**Traffic Light Controller Implementation**

Distributed control systems

**Project professor:** Dahlia Al-Janabi

**Team**: Team 6

**Team members:**

Gruia Sergiu-Dimitrie

Brezae Tudor

Dutkas Alexis-Andrei

Diaconu Paul-Costin

Contents

[Maps 3](#_Toc156438755)

[Petri Net Design 5](#_Toc156438756)

[Types 5](#_Toc156438757)

[Guards 6](#_Toc156438758)

[Mappings 6](#_Toc156438759)

[Controller Design 7](#_Toc156438760)

[Types 8](#_Toc156438761)

[Guards and Mappings 8](#_Toc156438762)

[Component Diagram 10](#_Toc156438763)

[Implementation 10](#_Toc156438764)

# Maps

For this project we were assigned 2 intersections from Cluj-Napoca, Romania, as seen in Figure 1. First Intersection and Figure 2. Second Intersection: A screenshot of a map

Description automatically generated

Figure . First Intersection

Figure . Second Intersection

An aerial view of a crosswalk

Description automatically generated

For these we took the directions as seen in Figure 3. Road Directions Overlay

An aerial view of a city

Description automatically generated

Figure . Road Directions Overlay

Present as well in Figure 4. Simplified Road Directions in a simplified version.

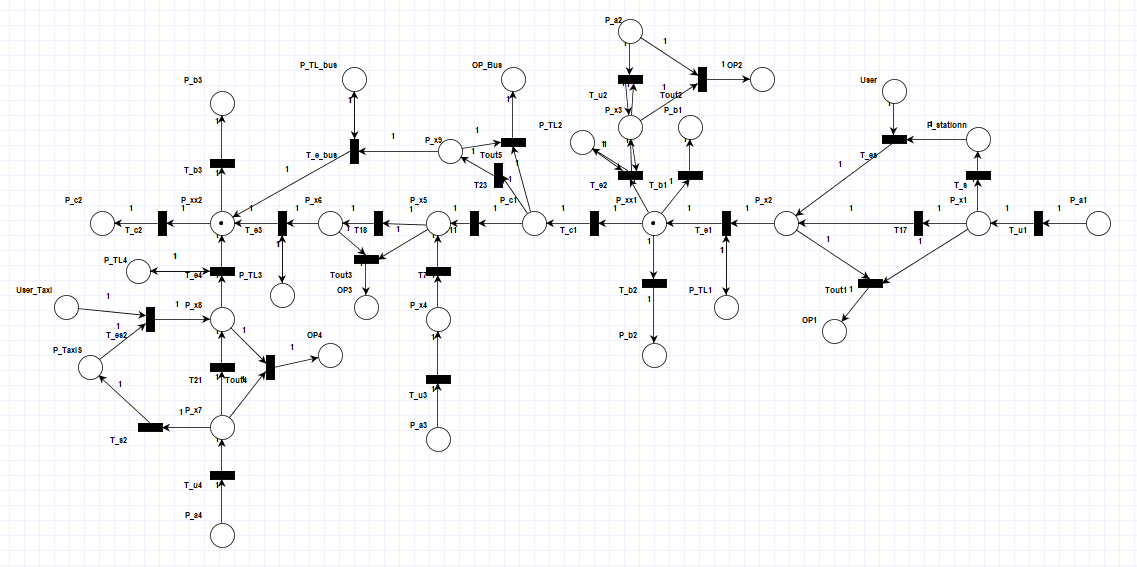
A drawing of a road with arrows

Description automatically generated

Figure . Simplified Road Directions

# Petri Net Design

We then modeled the intersection using Petri Nets:



For which we have the following:

## **Types**

Type(P\_a1) = Type(P\_a2) = Type(P\_a3) = Type(P\_a4) = **DataCar()** // *inputs of the lanes*

Type(P\_b1) = Type(P\_b2) = Type(P\_b3) = **DataCar()** // *outputs of the lanes*

