System Validation 2022
Homework Part 1 - Model Checking

Marieke Huisman, Raúl Monti and Philip Tasche

Deadline: 23:59 CET, Friday Sept. 30, 2022

The assignment should be made in pairs.
This assignment consists of

- Writing temporal logic specifications.
- Using them to analyse some given models.
- Adapting the models to fix errors discovered with the first two steps.

All solutions should be uploaded to canvas.
You should hand in a single ZIP file with:

- The report in PDF format as `report.pdf`. The report should fully describe your solutions. You will find guidelines regarding this in the exercises themselves.
- The input and output files as required in each of the exercises.

In all files, and on the first page of the report, write your names and student numbers. If you have any questions regarding the assignment, contact r.e.monti@utwente.nl (Raúl Monti).

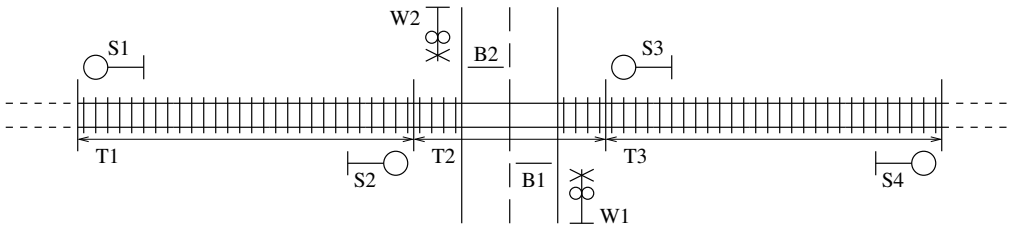

Figure 1: Schematic of the level crossing.

```
#include "crossing-environment-trivial.smv"
#include "crossing-interlocking-trivial.smv"
#include "crossing-main.smv"
#include "crossing-props.smv"
```

Figure 2: Empty behavior case for the level crossing (`crossing-trivial.smv`).

1 Modeling and Verification of a level crossing Interlocking System

Description of the level crossing interlocking system

We consider the requirements of a level crossing of a road and a single track railway. In figure 1, we have drawn a picture of the situation.

From left to right there is a signal `S1` that controls access to the unmodeled tracks to the left, then there is the track `T1` that leads to a signal `S2`. This signal controls access to both the track section `T2`, which is the crossing itself, and the track section `T3` that follows the crossing. Similarly the signal `S3` controls access to `T2` and `T1`, while `S4` controls the unmodeled tracks to the right.

For the cars there are in each direction (top-down and bottom-up), both a warning device that has blinking lights and bells to warn drivers and a barrier that prevents cars from driving onto the level crossing. The warning devices are labeled `W1` and `W2`. The barriers are labeled `B1` and `B2`. Warning devices should start blinking before the barriers come down and should remain blinking until the barriers are up again.

Of course, we would like this system to prevent trains and cars running into each other. For this, we would like that the signals, warnings and barriers to work as expected. For instance, we would want `S2` and `S3` to signal *stop* if the road barriers are open, and that barriers eventually come down for trains to traverse `T2`.

NuSMV models of the level crossing interlocking system

To analyse the interlocking system, we will divide our NuSMV models into three parts:

- **Environment**. This part describes the behavior of the environment in which the interlocking is to be validated.
- **Interlocking**. Describes the behavior/implementation of the interlocking
- **Specification**. Contains all of the properties that the system must satisfy.

We will also make the following assumptions in our models:

- We assume that signals have both a red and a green light, where red means stop and green means go.
- We assume that every train occupies precisely one track. (This is, we use a point model of trains.)
- Finally, to simplify the modeling and the properties, we assume that the interlocking reacts to the environment without delay. This means that:
  - The next state of the environment is computed based on the current state of the system as a whole.
  - The next state of the interlocking is computed based on the current state of the system as a whole and the next state of the environment.

Along with this assignment, we give you a set of files corresponding to each of the parts (environment, interlocking and properties). Also, we ask you to implement your own models and specify your own properties in the exercises. In order to be able to interchange and combine the various environments, interlockings and specifications, they must be in separate files and use a common interface. You can find the interface file named `crossing-main.smv` within the given files for this assignment. The atomic properties defined by this interface are:

T*id*`_occupied`	Track *id* is occupied (by a train).
`cars_crossing`	One or more cars are crossing at track `T2`.
S*id*`_red`	Signal *id* is red.
S*id*`_green`	Signal *id* is green.
W*id*`_active`	Warning device *id* is active.
B*id*`_open`	Barrier *id* is open.
B*id*`_closed`	Barrier *id* is closed.
B*id*`_request`	The interlocking requests that the environment closes Barrier *id*.

