Student No 100964641

Carleton University

5/19/2015

Carleton University

Department of Systems and Computer Engineering

SYSC 5104 Assignment 1

Cristina Ruiz Martin

**Table of contents**

[1. Conceptual Model 4](#_Toc422411107)

[1.1. Component behavior description 4](#_Toc422411108)

[Crossroad 4](#_Toc422411109)

[Car Line 5](#_Toc422411110)

[Pedestrian Line 5](#_Toc422411111)

[Crossroad Controller 5](#_Toc422411112)

[Crossroad Traffic Light 6](#_Toc422411113)

[2. System’s Model 7](#_Toc422411114)

[2.1. Model definition 7](#_Toc422411115)

[2.2. Coupled models: PDEVS formal specification 7](#_Toc422411116)

[Crossroad 7](#_Toc422411117)

[Pedestrian Line coupled model 8](#_Toc422411118)

[Car Line coupled model 8](#_Toc422411119)

[Crossroad Controller coupled model 9](#_Toc422411120)

[Crossroad Traffic Light coupled model 9](#_Toc422411121)

[Traffic Light coupled model 9](#_Toc422411122)

[Filter & Atomic model 10](#_Toc422411123)

[2.3. Atomic models: PDEVS formal specification 10](#_Toc422411124)

[Filter 10](#_Toc422411125)

[Pedestrian Button atomic model 11](#_Toc422411126)

[Car Line atomic model 12](#_Toc422411127)

[Traffic Controller atomic model 14](#_Toc422411128)

[Crossed Cars atomic model 16](#_Toc422411129)

[Traffic Light Coordinator atomic model 17](#_Toc422411130)

[Car Traffic Light atomic model 19](#_Toc422411131)

[Pedestrian Traffic Light atomic model 20](#_Toc422411132)

[2.4. Testing strategy 21](#_Toc422411133)

[3. Simulation model 22](#_Toc422411134)

[3.1. Atomic model testing 22](#_Toc422411135)

[Filter atomic 22](#_Toc422411136)

[Pedestrian Button atomic model 23](#_Toc422411137)

[Car Line atomic model 25](#_Toc422411138)

[Crossed Cars atomic model 27](#_Toc422411139)

[Traffic Controller atomic model 30](#_Toc422411140)

[Traffic Light Coordinator atomic model 33](#_Toc422411141)

[Car Traffic Light atomic model 35](#_Toc422411142)

[Pedestrian Traffic Light atomic model 37](#_Toc422411143)

[3.2. Coupled model testing 39](#_Toc422411144)

[Filter & Atomic model 39](#_Toc422411145)

[Car Line coupled model 41](#_Toc422411146)

[Traffic Light coupled model 43](#_Toc422411147)

[Crossroad Traffic Light coupled model 46](#_Toc422411148)

[Crossroad Controller coupled model 49](#_Toc422411149)

[3.3. Integration testing 56](#_Toc422411150)

[Pedestrian buttons and Crossroad Controller coupled model 56](#_Toc422411151)

[Pedestrian buttons, Crossroad Controller and car lines coupled model 64](#_Toc422411152)

[Crossroad top model 69](#_Toc422411153)

[3.4. Other simulations 78](#_Toc422411154)

1. Conceptual Model

The objective of this work is to model and simulate the traffic lights performance, both for pedestrians and cars in a crossroad using PDEVS

I will model the crossroad shown in Figure 1. For cars, the allowed directions are straightforward (represented by the arrows). Pedestrians can cross only perpendicularly to car directions. Yellow and purple pins represent the traffic lights for pedestrians; red and green pins represent the ones for cars. Pins of the same colour in the figure represent different physical devices of the same traffic light.