Type(P\_c1) = Type(P\_c2) = **DataCar()** // *outputs of the intersection*

Type(P\_x1) = Type(P\_x2) = Type(P\_x3) = Type(P\_x4) = Type(P\_x5) = Type(P\_x6) = Type(P\_x7) = Type(P\_x8) = **DataCarQueue()** // *intersection inputs represented as queues*

Type(P\_x9) = **DataCarQueue()** // *intersection input represented as queue for bus lane*

Type(P\_station) = Type(P\_taxi\_station) = **DataCarQueue** // *intersection inputs as queues for taxi and bus*

Type(OP1) = Type(OP1) = Type(OP1) = Type(OP1) = DataTransfer()

Type(P\_TL1) = Type(P\_TL2) = Type(P\_TL3) = Type(P\_TL4) = Type(P\_TL\_BUS) = **DataString()** // *traffic lights*

Type(P\_xx1) = Type(P\_xx2) = **DataCarQueue()** // *intersections*

Type(User) = Type(User\_taxi) = **DataString()** // *human input*

## **Guards and Mappings**

**Right Intersection**

T\_u1:

Grd = (P\_a1!=NULL && P\_x1.CanAddCars && P\_a1.IsBus)

Map = P\_a1.addElement(P\_x1)

T\_s:

Grd = (P\_x1.HaveCarForMe && ( P\_x1.isPriorityCar || P\_x1.haveBus))

Map = P\_x1.addElement(P\_station)

T\_es:

Grd = (user !=NULL && P\_station.haveBus)

Input = ArrayList<string>.add -> P\_station

-> user

Map = (input, P\_station.PopBusToQueue(P\_x2) )

T17:

Grd = (P\_x1 != NULL)

Map = P\_x1.addElement(P\_x2)

T\_e1:

Grd = (P\_x2.HaveCarForMe && P\_TL1==”Green”)

Map = P\_x2.PopElementWithoutTarget (P\_xx1)

T\_b1:

Grd = (P\_xx1.HaveCarForMe)

Map = P\_xx1.PopElementWithTarget(P\_b1)

T\_b2:

Grd = (P\_xx1.HaveCarForMe)

Map = P\_xx1. PopElementWithTarget (P\_b2)

T\_e2:

Grd = (P\_x3.HaveCarForMe && P\_TL2==”Green”)

Map = P\_x3.PopElementWithoutTarget (P\_xx1)

T\_u2:

Grd = (P\_a2.HaveCarForMe)

Map = P\_a2.addElement(P\_x3)

T\_c1:

Grd = (P\_xx1.HaveCarForMe)

Map = P\_xx1. PopElementWithTarget (P\_c1)

**Left Intersection**

T23:

Grd = (P\_c1.HaveCarForMe && ( P\_c1.isPriorityCar || P\_c1.haveBus))

Map = P\_c1.addElement(P\_x9)

T\_e\_bus:

Grd = (P\_x9.haveBus && P\_TL\_BUS==”Green”)

Map = P\_x9.PopElementWithoutTarget (P\_xx2)

T\_u3:

Grd = (P\_a3.HaveCarForMe)

Map = P\_a3.addElement(P\_x4)

T7:

Grd = (P\_x4 != NULL)

Map = P\_x4.addElement(P\_x5)

T11:

Grd = (P\_c1 != NULL)

Map = P\_c1.addElement(P\_x5)

T18:

Grd = (P\_x5 != NULL)

Map = P\_x5.addElement(P\_x6)

T\_e3:

Grd = (P\_x6.HaveCarForMe && P\_TL3==”Green”)

Map = P\_x6.PopElementWithoutTarget (P\_xx2)

T\_u4:

Grd = (P\_a4.HaveCarForMe)

Map = P\_a4.addElement(P\_x7)

T21:

Grd = (P\_x7 != NULL)

Map = P\_x7.addElement(P\_x8)

T\_e4:

Grd = (P\_x8.HaveCarForMe && P\_TL4==”Green”)

Map = P\_x8.PopElementWithoutTarget (P\_xx2)

T\_c2:

Grd = (P\_xx2.HaveCarForMe)

Map = P\_xx2.PopElementWithTarget(P\_c2)

T\_s2:

