# Development and Verification of the Traffic Light Controller

Formal Methods in Software Engineering 2019/2020
*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 the lane traffic light starts to compete for the resources. The resources are points of the crossroads where different lanes cross. If a process receives requested resources, it switches its traffic light to green allowing 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 development of processes for traffic light controllers that solve the traffic management problem at the crossroads. Each controller consists of several parallel processes-controllers (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 group receives their own 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 scheme can be ignored.

To successfully complete the homework assignment, it is necessary to:

1. construct a corresponding `Promela` model;
2. verify the model in `Spin` w.r.t. safety, liveness and fairness properties (described in Section 4).

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

## 2 The Simple Crossroads

This section presents simple algorithms for managing the trivial crossroads.

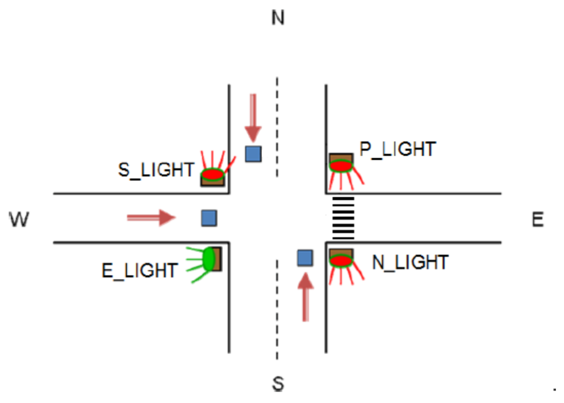

Figure 1: Trivial Crossroads

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` displays that the corresponding sensor detected queue. Movement on the corresponding direction is allowed when the corresponding traffic light is green. In this simple case:

1. cars do not turn at the crossroads, they can only cross it directly;
2. traffic lights have only two lights: red-green-red-green. . . ;
3. pedestrians cross the road only in a specific place.

A controller consists of three parallel processes: N, S, E, and P. Each process controls 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, a 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 event N_SENSE as follows:

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

This loop is executed in the separate process modeling the external environment. It is executed in parallel with N. The same is supposed to be used for the other crossroads directions.

To synchronize processes, we need variables shared among all processes. We introduce 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 the Pascal-like language. For instance, the algorithm of the process N for the north crossroads direction can be coded as follows:

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

The process S can be constructed similarly. Consider also the 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 these example algorithms has not been verified. If there are mistakes detected during the verification, they have to be fixed.

## 3 The Crossroads Configurations

In the homework assignment, you are asked to 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:

- to pass the crossroads without turning;
- to pass the crossroads turning right;
- to pass the crossroads turning left.

Each lane is controlled by its own traffic light — 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 each 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 the framework of this homework, each group is asked to work with a sub-configurations 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 groups is provided in the separate table.

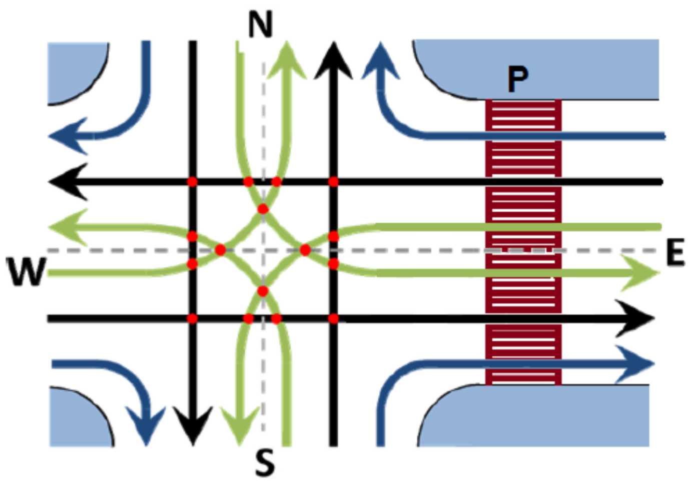

Figure 2: Crossroads configurations: complete figure

In addition to the requirements listed in Section 1, pay attention to the following:

1. Develop separate processes for the "environment" (for each lane) which generate car and pedestrian traffic.
2. Develop individual processes for lane controllers managing the corresponding traffic at the crossroads.
3. Do not use `atomic` to synchronize processes.

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

## 4 Verifying Traffic Light Controller

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

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

```
G !(( N_LIGHT=GREEN || S_LIGHT=GREEN) and (E_LIGHT = GREEN))
```

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

```
G ( N_SENSE and ( N_LIGHT=RED ) ⇒ F ( N_LIGHT=GREEN ) );

G ( S_SENSE and ( S_LIGHT=RED ) ⇒ F ( S_LIGHT=GREEN ) );

G ( E_SENSE and ( E_LIGHT=RED ) ⇒ F ( E_LIGHT=GREEN ) );

G ( P_SENSE and ( P_LIGHT=RED ) ⇒ F ( P_LIGHT=GREEN ) );
```

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

```
GF !( (N_LIGHT=GREEN) & N_SENSE );

GF !( (S_LIGHT=GREEN) & S_SENSE );

GF !( (E_LIGHT=GREEN) & E_SENSE );

GF !( (P_LIGHT=GREEN) & P_SENSE );
```

Formulation of these properties and corresponding LTL properties fully depends on the organization of the specific `Promela` model for the traffic light controller managing the croccroads. Do not forget to include `Spin` verification results in the report. What is more important, you are asked 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!**