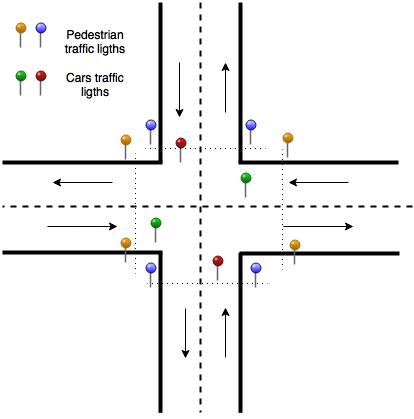


Figure 1. Crossroad schema

In this crossroad, there are 4 traffic lights to be controlled. A controller will be in charge of determining which traffic lights must to change to green, and a coordinator in charge of changing the colour of the four traffic lights. For pedestrian traffic lights, the controller will take the decision based on if there are people who have press the button to cross the street.

# Component behavior description

## Crossroad

The Crossroad top model is defined in Figure 2. The top model determines the state of the traffic lights, the number of cars in the lines, the time the first car in each line has been waiting and the state of the pedestrian buttons.

## Car Line

The Car Line receives the cars arriving and leaving the crossroad and sends to the output of the model how many cars are waiting to cross in each line and the time the first car in each line has been waiting in the intersection. Each Car Line is composed by its left and right lines.

## Pedestrian Line

The Pedestrian line receives the pedestrians arriving at the crossroad and sends both the Crossroad Controller and the output of the model if there is any pedestrian waiting to cross. If there is any pedestrian, the Pedestrian Button will be active, otherwise it will be passive.

