Car Interlock (Spec)

Jonathan Ostroff (EECS)

October 29, 2018

Revisions

Date	Revision	Description
12 September 2016	1.0	Initial write-up with invariant
26 September 2017	1.1	Added appendix for instructors
29 October 2018	1.2	Minor edits for TR

Contents

List of Figures

On your own, please do all the parts that are highlighted.

1. Introduction

In the Car Interlock Problem (Fig. 1), you describe the requirements as a complete and disjoint function table and validate the requirements with a *safety invariant* and a *use case*. This is also the first complete requirements document that you write and submit (atomic requirements, list of monitored/controlled variables, function table specification and validation of the function table).

Also note, that in a requirements document you should track revisions. We do so in this document.

2. Informal Specification

Our system under description is a small example.

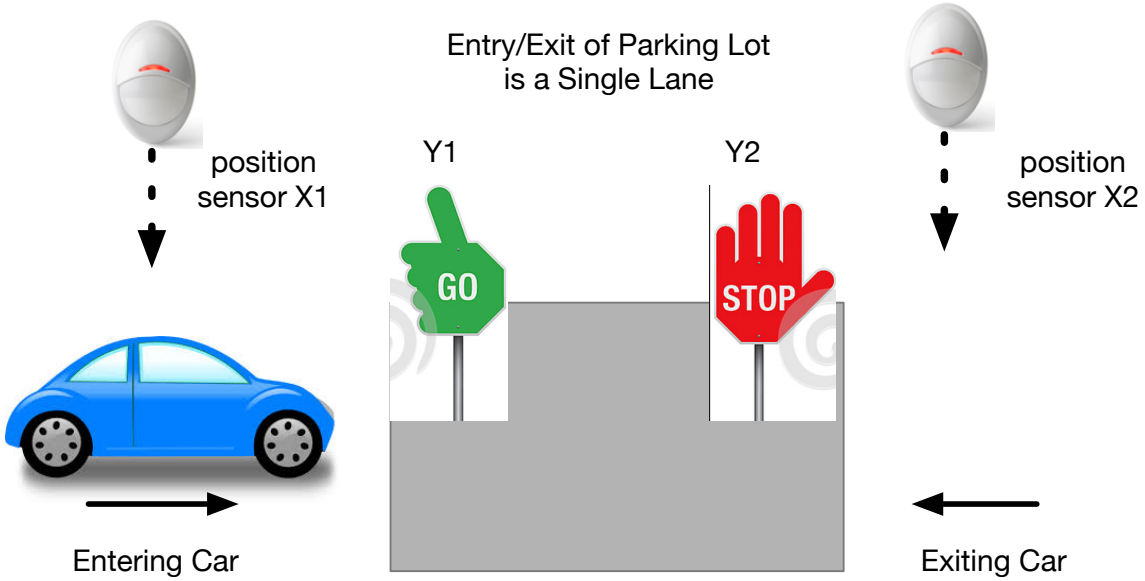

Figure 1: Car Interlock

The entry/exit to a parking lot is a single lane passage. By controlling the indicators, ensure that only one car can pass through the entry/exit at a time so as to prevent car accidents between entering and leaving cars.

In T-ASM theory we might write the specification as shown in Listing 1. You must complete the function table in the PVS specification shown in the listing.

```
car_interlock[delta:posreal]: THEORY
BEGIN
  importing Time[delta]
  i: VAR DTIME
  SENSOR: TYPE   = {on, off}
  ACTUATOR: TYPE = {stop, go}
  x1, x2: [DTIME -> SENSOR]
  y1, y2: [DTIME -> ACTUATOR]

  % Function Table. TBD
  control_ft(i): bool =
    COND
    i = 0 -> y1(0) = stop AND y2(0) = stop,
    i > 0 ->
      % TBD
    ENDCOND

  % Invariant
  inv(i): bool =
    NOT (y1(i) = go AND y2(i) = go)

  inv_holds: CONJECTURE
     (FORALL i: control_ft(i))
    =>
     (FORALL i: inv(i))
```

Table 1: PVS Specification

Obviously your function table must be proved complete and disjoint and will thus use the COND construct.

