ECE 498SM: Principles of Safe Autonomy Fall 2020

Lecture 1 — DATE, 2020
Proving safety by induction

Sayan Mitra *Scribe: YOUR NAME*

In this section, we will develop a simple model of a vehicle and discuss how we can use basic induction to prove its safety. As a by-product of this modeling and analysis, we will identify and discover the precise assumptions we are making about the vehicle's sensor, algorithms, and mechanics.

1 Modeling a scenario

Consider a vehicle cruising down a straight road that that needs to stop when an obstacle (say a pedestrian) is detected by its is *range sensor* (e.g., a LIDAR). For autonomous vehicles, this is a basic but nontrivial requirement [**?**].

Let us try to describe this scenario with a mathematical model. For any model, first we need to first identify the *state variables*. The values of these variables will define the state of the system. For this scenario, it makes sense to have the position x_1 and the velocity v_1 of our vehicle as state variable. Let us also include the position x_2 and the velocity v_2 of the pedestrian. (Note. We are assuming there is one pedestrain/obstacle ahead of the vehicle).

With these and a few other variables in place we can write down a *discrete time* (program-like) model of the system as follows. This model defines how the state of the vehicle (and its environment) changes in each *time step*.

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
    $v_1 = 0$
$x_1 = x_1 + v_1$

Although simple, one-dimensional scenarios are commonly used safety assurance cases for vehicles and even aircraft [ISO11, Fab12, PMTPB13, DWM+14].

1.1 Assumptions baked into the model

1. *Perception.*
   (a) Sensor s detects the obstacle **iff** distance $d \leq D_{sense}$. No false-positives, no false-negatives, no probabilities. This is a very idealized model of a sensor. More realistic sensor specifications will give distances and probabilities specific to detecting particular objects like people, cars, bicycles; detection zone will be directional.
   (b) The pedestrian is known to be moving with constant velocity $v_0 = 0$ from initial position x_{20}. This assumption will be heavily used in the safety analysis, but it is not used the vehicle's automatic braking algorithm.
2. *No sensing, computation, actuation delay.* The time step in which $d \leq D_{sense}$ becomes smaller is exactly when the velocity starts to decrease.
3. *Mechanics.*
   (a) Vehicle and pedestrian moving in 1-D lane.
   (b) Does not go backwards.
   (c) Perfect discrete kinematic model for velocity and acceleration.
4. *Nature of time.*
   (a) *Discrete time steps.* Each execution of the above function models advancement of time by *1 step*. For convenience, here we consider 1 step = 1 second, and hence $x_1(t+1) = x_1(t) + v_1(t)$. If 1 step = Δ seconds then we'd have $x_1(t+1) = x_1(t) + v_1(t)\Delta$. Either way, we are not allowed to talk about what happens between $[t, t+1]$.
   (b) *Atomic steps*.* We consider 1 step to be the complete (atomic) execution of the program. We cannot directly talk about the states that the program visits as it executes the individual lines/statements in the program.

2 Defining Safety

2.1 What does safety mean?

Our modest goal for now is to show that the car does not collide with the pedestrian. That is $d := x_2 - x_1 > 0$. Such statements about behaviors of a system are called *requirements*.

Definition 2.1. *Requirements* are precise statements about what the behaviors of the system should or should not do.

The staement $d > 0$ is not quite precise, yet. In this case, implicitly we want this requirement $d > 0$ to hold *always*. Requirements that are supposed to hold *always* or at all times are called *invariants*.

Definition 2.2. An *invariant* is a requirement of the form, always A, where A is some condition on the variables of the system.

For example, we could write the following invariant:

Invariant 2.1. $\forall t, d(t) > 0$

Here we are using the convention that $d(t)$ is the *valuation* of the state variable at time t. That is, $d(0) = x_2(0) - x_1(0) = x_{20} - x_{10}$, $d(1)$ is the value of d after the program is executed once, $d(2)$ after the program is executed a second time, and so on. Similarly we could talk about $x_1(t)$, $x_1(t+1)$, etc.