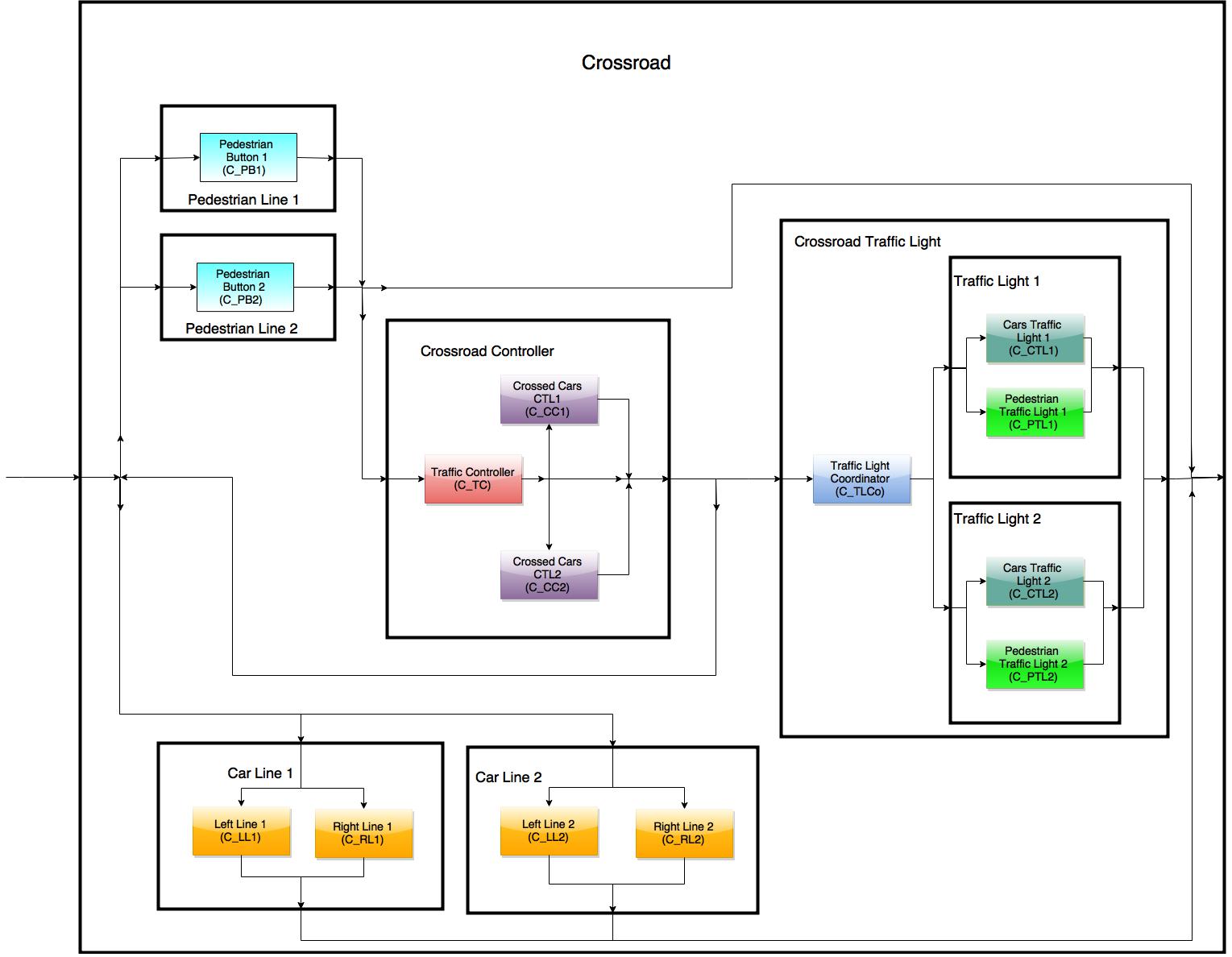


Figure 2 . Crossroad top model

## Crossroad Controller

The Crossroad Controller coupled model receives the state of the pedestrian button and sends which traffic lights must switch its color to green and the cars that can be crossing the intersection. It is composed by a Traffic Controller and two Crossed Cars models.

The **Traffic Controller** takes the decision about what traffics lights have to be in green and the **Crossed Cars** estimates the cars which can be crossing the intersection.

## Crossroad Traffic Light

The Crossroad Traffic Light models the physical traffic lights and how they are coordinated. It is composed by two Traffic Lights and a Coordinator

The **Traffic Light Coordinator** tells each Traffic Light its state (green or red) based on the information received by the Crossroad Controller.

Each **Traffic Light** is composed by two lights, one for the cars and one for the pedestrian. They are coupled in the way that they both can be in the same state. If we look at Figure 1, Traffic Light 1 is represented by red and yellow pins whereas Traffic Light 2 is represented by green and purple pins.

Both **Car Traffic Light** and **Pedestrian Traffic Light** model the device that tells the cars and pedestrians if they are allowed to pass the crossroad or not.

1. System’s Model

I will model the crossroad show in Figure 1 and sketched in Figure 2 using PDEVS.

To implement the model, I will use the CDBoost simulator. This simulator does not allow ports neither for atomic nor for coupled models (i.e. it only has an input and an output). Moreover, all the messages have to be the same type. These issues have been taken into account while defining the model.

# Model definition

The model definition is presented in Figure 2. The Crossroad top model is composed by six coupled models described in the following section.

To fulfill the requirements of CDBoost all the messages are multicast.

Each model represented in a different color in Figure 2 represents a coupled model composed by two atomic models: a filter and the atomic model itself as show in Figure 3.

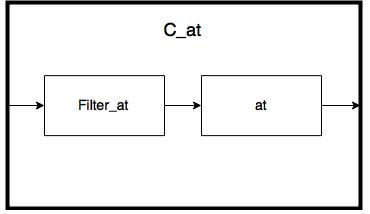


Figure 3 . Coupled model with filter

As in CDBoost all the messages must be the same type, I will define a new type of message T which has the following structure:

T {

Vector <string> to;

String from;

Int amount;

String information;

BRITime time; // A particular class to handle time

}

The variables from the message structure T not defined in a specific input or output in the model are not relevant in that message.

# Coupled models: PDEVS formal specification

## Crossroad

Crossroad = <X, Y, D, Md, EIC, EOC, IC>

Where:

X = {(T | (T.to[i] ∈ {PB1, PB2, LL1, RL1, LL2, RL2}) && (T.from = model\_input) && (T.amount ∈ N))}

Y = {(T | (T.to[i] ∈ {TC, model\_output}) && (T.from ∈ {PB1, PB2}) && (T.information ∈ {active, passive})) ∪ (T | (T.to[i] = model\_output) && (T.from ∈ {CTL1, PTL1, CTL2, PTL2}) && (T.information ∈ {red, green, yellow})) ∪ (T | (T.to[i] = model\_output) && (T.from ∈ {LL1, RL1, LL2, RL2}) && (T.amount ∈ N)&& (T.time ∈ BRITime))}

