safety?

To prove no crash $x_2 > x_1$ in all reachable states, we will need assumptions about initial conditions ($x_{10}, x_{20}, v_{10}, v_{20}$), sensing distance ($d_s$), and braking acceleration (a_b)

Discovering these assumptions (for system correctness) is a valuable side-effect of verification

Assumption: $x_{20} - x_{10} > d_s > \frac{v_{10}^2}{a_b}$ and $a_b \leq v_{10}$ (ensure $v_1 \geq 0$)

"initial distance is greater than d_s, and breaking is strong enough"

The proof of correctness (as expected) will relate total time of braking with the initial separation. We need a braking timer.

Checking Inductive Invariant for AEB

Transition relation D
(written as a program)

timer = 0
If $x_2 - x_1 \leq d_s$
  If $v_1 \geq a_b$
    $v_1 = v_1 - a_b$
    timer := timer+1
  else
    $v_1 = 0$
else
  $v_1 = v_1$
$x_2 = x_2 + v_{20}$
$x_1 = x_1 + v_1$

Invariant. I_1: $v_1 \geq 0 \wedge \text{timer} + \frac{v_1}{a_b} \leq \frac{v_{10}}{a_b}$. Bound on total braking time in terms of velocity and deceleration

Proof. We need to check two conditions for this to be an inductive invariant: (i) $Q_0 \subseteq I_1$ and (ii) $Post(I_1) \subseteq I_1$.

(i) Consider any $q \in Q_0$. We need to show $q \in I_1$.

$q.timer + \frac{q.v_1}{a_b} = 0 + \frac{v_{10}}{a_b} \leq \frac{v_{10}}{a_b}$.

(ii) Consider any $(q, q') \in D$ with $q \in I_1$. We need to show $q' \in I_1$.

As there are three branches in D, there are 3 cases.

(a) $q'.timer + \frac{q'.v_1}{a_b} = q.timer + 1 + \frac{q.v_1 - a_b}{a_b} = q.timer + \frac{q.v_1}{a_b} \leq \frac{v_{10}}{a_b}$

(b) $q'.timer + \frac{q'.v_1}{a_b} = q.timer + 0 \leq q.timer + \frac{q.v_1}{a_b} \leq \frac{v_{10}}{a_b}$

(c) $q'.timer + \frac{q'.v_1}{a_b} = q.timer + \frac{q.v_1}{a_b} \leq \frac{v_{10}}{a_b}$

Given the *inductive* invariant I_1, we know I_2: $\text{timer} \leq \frac{v_{10}}{a_b}$ is an invariant

Invariants and assumptions give correctness proof

Consider any two reachable states:

q_1 is the last state with $x_2 - x_1 > d_s$ (braking starts at the next step), and

q_2 is reached from q_1 with $q_2.x_2 - q_2.x_1 \leq d_s$ (other reachable states are safe)

$q_2.x_2 - q_2.x_1$

$\geq q_1.x_2 - q_2.x_1$ [Because x_2 does not decrease, as $v_{20} \geq 0$]

$\geq q_1.x_2 - (q_1.x_1 + v_{10}.\frac{v_{10}}{a_b})$ [$I_2 \Rightarrow$ timer $\leq \frac{v_{10}}{a_b}$ and $v_1 \leq v_{10}$]

$> d_s - \frac{v_{10}^2}{a_b}$ [By def of q_1]

> 0 [By Assumption]

Distance always positive, so this AEB is safe!

Summary

- Testing alone is inadequate---in theory and practice
- Automaton (state machine) models, executions, and requirements give us the language to state correctness claims precisely
- Verification is the problem of proving/disproving such claims
- Safety claims are a (prevalent) subset of correctness claims
- Reachability analysis can prove/disprove safety
- In general, reachability and verification are hard (state space explosion, undecidability)
- Inductive invariants over-approximating reachable states give a practical method for proving safety