Development and Verification of a Traffic Light Controller

Formal Methods in Software Engineering 2021-2022
Homework

1. Introduction

You are asked to develop a traffic light controller managing traffic at the crossroads. Each road lane is controlled by its own local sensor. Sensors on the road lanes detect cars. When there are no cars on a lane, the traffic light is **red**. The similar sensor controls a pedestrian lane. When cars or pedestrians appear on the corresponding lane, the process of this lane traffic light starts to compete for the resources. The resources are points in the crossroads where different lanes cross. If a process receives requested resources, it switches its traffic light to **green** allowing cars or pedestrians to move along its lane. The core of the homework is to develop correct algorithms for requesting and releasing resources in traffic light processes.

This homework includes the development of processes for traffic light controllers that solve the traffic management problem at the crossroads. Each controller consists of several parallel processes-subcontrollers (for each direction) and global shared variables, which can be modified and read by these processes. Each traffic light is controlled by its own process. A process allows for or restricts the movement along the corresponding direction. One of the local variables holds the value that is permissive (**green**) or prohibitive (**red**) color linked with a traffic light process. Global variables are used to implement the synchronization among processes. Developed algorithms must guarantee the correct usage of shared resources.

You may use the implementation of the trivial crossroads management algorithm given in Section 2 as the starting point for your solution. In Section 3, you can find the scheme of the more sophisticated algorithm with several crossing directions. Each student receives a configuration according to the distribution with several crossing lanes denoted by capital letters of the from/to directions (e.g., **NW** — from North to West). Other lanes of the crossroads on the general scheme are ignored.

Successful completion of the homework assignment involves:
1. Constructing a corresponding Promela model;
2. Verifying the model in Spin with respect to safety, liveness and fairness properties (the scheme described in Section 4, in your case the corresponding properties can be specified differently).

You are also asked to prepare a short report describing the crossroads configuration, developed algorithms, solution's architecture and, what is more important, the verification results.

2. The Simple Crossroads

This section presents simple algorithms for managing the trivial crossroads.

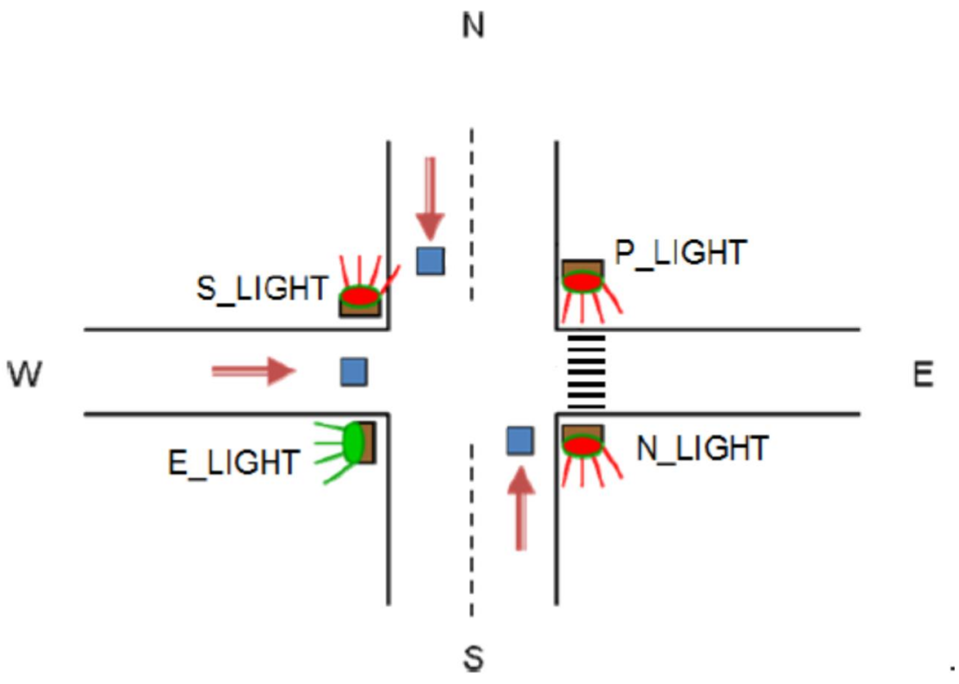

Figure 1. Trivial crossroads

Let us construct a controller for the traffic lights of the crossroad in Fig. 1. The controlled traffic lights are **N** (to North), **S** (to South), **E** (to East), **P** (pedestrian crossing). The external events N_SENSE, S_SENSE, E_SENSE, P_SENSE indicates that the corresponding sensor detected a queue. Movement via a direction is allowed when the corresponding traffic light is **green**. In this simple case:

1. Cars do not turn at the crossroads; they can pass it only directly.
2. Traffic lights have only two lights: **red**-**green**-**red**-...
3. Pedestrians are allowed to cross the road only in the specific place.

A traffic light includes four parallel processes **N**, **S**, **E**, and **P**. Each process supervises its own traffic light, which allows for the movement along its direction. The output of the controller is the assignment of the traffic light variables, i.e., the local variables of processes N_LIGHT, S_LIGHT, E_LIGHT, P_LIGHT are assigned **RED** or **GREEN**. For instance, process N resets its own local boolean variable N_REQ, when the traffic in the north direction is detected. The value of N_REQ may be set by the external event N_SENSE as follows:

```
while true do if (!N_REQ & N_SENSE) then N_REQ := true;
```

This loop is executed in a separate process representing the external environment — traffic generation. It is executed in parallel with process **N**. The same is supposed to be used for the other crossroads directions.