There is something still missing in Invariant 2.1. Notice that SimpleCar is simple, but it still models an infinite family of scenarios—one for each possible tuple of values for the parameters $D_{sense}, v_0, x_{10}, x_{20}$, and a_b. So, our safety claim should really be:

Invariant 2.2. $\forall D_{sense}, v_0, x_{10}, x_{20}, a_b, t$: $x_{20} > x_{10} \Rightarrow d(t) > 0$.

It is easy to see that this statement does not hold. Consider the situation: $D_{sense} = x_{20} - x_{10} - \varepsilon$ and $v_0 = x_{20} - x_{10} + \varepsilon$, for some positive ε. Then we have the following behavior:

$$\begin{aligned} x_1(0) = x_{10};\ x_2(0) = x_{20}; \qquad & d(0) = x_{20} - x_{10} > D_{sense};\ v_1(0) = v_0 \\ x_1(1) = x_{10} + v_0;\ x_2(1) = x_{20}; \qquad & d(1) = x_{20} - x_{10} - v_0 = -\varepsilon \end{aligned}$$

which violates Invariant 2.3. Clearly, we would want to add some assumptions to Invariant 2.3 so that the modified version of it holds.

Invariant 2.3. $\forall D_{sense}, v_0, x_{10}, x_{20}, a_b, t$:
if $x_{20} > x_{10}$ *and* some assumptions on $D_{sense}, v_0, x_{10}, x_{20}, a_b$ hold
then $d(t) > 0$.

Identifying these assumptions will be a key product of the safety analysis enterprise.

Remark 2.1. $x_1(1) = x_{10} + v_0$ because at the beginning of step 1, $d = d(0) > D_{sense}$. This is a subtle point related to atomic execution of our program model described earlier $4(b)$. Notice also that at the end of step 1, $d = d(1) = -\varepsilon < D_{sense}$, but by that time it is too late.

3 Explicit model of the scenario

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

$s(t+1) = 0$
15 $v_1(t+1) = v_1(t)$
$timer(t+1) = timer(t)$
17 $x_1(t+1) = x_1(t) + v_1(t)$

Let us call this model $\mathcal{A}$. Notice that fixing values of $D_{sense}, v_0, x_{10}, x_{20}, a_b$, the above function defines for any possible state (*prestate*)—$x_1(t), x_2(t), v_1(t), s(t)$—a unique *poststate* $x_1(t+1), x_2(t+1), v_1(t+1), s(t+1)$. This is because the function is *deterministic*. There is no uncertainty. We will have a lot more to say about uncertainties, nondeterminism, and probabilities later in this course.

An *execution* of $\mathcal{A}$ is a sequence of states $\alpha = \mathbf{x}(0), \mathbf{x}(1), \ldots$, where (a) $\mathbf{x}(0)$ satisfies the **initially** statement and (b) for each t, $\mathbf{x}(t)$ and $\mathbf{x}(t+1)$ are related according to the function defined by SimpleCar.

For any state $\mathbf{x}$ or $\mathbf{x}(t)$ of $\mathcal{A}$, we refer to the corresponding components by $\mathbf{x}.x_1, \mathbf{x}.v_1, \mathbf{x}.s$, etc., and $\mathbf{x}(t).x_1, \mathbf{x}(t).v_1, \mathbf{x}(t).s$, etc., respectively.

Now that we have defined the model $\mathcal{A}$ explicitly and everything is absolutely precise, we can make some symbols implicit for the sake of brevity. Invariant 2.3 can be rewritten more succinctly as

Invariant 3.1. *if $x_{20} > x_{10}$, then over any execution $\mathbf{x}(0), \mathbf{x}(1), \ldots \mathbf{x}(t)$ of $\mathcal{A}$, $\mathbf{x}(t).d > 0$.*

When we say $x_{20} > x_{10} \Rightarrow d > 0$ this is really what we mean.