D = {Pedestrian\_line\_1, Pedestrian\_line\_2, Car\_line\_1, Car\_line\_2, Crossroad\_controller, Crossroad\_traffic\_light}

Md= {MPedestrian\_line\_1, MPedestrian\_line\_2, MCar\_line\_1, MCar\_line\_2, MCrossroad\_controller, MCrossroad\_Traffic\_Light}

EIC = {Pedestrian\_line\_1, Pedestrian\_line\_2, Car\_line\_1, Car\_line\_2}

EOC = {Pedestrian\_line\_1, Pedestrian\_line\_2, Car\_line\_1, Car\_line\_2, Crossroad\_traffic\_light}

IC = {(Pedestrian\_line\_1, Crossroad\_controller), (Pedestrian\_line\_2, Crossroad\_controller), (Crossroad\_controller, Pedestrian\_line\_1), (Crossroad\_controller, Pedestrian\_line\_2), (Crossroad\_controller, Car\_line\_1), (Crossroad\_controller, Car\_line\_2), (Crossroad\_controller, Crossroad\_traffic\_light)}

## Pedestrian Line coupled model

∀ j ∈ {1,2}

Pedestrian\_line\_j = <Xj, Yj, Dj, Mdj, EICj, EOCj, ICj>

Where

Xj = {(T | (T.to[i] ∈ {PB1, PB2, LL1, RL1, LL2, RL2}) && (T.from = model\_input) && (T.amount ∈ N)) ∪ (T | (T.to[i] ∈ {PB1, PB2}) && (T.from = TC) && (T.amount ∈ N))}

Yj = {(T | (T.to[i] ∈ {TC, model\_output})&&(T.from = PBj) && (T.information ∈ {active, passive}))}

Dj = {C\_PBj}

Mdj= {MC\_PBj}

EICj = {C\_PBj}

EOCj = {C\_PBj}

ICj = {}

## Car Line coupled model

∀ j ∈ {1,2}

Car\_line\_j = <Xj, Yj, Dj, Mdj, EICj, EOCj, ICj>

Where

Xj = {(T | (T.to[i] ∈ {PB1, PB2, LL1, RL1, LL2, RL2}) && (T.from = model\_input) && (T.amount ∈ N))∪ (T | (T.to[i] ∈ {LL1, RL1, LL2, RL2}) && (T.from = CCj) && (T.amount ∈ N))}

Yj = {T | (T.to[i] = model\_output) && (T.from ∈ {LL1, RL1, LL2, RL2}) && (T.amount ∈ N) && (T.time ∈ BRITime))}

Dj  = {C\_LLj, C\_LRj}

Mdj = {MC\_LLj , MC\_LRj }

EICj = {C\_LLj, C\_LRj}

EOCj = {C\_LLj, C\_LRj}

ICj = {}

## Crossroad Controller coupled model

Crossroad\_controller = <X, Y, D, Md, EIC, EOC, IC>

Where:

X = {(T | (T.to[i] ∈ {TC, model\_output}) && (T.from ∈ {PB1, PB2}) && (T.information ∈ {active, passive}))}

Y = {(T | (T.to[i] ∈ {LL1, RL1, LL2, RL2}) && (T.from ∈ {CC1, CC2}) && (T.amount ∈ N))∪ (T | (T.to[i] ∈ {CTL1, PTL1, CTL2, PTL2, CC1, CC2}) && (T.from = TC) && (T.information = green))∪ (T | (T.to[i] ∈ {PB1, PB2}) && (T.from = TC)}

D = {C\_CC1, C\_CC2, C\_TC}

Md = {MC\_CC1, MC\_CC2, MC\_TC}

EIC = {C\_TC}

EOC = {C\_CC1, C\_CC2, C\_TC}

IC = {(C\_TC, C\_CC1), (C\_TC, C\_CC2)}

## Crossroad Traffic Light coupled model

Crossroad\_traffic\_light = <X, Y, D, Md, EIC, EOC, IC>

Where:

