CIS 771, Software Specification, Spring 2017.

Assignment #1. Due on Thursday, February 16, 2017, 11:59pm (US Central)

Below you will find the CIS771 Homework 1. You are required to

1. define an Alloy model, with execution commands, so as to
2. answer the questions given below;
3. then submit your solution via K-State Online.

This homework is to be completed independently!

Purpose of Assignment

To help you learn

1. how to construct simple static Alloy specifications, and
2. how to use the Alloy Constraint Analyzer (ACA) to generate model instances and counterexamples.

Problem Description

You are to develop a simple model of a railroad signalling system. Most signaling systems used colored lights (similar to auto traffic lights) placed at regular intervals along a railroad track. The goal of the signaling system is to ensure that two trains are not in the same place at the same time (if this simple constraint is satisfied, there can be no collision between two trains).

In this assignment, we will consider a simplified version of real signalling system based on the following *Principles of Signalling* taken from a web-page maintained by Mark D. Bej. Everything you need to know about signalling to complete this assignment is given in this assignment description. However, if you are curious about real signalling systems, you can check out http://www.railway-technical.com/ (look for *Signalling & Control* in the side bar). For our next Alloy assignment, we will considering more complicated aspects of signaling.

Below is a excerpt from *Principles of Signalling*:

> Railroad signals are similar in purpose to highway traffic signals for cars. Keep in mind, though, in reading this, that it was highway signals that developed from railroad signals, not the other way around.
>
> Everyone needs some warning before getting a signal to stop. For your car, that warning is the yellow light the appears for a few seconds before the red light appears. The yellow light is timed according to the prevailing speed on the road it governs and, of course, according to the required

stopping distances for cars and trucks. It does not take much time or distance (relative to trains) for cars and trucks to stop. (Cars and trucks can also turn out of the way of danger.) Therefore, the yellow warning light can be provided at the same physical location (the same signal apparatus) as the red danger light.

Trains, however, are heavy. They may require 1/2 to 1+1/2 mile to stop. Thus, they need a warning well before the point where they have to stop. The arrangement must inherently be different from what is present on roads. Here are the basics of how it works.

Each section of track, from signal to signal, is one 'block'. So far as we are concerned here, the signal defines the block regardless of how the underlying hardware may work. Except in infrequent instances beyond the scope of this introduction, only one train is permitted in each block. Entry into the block is permitted or denied by the signal. That signal is said to 'govern' the block.

Blocks in the past, in the days of short trains, were often about a mile (about 1.5 km) long. With longer and heavier trains, they've been lengthened over the years, so that a more typical value on large U.S. mainlines at present is about 2-3 miles. However, there are many factors involved in determining block lengths.

A red signal can mean one of several things:

- The next block is occupied.
- If the signal is a controlled signal (under direct human control, not automatic), the person controlling it may simply want to hold the train at that location for some time, to allow some other train to pass;
- Some other danger situation exists. This may include a switch (points) in the next block not properly lined. Or, another route may have been set which does not allow the train to proceed (e.g., at a 90 degrees crossing).

A yellow signal indicates that the next block is clear, but that the block following that is occupied or unsafe. Thus, the following signal is red, and the train has the length of one block to stop.

A green signal, finally, means that the next two blocks (at least) are clear, and that the next signal is therefore either yellow, or also green.

This ends the excerpt from *Principles of Signalling*.

Based on this description above, we will model the signalling on a simple railway consisting of a single linear track with trains moving west to east.

```
                        >Train 1>                          >Train 2>
 West  s(1) ======= s(2) ======= s(3) === ..... === s(k) ======= s(k+1) East
              b(1)        b(2)                          b(k)

              ---- direction of train travel --->
```

The railway signalling system consists of signals s(1)..s(k+1) and blocks b(1)..b(k) where a signal s(i) governs block b(i) as described above. Each signal has a single color setting -- it is either set to red, yellow, or

green. As described above, a signal's setting is determined by occupancy state of the block that it governs and the state of signals that follow it in the direction of train travel.

Your task

is to write an Alloy specification that models such a railway signaling system, and show how to use it to generate instances with

1. zero yellow signals
2. exactly one yellow signal, and
3. exactly two yellow signals.

Your specification should of course not be "over-constrained" (in which case you cannot generate instances), but neither should it be "under-constrained" and allow instances that do not conform to the above picture, for example an instance that has

- *two* disjoint tracks, or
- two trains on the same block, or
- a cyclic track, or
- etc, etc.

When grading, we are going to check that your model disallows certain of these ill-formed systems, but you should already now write *three* assertions that each

1. checks a desirable property (e.g., that there cannot be cyclic tracks),
2. cannot be violated by your model (as you should verify within a reasonable scope), and
3. will be violated by making a (small) change to your specification.

What to submit

You should submit one file, an executable Alloy file, that contains

1. your Alloy specification
2. the 3 assertions that check desirable properties
3. in comments:
    1. a verbal description of each of the 3 instances generated (for 0, 1, 2 yellow lights).
    2. a description of how to change your model so as to violate each of the 3 assertions.

How to start

Your specification should be an *extension* of the following skeleton:

```
sig Block {}
sig Signal {}
sig Topology{
  westBlock: Block -> Block,
  eastBlock: Block -> Block,
```

```
  westSignal: Block -> Signal,
  eastSignal: Block -> Signal,
 }

sig Train{}
abstract sig Color {}
one sig Red, Yellow, Green extends Color {}
sig State {
  setting: Signal -> Color,
  location: Train -> Block
 }
```

General hints

There are many ways to solve this problem. Alloy allows set and relation multiplicities, and general constraints; the latter may be part of the signature or listed separately.

Also, there are multiple ways to generate instances that satisfy a particular property. For example, asserting the negation of the property and checking the assertion will cause ACA to generate a counter-example. (A counter-example to the negation of the property is an instance satisfying the property!)

I found it convenient to introduce a few additional subsets and relations to handle

- special cases of signals like the westmost signal and eastmost signal (it's helpful to identify these because unlike other signals, they don't have a block on both sides of them)
- other relations such as the notion of a "following signal" (relates a signal to the next signal to the east) that sometimes shorten what you need to write when specifying your invariants.