Reachability, monitors

These exercises are mostly taken from Rajeev Alur and Thomas A. Henziger lectures on computer aided verification.

Exercise 1 — Railroad

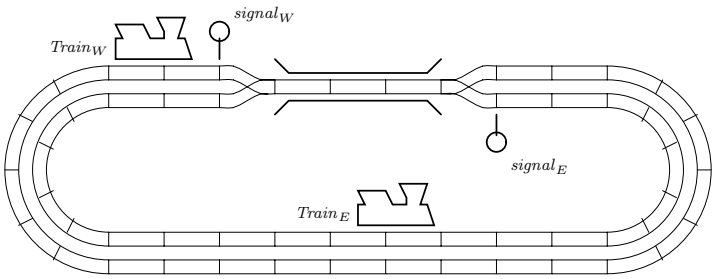

Figure 1 – A railroad

Figure 1 represents a railroad. It is composed of two tracks, one for trains traveling clockwise and the other one for trains traveling counterclockwise. The two tracks share a bridge. At both entrance (east and west) of the bridge are signals that allow or not trains to cross the bridge. Trains are modeled by the modules of Figures 2 and 3.

The goal is to design a controller module which prevents collisions between the trains, that is, which ensure that in all rounds at most one train is on the bridge. A proposal is the module of Figure 4.

This exercise aims at studying the system built from the trains and the controller, checking if it prevents collisions.

1. How many states does the module $Train_W \| Train_E \| Controller_2$ have ?

2. How many of these states are reachable ? To answer this question draw the reachable subgraph of the transition graph.

3. Is there a reachable state with both trains on the bridge ?

4. What can you say about the fairness of this system ?

5. Recall what is a monitor.

6. Why is a monitor required to check the fairness of the railroad system studied ?

7. Propose a module for monitoring the fairness of the railroad system with respect to the train traveling clockwise.

8. Use it to monitor the fairness of the railroad system.

module *Train* **is**
interface $pc : \{away, wait, bridge\};\ arrive, leave : \mathbb{E}$
external $signal : \{green, red\}$

lazy atom controls *arrive* **reads** *pc*
update
$[\!] \ pc = away \rightarrow arrive!$

lazy atom controls *leave* **reads** *pc*
update
$[\!] \ pc = bridge \rightarrow leave!$

lazy atom controls *pc* **reads** *pc*, *arrive*, *leave*, *signal* **awaits** *arrive*, *leave*
init
$[\!] \ true \rightarrow pc' := away$
update
$[\!] \ pc = away \wedge arrive? \rightarrow pc' := wait$
$[\!] \ pc = wait \wedge signal = green \rightarrow pc' := bridge$
$[\!] \ pc = bridge \wedge leave? \rightarrow pc' := away$

FIGURE 2 – A train

module $Train_E$ **is**
$Train[pc, arrive, signal, leave := pc_E, arrive_E, signal_E, leave_E].$
module $Train_W$ **is**
$Train[pc, arrive, signal, leave := pc_W, arrive_W, signal_W, leave_W],$

FIGURE 3 – Modules for trains

module *Controller2* **is**
  private $near_W, near_E : \mathbb{B}$
  interface $signal_W, signal_E : \{green, red\}$
  external $arrive_W, arrive_E, leave_W, leave_E : \mathbb{E}$

  passive atom controls $near_W$
    reads $near_W, arrive_W, leave_W$
    awaits $arrive_W, leave_W$
    init
      $[\!] \ true \rightarrow near'_W := false$
    update
      $[\!] \ arrive_W? \rightarrow near'_W := true$
      $[\!] \ leave_W? \ \rightarrow near'_W := false$

  passive atom controls $near_E$
    reads $near_E, arrive_E, leave_E$
    awaits $arrive_E, leave_E$
    init
      $[\!] \ true \rightarrow near'_E := false$
    update
      $[\!] \ arrive_E? \rightarrow near'_E := true$
      $[\!] \ leave_E? \ \rightarrow near'_E := false$

  lazy atom controls $signal_W, signal_E$
    reads $near_W, near_E, signal_W, signal_E$
    init
      $[\!] \ true \rightarrow signal'_W := red; \ signal'_E := red$
    update
      $[\!] \ near_W \wedge signal_E = red \rightarrow signal'_W := green$
      $[\!] \ near_E \wedge signal_W = red \rightarrow signal'_E := green$
      $[\!] \ \neg near_W \rightarrow signal'_W := red$
      $[\!] \ \neg near_E \rightarrow signal'_E := red$

FIGURE 4 – A possible controller

`Exercise 2` `Monitoring alternation`

Consider a module P with an interface variable x that ranges over non-negative integers. Assume it is ok for x to decrease during one update round but not to decrease twice in a row (that is during two consecutive update rounds).

1. Why is a monitor required for checking this requirement ?

2. Propose a monitor which checks this requirement.

`Exercise 3` `Finite reachability`

1. Recall the definition of finitely reaching for a transition graph.

2. Prove that every finite transition graph is finitely reaching.

3. Prove that every graph which is both finitely branching and finitely reaching is such that its reachable subgraph is finite.