X = {(T | (T.to[i] ∈ {CTL1, PTL1, CTL2, PTL2,CC1,CC2}) && (T.from = TC) && (T.information= green))}

Y = { (T | (T.to[i] = model\_output)&& (T.from ∈ {CTL1, PTL1, CTL2, PTL2}) && (T.information ∈ {red, green, yellow}))}

D = {C\_TLCo, Traffic\_light\_1, Traffic\_light\_2}

Md= {MC\_TLCo, MTraffic\_light\_1 , MTraffic\_light\_2}

EIC = {C\_TLCo}

EOC = {Traffic\_light\_1, Traffic\_light\_2}

IC = {((C\_TLCo, Traffic\_light\_1), (C\_TLCo, Traffic\_light\_2)}

## Traffic Light coupled model

∀ j ∈ {1,2}

Traffic\_light\_j = <Xj, Yj, Dj, Mdj, EICj, EOCj, ICj>

Where

Xj = {(T | (T.to[i] ∈ {CTL1, PTL1, CTL2, PTL2}) && (T.from = TLCo) && (T.information ∈ {green, red}))}

Yj = {(T | (T.to[i] = model\_output) && (T.from ∈ {CTLj, PTLj,}) && (T.information ∈ {red, green, yellow}))}

Dj  = {c\_CTLj, PTLj\_c}

Mdj = {MC\_CTLj , MC\_PTLj }

EICj = {C\_CTLj, C\_PTLj}

EOCj = {C\_CTLj, C\_PTLj}

ICj = {}

## Filter & Atomic model

The set of inputs and outputs varies regarding to coupled model the filter belongs to, but we can give a general formal definition of these coupled models represented fin Figure 3.

∀ at ϵ {PB1,PB2, LL1, RL1, LL2, RL2, TC, CC1, CC2, PTL1, CTL1, PTL2, CTL2}

Filter & at = <Xat, Yat, Dat, Mdat, EICat, EOCat, ICat>

Where

Xat = {(T)}

Yat = {(T)}

Dat = {Filterat, at}

Mdat= {MFilterat , Mat }

EICat = {Filterat}

EOCat = {at}

ICat = {(Filterat, at)}

# Atomic models: PDEVS formal specification

## Filter

The filter atomic model receives the messages multicast and its output is the messages that satisfy the filter criteria.

∀ at ϵ {PB1,PB2, LL1, RL1, LL2, RL2, TC, CC1, CC2, PTL1, CTL1, PTL2, CTL2}

Filterat = <X, Y, S, δint, δext, δcon, λ, ta>

Where

X = {(T)}

Y = {(T)}

S = {idat, msg}

// “idat” is and string that identifies to which model the filter is related to

// “msg” is an auxiliar vector <T> to store the messages which has passed the filter

δint (s){

msg.clear();

ta = INF;

} // Passivates the filter until an external event arrives.

δext(x,s,e){

msg.clear(); // Clean the msg vector for the messages

for (i = 0; i<x.size(); i++){

for (j = 0; j < x[i].to.size(); j++){

if (x[i].to[j] == id){

msg.push\_back(x[i]);

}

}

} // Check if the message pass the filter

if (msg.empty()){

ta = INF; // If no messages pass the filter, passivate it.

}else {

ta = INSTANTANEOUS;

}// else, made an instantaneous internal to send the message

}

δcon(x,s,e){

δext

} // Call the external because we only made the internal to send the output

λ(s){

return msg;

}

ta(s){

return ta;

}

## Pedestrian Button atomic model

∀ j ϵ {1, 2}

PBj = <X, Y, S, δint, δext, δcon, λ, ta>

Where

X = {(T | (T.to[i] ∈ {PB1, PB2, LL1, RL1, LL2, RL2}) && (T.from = model\_input) && (T.amount ∈ N)))∪ ( T | (T.to[i] ∈ {PB1, PB2}) && (T.from = TC)))}

Y = {(T | (T.to[i] ∈ {TC, model\_output}) && (T.from = idj) && (T.information ∈ {active, passive}))}

S = {idj, state ∈ {active, passive}, sending ∈ {true, false}}

// idj is the atomic model identification

δint(s){

ta = INF;

sending = false;

} // Passivates the pedestrian button until an external event arrives.