Note: Ensure that you do not specify a circular function table. You cannot observe and change a control variable at the very same time.

Note: There is no need to instantiate δ (associated with the Timing Resolution TR). Thus, our theory allows for an arbitrarily small or large δ.

Our customer has allowed us to assume that cars take no longer than 10 seconds to cross the shared area from the time that a car leaves a sensor, i.e. cars are assumed not to halt in the shared area (our customer is not willing to add extra sensors due to the cost; if cars do halt then crashes must be avoided by drivers who can clearly see the offending car). This case (of offending cars) is an example of an environmental

```
 % use case
   assume: bool =
         control_ft(0)
     AND control_ft(1)
     AND control_ft(2)
     AND control_ft(3)
     AND x1(1) = on AND x2(1) = on AND y1(0) = stop
     AND x1(2) = on AND x2(2) = off
     AND x1(3) = on AND x2(3) = off

 %% exit car goes first, then entrance car
   use_case: CONJECTURE
      assume
     =>
           y2(1) = go AND y1(1) = stop
       AND y1(2) = stop AND y2(2) = stop
       AND y2(3) = stop AND y1(3) = go
END car_interlock
```

Table 2: USe Case

phenomenon over which the computer system (under development) has no control. In any event, the actuators should never both show the green light go simulatanously! On these assumptions it appears that sampling the sensors every 10 seconds may be sufficient for this control system.

There are various requirements that you must capture in your function table.

Your function table can be validated with a system invariant (`inv_holds`) that ensures that both directions cannot travel at the same time. Why?

You must also ensure in your function table that cars waiting to exit go before cars waiting to enter. Why? To ensure that your function table satisfies this property, you must prove the Use Case shown in Listing 2.

3. Writing a Requirements Document for the Car Interlock

You must write a requirements document for the Car Interlock system.. The instructions below should help.

- Where is the System Boundary?
- What are the monitored variables? What are their types?
- What are the controlled variables? What are their types?
- Specify a complete and disjoint function table that describes the input/output behaviour of the SUD? (You must actually draw this table in your requirements document; the PVS specification is insufficient.
- Now (a) write out the atomic E/R-descriptions for the plant (number them) (b) state what the monitored variables are and in a different table what the controlled variables are and (c) draw the function table for the car interlock system and provide evidence that you have validated the function table.
- Submit your document as `car-interlock.pdf`.
- We will be using Latex to prepare documentation for the assignment and project. You may try to prepare the document using Latex. See `https://wiki.eecs.yorku.ca/project/sel-students/p:tutorials:latex:` (login). There is a link to a Latex table generator. Also, you may use the Project template (in the SVN) which has all the relevant parts needed for a Requirements Document.
- You are not absolutely required to use Latex for this Lab but it is highly recommended as it will be needed for the Assignment and the Project.

Ensure that it is neat! Ensure that it is minimal! It does not require more than half a page in a large font. Too many cooks spoil the broth (i.e. too many rows and not enough organization and thought spoil the function table).

4. What will be in your document

Understand and agree upon what users want before attempting to create solutions.

Finding out what is needed instead of rushing into presumed solutions is the key to every aspect of system development. Most technical problems can be solved, given determination, patience, a skilled team—and a well-defined problem to solve.

A precise requirements document will contain all the information needed by the developers to build the system wanted by the users—and no more (i.e. it should not be polluted with design and implementation detail).

It is important to write the informal requirements atomically (with a number to track the requirements), e.g.

REQ1	The traffic stop/go lights shall not both be green at the very same time

The above requirement will be checked using the invariant conjecture in PVS.
Your document will contain:

- Informal statement of the problem
- Context diagram
- Table of monitored variables with its types and another table of controlled variables with their types.
- Atomic requirements.
- The function table that is complete and disjoint.
- Various use cases (one is enough for this introductory document)
- The PVS specification of the function table and validation of completeness/disjointness, invariants and use cases.

A. Appendix