To synchronize processes, we use variables shared among all processes. We introduce two global boolean variables (direction flags) — NS_LOCK and WE_LOCK. Processes **N**, **S**, and **E** will communicate through these shared boolean variables. All processes below are written using a Pascal-like pseudocode. For instance, the algorithm of process **N** for the north crossroads direction can be represented as follows:

```
process N(){
        /* endless loop */
        while true do
                if (N_REQ)                          /* movement is requested */
                      wait (!WE_LOCK)               /* blocks until WE direction is free */
                      NS_LOCK = true                /* locks the direction */
                      N_LIGHT = green               /* sets the traffic light */
                      wait (!N_SENSE)               /* waits until all cars pass */
                      if (S_LIGHT = red)
                            NS_LOCK = false
                      fi
                      N_LIGHT = red                 /* releases resources */
                      N_REQ = false
                fi
        od
}
```

Process **S** can be constructed similarly. Also consider an example for the algorithm of the process **E**:

```
process E(){
        while true do
                if (E_REQ)
                      WE_LOCK = true
                      E_LIGHT = green
                      wait (!NS_LOCK)
                      wait (!E_SENSE)
                      WE_LOCK = false
                      E_LIGHT = red
                      E_REQ = false
                fi
        od
}
```

Pay attention that the correctness of the above example algorithms has not been verified, especially concerning process interactions. If there are mistakes detected during verification, they have to be fixed.

3. Crossroads Configurations: a General Scheme

In this homework assignment, you will deal with more sophisticated crossroads configurations with four two-way directions (see Fig. 2). For each direction, there are three lanes for three possible paths:

1. To pass the crossroads directly;
2. To pass the crossroads turning right;
3. To pass the crossroads turning left.

Each lane is controlled by its own traffic lights — the one for moving forward, the one for turning left, and the one for turning right. In addition, there are pedestrians crossing.

Moreover, turning right is always possible (blue lanes in Fig. 2). In this case, a traffic light must be set to **red** if there are no cars on the lane. When cars appear, it is switched to **green** until there are no cars. When there are no cars, the traffic light is switched back to **red**.

For every path, a traffic light turns **green** provided that:

1. there is a request from cars on the lane to pass the crossroads;
2. the movement is prohibited on all other lanes crossing this path, i.e., the corresponding traffic lights are **red**.

Within this homework, each student is asked to work with a sub-configuration of the complete crossroads from Fig. 2. In addition, pay special attention to lanes crossing the pedestrian lane in the sub-configuration. The distribution of sub-configurations among students is provided in the separate file.

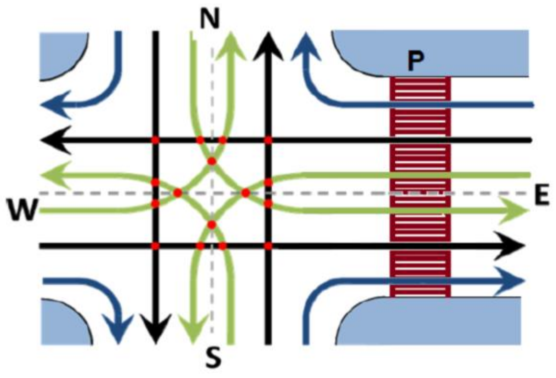

Figure 2. Crossroads configurations: a complete picture

In addition to requirements discussed in Introduction, pay attention to the following:

1. Develop separate processes for the "environment" (for every lane) generating traffic — cars and pedestrians.
2. Develop individual processes for lane controllers managing traffic.
3. Do not use *atomic* or *d_step* to synchronize processes.

A short written report should include the assigned configuration and necessary comments regarding the architecture of the traffic light controller model.

4. Verifying a Traffic Light Controller

The assignment also includes the verification of the developed Promela model for the traffic light controller managing the crossroads. Below there are some examples of LTL formulae concerning the verification of a model for the trivial crossroads discussed earlier.

SAFETY: It is not possible to simultaneously move through crossing lanes.

□ !((N_LIGHT == **GREEN** || S_LIGHT == **GREEN**) && (E_LIGHT == **GREEN**))

LIVENESS: When a car or pedestrian appears on a lane, they will eventually get an opportunity to cross the crossroads.

□ (N_SENSE && (N_LIGHT == **RED**) ⇒ ◊ (N_LIGHT == **GREEN**))

□ (S_SENSE && (S_LIGHT == **RED**) ⇒ ◊ (S_LIGHT == **GREEN**))

□ (E_SENSE && (E_LIGHT == **RED**) ⇒ ◊ (E_LIGHT == **GREEN**))

□ (P_SENSE && (P_LIGHT == **RED**) ⇒ ◊ (P_LIGHT == **GREEN**))

Liveness properties are to be verified under the following FAIRNESS assumptions. Intuitively, the traffic on each lane cannot be continuous, i.e., a corresponding traffic sensor is supposed to be set to false infinitely often:

□ ◊ !((N_LIGHT == **GREEN**) && N_SENSE)

□ ◊ !((S_LIGHT == **GREEN**) && S_SENSE)

□ ◊ !((E_LIGHT == **GREEN**) && E_SENSE)

□ ◊ !((P_LIGHT == **GREEN**) && P_SENSE)

Representation of these properties and LTL formulae fully depends on the specific Promela model for a traffic light controller.

Do not forget to include Spin verification reports in the report. What is more important, you need to check that the verification is exhaustive. Thus, pay special attention to the inner organization of the Promela model, so that its state space is finite and can be stored in memory.

Good luck!