δext(x,s,e){

if ( (all x[i].from == model\_input) && (state == passive)){

state = active;

sending = true;

} // if any pedestrian arrives and the button is not active, activate it.

// The pedestrian light is green only if the button is active.

else if ((all x[i].from == TC) && (state == active)){

state = passive;

sending = true;

} // Passivate the button when the TC send a message.

// TC sends the message when the light has to change to red.

else if (not all x[i].from == TC){

state = active;

sending = true;

} // If TC says pasivate and new pedestrians arrive, activate it.

ta = INSTANTANEOUS;

}

δcon(x,s,e){

sending = false;

δext;

} // Call the external because we only made the internal to send the output and before the confluence the output is sent

λ(s){

vector<T> output;

T msg;

if (sending == true){

msg.to= [TC, output\_model];

msg.from = idj;

msg.information = state;

output.push\_back(msg);

}

return output;

}

ta(s){

return ta;

}

## Car Line atomic model

∀ at ϵ {LL1,RL1, LL2, RL2}

at = <X, Y, S, δint, δext, δcon, λ, ta>

Where

X = {(T | (T.to[i] ∈ {PB1, PB2, LL1, RL1, LL2, RL2}) && (T.from = model\_input) && (T.amount ∈ N)))∪( T | (T.to[i] ∈ {LL1, RL1, LL2, RL2}) && (T.from ∈{CC1,CC2}) && (T.amount ∈ N)))}

Y = {(T | (T.to[i] = model\_output) && (T.from = idat) && (T.amount ∈ N) && (T.time ∈ BRITime))}

S = {idat, time\_items\_queue, previous\_items\_in\_queue\_0 ∈ {true,false}}

// idj is the atomic model identification

// time\_items\_queue is a vector that stores the time each car has been waiting in the line so far.

δint(s) {

for (i = 0; i < time\_items\_queue.size(); i++){

time\_items\_queue[i] = time\_items\_queue[i] + INSTANTANEOUS;

ta = INF;

}

}

δext(x,s,e) {

if(! time\_items\_in\_queue.empty()){

for (i = 0; i < time\_items\_queue.size(); i++){

time\_items\_queue[i] = time\_items\_queue[i] + elapsed\_time;

previous\_items\_in\_queue\_0 = false;

} // update the time the cars have been in the queue

}else {

previous\_items\_in\_queue\_0 = true;

}

if ( (all x[i].from == model\_input){

for (i=0; i < x.size(); i++){

for (j=0; j < X[i].amount; j++){

time\_items\_queue.push\_back(0);

}

}

} // if only cars arriving add, them to the line with waiting time cero.

else if ((all x[i].from == CC1 or all x[i].from == CC2)){

for (i=0; i < x.size(), i++){

for (j=0; j < x[i].amount; j++){

if (! time\_items\_queue.empty()) {

time\_items\_queue.erase(time\_items\_queue.begin()); //remove the first

}

}

}

}// if only cars leaving, remove them from the line.

//The model only recieves messages from CC1 or CC2, never from both

else if (not all x[i].from == model\_input){

for (i=0; i < x.size(); i++){

if ( x[i].from == model\_input){

for (j=0; j < x[i].amount; j++){

time\_items\_queue.push\_back(0);

}

}

}

for (i=0; i < x.size(); i++){

if ( x[i].from == CC1 or x[i].from == CC2){

for (j=0; j < x[i].amount; j++){

if (! time\_items\_queue.empty()) {

time\_items\_queue.erase(time\_items\_queue.begin());

}

}

}

} // If there are cars arriving and leaving, add the same time, first add and then remove.