4 Safety analysis

Assumption 4.1. $D_{sense} \geq v_0^2/a_b$

Invariant 4.1. *Over all executions of $\mathcal{A}$, $timer + v_1/a_b \leq v_0/a_b$.*

Proof. Base case.

$$timer(0) + v_1(0)/a_b = 0 + v_0/a_b.$$

Inductive step. We assume that for a given t, $timer(t) + v_1(t)/a_b \leq v_0/a_b$ (*induction hypothesis*) and we have to show that $timer(t+1) + v_1(t+1)/a_b \leq v_0/a_b$. There are really three cases to consider based on the three branches of the SimpleCar program.

if $d(t) \leq D_{sense}$ and $v_1(t) \geq a_b$. From line 7-9,

$$\begin{aligned} timer(t+1) + v_1(t+1)/a_b & \\ &= timer(t) + 1 + (v_1(t) - a_b)/a_b \\ &= timer(t) + v_1(t)/a_b \\ &\leq v_0/a_b. \end{aligned}$$

if $d(t) \leq D_{sense}$ and $v_1(t) < a_b$. From line 11, $timer(t+1) + v_1(t+1) = timer(t) + 0 \leq v_0/a_b$. Why? Because from the induction hypothesis we know:

$$\begin{aligned} timer(t) + v_1(t)/a_b &\leq v_0/a_b \\ timer(t) &\leq (v_0 - v_1(t))/a_b \\ &\leq v_0/a_b. \end{aligned}$$

Last step holds because $\forall t, v_1(t) \geq 0$ (This is an invariant which can be proved first). We also know that in this case $v_1(t) < a_b$, but that fact is not used.

if $d(t) > D_{sense}$. This is the simplest case as $v_1(t+1) = v(t)$ and $timer(t+1) = timer(t)$, it follows that $timer(t+1) + v_1(t+1)/a_b \leq v_0/a_b$ from induction hypothesis.

□

Since $v_1 \geq 0$ is also an invariant of $\mathcal{A}$, from Invariant 4.1, the following invariant is implied.

Invariant 4.2. *Over all executions of $\mathcal{A}$, $timer \leq v_0/a_b$.*

Exercise 4.1. Can you prove Invariant 4.2 using the inductive method used to prove invartiant 4.1?

At this point most of you can probably "see" this line of reasoning: Since $v_1(t) \leq v_0$, the total distance the vehicle traverses while the condition $v_1 \geq a_b$ holds is at most $v_0 \times v_0/a_b$. Therefore, if $x_{20} - x_{10} \geq D_{sense} > v_0^2/a_b$, then the vehicle comes to a halt before the d drops down to 0.

Here is a candidate invariant:

Invariant 4.3. *Over all executions of $\mathcal{A}$, if $x_{20} - x_{10} \geq D_{sense}$ and $D_{sense} > v_0^2/a_b$ then $d > 0$.*

Exercise 4.2. Can you prove Invariant 4.3 using the inductive method used to prove invartiant 4.1 and any of the invariants proved earlier? If you cannot prove this invariant directly, then prove other intermediate invariants or strengthen the assumptions to prove safety, i.e., $d > 0$.

Exercise 4.3. Let us introduce some delay in the sensing-computaion-actuation pipeline, say T_{react}. This could model cognitive delay of a human driver or processing delay in electronics and computers. (a) Create a new version of SimpleCar that introduces exactly T_{react} delay from the time of sensing. (b) Create another model that introduces *at most* T_{react} delay.

Exercise 4.4. How should Assumption 4.1 be updated to maintain safety with the exact delay model?

Exercise 4.5. Allow the pedestrian to move with a range of non-zero velocities $v_2 \in [v_{2min}, v_{2max}]$. How should Assumption 4.1 be updated to maintain safety with the exact delay model?

5 A simulator for the scenario

References

[DWM+14] Parasara Sridhar Duggirala, Le Wang, Sayan Mitra, Mahesh Viswanathan, and César A. Muñoz. Temporal precedence checking for switched models and its application to a parallel landing protocol. In *FM'14: Proceedings of the 19th International Symposium on Formal Methods*, pages 215–229. Springer, 2014.

[Fab12] Simone Fabris. Method for hazard severity assessment for the case of undemanded deceleration. Technical report, TRW Automotive, 2012.

[ISO11] ISO. Road vehicles—functional safety. Technical Report ISO 26262, International Organization for Standardization (ISO), 2011.

[PMTPB13] Raleigh B. Perry, Michael M. Madden, Wilfredo Torres-Pomales, and Ricky W. Butler. *The simplified aircraft-based paired approach with the ALAS alerting algorithm*. Technical Report NASA/TM-2013-217804, NASA, Langley Research Center, 2013.