Grd = (P\_x7.HaveCarForMe && ( P\_x7.isPriorityCar || P\_x7.haveBus))

Map = P\_x7.addElement(P\_taxi\_station)

T\_es2:

Grd = (user\_taxi !=NULL && P\_taxi\_station.HaveCarForMe)

Input = ArrayList<string>.add -> P\_taxi\_station

-> user\_taxi

Map = (input, P\_taxi\_station.PopTaxiToQueue(P\_x8))

Tout1:

Grd = (P\_x1 != NULL && P\_x2.CanNotAddCars)

Map = OP.SendOverNetwork(“Full”)

Tout2:

Grd = (P\_a2 != NULL && P\_x3.CanNotAddCars)

Map = OP.SendOverNetwork(“Full”)

Tout3:

Grd = (P\_x5 != NULL && P\_x6.CanNotAddCars)

Map = OP.SendOverNetwork(“Full”)

Tout4:

Grd = (P\_x7 != NULL && P\_x8.CanNotAddCars)

Map = OP.SendOverNetwork(“Full”)

Tout5:

Grd = (P\_c1 != NULL && P\_x9.CanNotAddCars)

Map = OP.SendOverNetwork(“Full”)

# Controller Design

A diagram of a network

Description automatically generated

For which we have the following:

## **Types**

Type(r1r2r3r4r5) = Type(g1r2r3r4r5) = Type(y1r2r3r4r5) = Type(r1g2r3r4r5) = Type(r1y2r3r4r5) = Type(r1r2g3r4r5) = Type(r1r2y3r4r5) = Type(r1r2y3r4r5) = Type(r1r2r3g4g5)= **DataString()**

Type(OP1) = Type(OP2) = Type(OP3) = Type(OP4) = Type(OP\_Bus) = **DataTransfer()**

## **Guards and Mappings**

T1:

Grd = (r1r2r3r4r5 != NULL)

Map1 = r1r2r3r4r5.Move(g1r2r3r4r5)

Map2 = green.SendOverNetwork(OP1)

T2:

Grd = (g1r2r3r4r5 != NULL)

Map1 = g1r2r3r4r5.Move(y1r2r3r4r5)

Map2 = yellow.SendOverNetwork(OP1)

T3:

Grd = (y1r2r3r4r5 != NULL)

Map1 = y1r2r3r4r5.Move(r1g2r3r4r5)

Map2 = red.SendOverNetwork(OP1)

Map3 = green.SendOverNetwork(OP2)

T4:

Grd = (r1g2r3r4r5 != NULL)

Map1 = r1g2r3r4r5.Move(r1y2r3r4r5)

Map2 = yellow.SendOverNetwork(OP2)

T5:

Grd = (r1y2r3r4r5!= NULL)

Map1 = r1y2r3r4r5.Move(r1r2g3r4r5)

Map2 = red.SendOverNetwork(OP2)

Map3 = green.SendOverNetwork(OP3)

T6:

Grd = (r1r2g3r4r5 != NULL)

Map1 = r1r2g3r4r5.Move(r1r2y3r4r5)

Map2 = yellow.SendOverNetwork(OP3)

T7:

Grd = (r1r2y3r4r5 != NULL)

Map1 = r1r2y3r4r5.Move(r1r2r3g4g5)

Map2 = red.SendOverNetwork(OP3)

Map3 = green.SendOverNetwork(OP4)

Map4 = green.SendOverNetwork(OP\_Bus)

T8:

Grd = (r1r2r3g4g5 != NULL)

Map1 = r1r2r3g4g5.Move(r1r2r3y4y5)

Map2 = yellow.SendOverNetwork(OP4)

Map3 = yellow.SendOverNetwork(OP\_Bus)

T9:

Grd = (r1r2r3y4y5 != NULL)

Map1 = r1r2r3y4y5.Move(r1r2r3r4r5)

Map2 = red.SendOverNetwork(OP4)

Map3 = red.SendOverNetwork(OP\_Bus)

Petri net for Intersection:

A screenshot of a computer

Description automatically generated

Petri net for Controller:

A screenshot of a computer

Description automatically generated