}

ta = INSTANTANEOUS;

}

δcon (x,s,e){

δext;

} // Call the external because we only made the internal to send the output. Before the confluence the output is sent

λ(s){

vector<T> output;

T msg;

if (! time\_items\_queue.empty()){

msg.to.push\_back(string("output\_model"));

msg.from = idat;

msg.amount = time\_items\_queue.size();

msg.time = time\_items\_queue[0];

output.push\_back(msg);

}else if (previous\_items\_in\_queue\_0 == false){

msg.to.push\_back(string("output\_model"));

msg.from = idat;

msg.amount = 0;

msg.time = 0;

output.push\_back(msg);

}

return output;

}// Send the number of cars and how long the fist car in the line has been waiting if the line

ta(s){

return ta;

}

## Traffic Controller atomic model

TC = <X, Y, S, δint, δext, δcon, λ, ta>

Where

X = {(T | (T.to[i] ∈ {TC, model\_output}) && (T.from ∈ {PB1, PB2}) && (T.information ∈ {active, passive}))}

Y = {(T | (T.to[i] ∈ {CTL1, PTL1, CTL2, PTL2, CC1, CC2}) && (T.from = id) && (T.information = green))∪ (T | (T.to[i] ∈ {PB1, PB2}) && (T.from = id) && (T.amount ∈ N))}

S = {car\_state ∈ {CTL1,CTL2,red}, pedestrian\_state ∈ {PTL1, PTL2, red}, pb\_state\_1 ∈ {active, passive}, pb\_state\_2 ∈ {active, passive}, sending ∈ {true, false}, cicle\_time ∈ BRITime, id = TC, aux}

// cicle\_time in the model must be grater than the sum of time in yellow of the car traffic light plus time in yellow of the

//pedestrian traffic light plus dealy in changing to green (these are parameters of other atomic models)

// aux is a string variable to keep if we have to send a message to a PB to change its state to passive

δint (s) {

aux.clear ();

if (sending == true){

sending = false;

ta= cicle\_time;

} // the message has just been sent.

else{

sending = true;

ta= INSTANTANEOUS;

if (car\_state == red){

car\_state = CTL2;

if (pb\_state\_2 == active){

pedestrian\_state = PTL2;

}else {

pedestrian\_state = red;

} // when initializing both traffic lights are red.

}else if (car\_state == CTL1){

car\_state = CTL2; // Change the colour of the CTL

if (pedestrian\_state == PTL1){

aux.pushback(PB1); // Pasivate PB1 is the PTL1 was green

}

if (pb\_state\_2 == active){

pedestrian\_state = PTL2; // Put green PTL2 is the PB2 is active

}else{

pedestrian\_state = red; // Both PTL are red

}

} else if (car\_state == CTL2){

car\_state == CTL1; // Change the colour of the CTL

if (pedestrian\_state == PTL2 ){

aux.pushback(PB2); // Pasivate PB2 is the PTL2 was green

}

if (pb\_state\_1 == active){

pedestrian\_state = PTL1; // Put green PTL1 is the PB1 is active

}else{

pedestrian\_state = red; // Both PTL are red

}

}

}

}

δext(x,s,e){

for (i = 0; i < x.size(); i++){

if((x[i].from == PB1) && (x[i].information == active)){

pb\_state\_1 = active;

}else if ((x[i].from == PB2) && (x[i].information == active)){

pb\_state\_2 = active;

}else if((x[i].from == PB1) && (x[i].information == passive)){

pb\_state\_1 = passive;

}else if ((xi].from == PB2) && (x[i].information == passive)){

pb\_state\_2 = passive;

}

}

ta = ta – e;

}// Actualize the state of the pedestrian buttons

δcon(x,s,e){

if (ta != INSTANTANEOUS){

δext;

δint;}// if the confluence is due to a change in the semaphore, actualize the buttons

//then change the state of the traffic light.

else{

sending = false;

δext;

ta= cicle\_time;

}// before the confluence the message has just been sent. Actualize the buttons. Actualize ta to cicle\_time.