The interface is the glue code that allows you to compose an instance by including an environment, an interlocking, the interface and the specification. You can then combine your different files into a complete NuSMV model by writing a new file that uses pre-compiler instructions as in Fig. 2.

Provided files

To carry out the exercises, we provide you with the following files:

`crossing-main.smv`	The glue code between environment, interlocking and properties. **Do not make any changes to this file yourself.**
`crossing-props-example.smv`	An example of how properties can be modularized.
`crossing-trivial.smv`	Complete trivial instance of the system.
`crossing-environment-trivial.smv`	Trivial model of the environment.
`crossing-interlocking-trivial.smv`	Trivial model of the interlocking.
`crossing-environment-slow.smv`	Model of an environment with a slow train.
`crossing-environment-fast.smv`	Model of an environment with a fast train.
`crossing-interlocking-flawed.smv`	Flawed model of the interlocking.
`crossing-flawed-slow.smv`	Complete instance with flawed interlocking and slow train.
`crossing-flawed-fast.smv`	Complete instance with flawed interlocking and fast train.
`crossing-unknown-model.smv`	Complete unknown instance combining an interlocking system and an environment.
`crossing-unknown.smv`	Combination of unknown instance and your properties.

You may want to inspect these files now, and maybe play around analysing them with NuSMV. There are hints and instructions at the end of this sheet that may help you to do so.

Exercises

i. Write a set of properties that validates the correct behaviour of the interlocking system. With *correct behaviour* we mean that: **(50 pt)**

- Signals work as expected (they go green to let trains pass and red to stop them from running into cars).
- Warning devices and barriers work as expected (the barrier is down when a train is in T2, the warnings are also blinking, etc ...).
- Trains always make progress.
- Cars always make progress.
- The environment evolves as expected: E.g.
  - Regardless of the history, the environment can evolve into a situation where no trains are visible.
  - Regardless of the history, the environment can evolve into a situation where a train is crossing the road.
  - Similarly with cars.

Put these properties in a file `crossing-props.smv`, and place the file in the ZIP archive you will submit. Also include the properties in the report along with a description of the intention of each formula in plain English. Also in the report, classify the properties as safety, liveness, or other.

ii. The given interlocking implementation (`crossing-interlocking-flawed.smv` is not error free. **(30 pt)**
To expose the error:

(a) Copy `crossing-environment-fast.smv` to `crossing-environment-fixed.smv`.

(b) Copy `crossing-interlocking-flawed.smv` to `crossing-interlocking-fixed.smv`.

(c) Refactor the environment model `crossing-environment-fixed.smv` by defining separate modules for the train, the car, and two instances of a barrier module.

(d) Rewrite the barrier to have a non-deterministic closing time.

(e) Running NuSMV on `crossing-fixed.smv` should now produce a safety violation. If not, then you should revisit your properties from exercise (**i**).

Now that you have exposed the error in `crossing-interlocking-fixed.smv`, fix it (NuSMV should not find any safety or liveness issues).
Put the following files (final fixed versions) into the ZIP archive:

- `crossing-environment-fixed.smv`
- `crossing-interlocking-fixed.smv`

Also, in your report, explain the error you found, mention the property that was falsified, and describe how you fixed it.

iii. The file `crossing-unknown.smv` applies the set of formulas that you write to an unknown model of the interlocking system, contained in the file `crossing-unknown-model.smv`. Running NuSMV on this instance should result in a few counter-examples to liveness properties. If not then you should revisit your liveness properties. **(20 pt)**
In the report:

- Include a listing of one of the counter-examples to a liveness property.
- Following the counter-example, explain what happened in terms of trains driving along tracks and cars crossing the tracks. Explain how this contradicts the property.

Instructions and practical matters

Verifying with NuSMV To verify an instance of the interlocking system such as `crossing-trivial.smv` with NuSMV, you need to use the `-cpp` flag. This flag instructs NuSMV to first run the C Pre Processor on its input. [1] For instance, try running:

```
NuSMV -cpp crossing-trivial.smv
```

Grading The grading process will have a machine testing part and manual inspection part. For the machine testing it is essential that:

- Your properties do not use atomic propositions other than the ones specified, unless they are defined in your properties file.

[1] Comments can lead to warnings from cpp because they are not legal C code. To force a comment to be both a NuSMV and a C comment start them with `--//`.

- You do not make changes to the interfaces of the Environment and the Interlocking.
  Note that changing a defined boolean to a boolean variable and vice versa does not change the interface.
- You do not change the names of files. More precisely, you should make sure to include only the following files in your ZIP archive:

`report.pdf`	
`crossing-props.smv`	From exercise (i)
`crossing-environment-fixed.smv`	From exercise (ii)
`crossing-interlocking-fixed.smv`	From exercise (ii)