}

λ(s){

vector<T> output;

if (sending == true){

T msg\_a;

T msg\_b;

msg\_a.from = id;

msg\_a.to.pushback(car\_state);

msg\_a.to.pushback(CC1);

msg\_a.to.pushback(CC2);

if (pedestrian\_state != red){

msg\_a.to.pushback(pedestrian\_state);

}

msg\_a.information = green;

output.pushback(msg\_a);

if(!aux.empty()){

msg\_b.from = TC;

msg\_b.to.pushback(aux[0]);

output.pushback(msg\_b);

}

}

return output;

}// Which traffic lights have to change to green and which pedestrian button has to passivate

ta(s){

return ta;

}

## Crossed Cars atomic model

∀ j ϵ {1, 2}

CCj = <X, Y, S, δint, δext, δcon, λ, ta>

Where

X = {(T | (T.to[i] ∈ {CTL1, PTL1, CTL2, PTL2, CC1, CC2}) && (T.from = TC) && (T.information = green))}

Y={(T | (T.to[i] ∈ {LL1, RL1, LL2, RL2}) && (T.from = idj) && (T.amount ∈ N)))}

S = {time\_to\_cross ∈ BRITime, state ∈ {active, passive, transition}, idj, tl\_id\_responsible, car\_line\_ids\_responsible, time\_yellow\_car, time\_yellow\_pedestrians, delay\_to\_green, time\_to\_passive}

//time\_to\_cross is the time its take a car to cross the intersection

//idj is the atomic model identification

//tl\_id\_responsible is the id of the car traffic light the crossed car atomic model is in charge

//car\_line\_ids\_responsible is a vector that contains the ids of the lines the crossed cars controls.

// time\_yellow\_car is the time the car traffic light will be in yellow

// time\_yellow\_pedestrians is the time the pedestrian traffic light will be in yellow

// delay\_to\_green of the traffic lights to change to green when all the other lights are in red

// time\_to\_passive is the time remains for the car traffic light to change to red.

δint(s){

if( state == transition){

time\_to\_passive = time\_to\_passive - ta;

if( time\_to\_passive > 0){

ta = time\_to\_cross;

} else {

ta = INF;

state = passive;

}

}else if (state == active){

ta = time\_to\_cross;

}

}

δext(x,s,e){

if( tl\_id\_responsible is a member of x[0].to){

// After the filter will recieve a single message per time

state=active;

ta= time\_yellow\_car + time\_yellow\_pedestrians + delay\_to\_green;

}//let cars cross when the traffic light has changed to geen.

else{

state = transition;

time\_to\_passive = in\_yellow\_pedestrian;

if((ta - e) > 0){

ta = ta - e;

}else{

ta = time\_to\_cross;

}

}

δcon(x,s,e){

δext;

}//if it is the case, δcon let the car cross before calling the external.

λ(s){

vector<T> output;

if (state != passive){

T msg;

msg.from = idj

msg.to = car\_line\_ids\_responsible;

msg.amount = 1;

output.pushback(msg);

}

return output;

}// tells the proper car line that a car can be leaving the intersection.

ta(s){

return ta;

}

## Traffic Light Coordinator atomic model

TLCo = <X, Y, S, δint, δext, δcon, λ, ta>

Where

X = {(T | (T.to[i] ∈ {CTL1, PTL1, CTL2, PTL2,CC1,CC2}) && (T.from = TC) && (T.information = green))}

Y = {(T | (T.to[i] ∈ {CTL1, PTL1, CTL2, PTL2}) && (T.from = id) && (T.information ∈ {green, red}))}

S = {output, id = TLCo}

//id is the atomic model identification

//output is a variable which stores the messages to be sent by the TLCo

δint(s){

output.clear();

ta= INF;

}//wait until a new instruction arrives.

δext(x,s,e){

// Only one instruction per time unit can arrive

T aux\_msg\_1;// definition of two auxiliary T structures

T aux\_msg\_2;

output.clear();

if (CTL1 is member of x[0].to){

aux\_msg\_1.to[0]=CTL1;

aux\_msg\_1.from = id;

aux\_msg\_1.information = green;

if (PTL1 is member of x[0].to){

aux\_msg\_1.to.pushback (PTL1);

}

aux\_msg\_2.to= [CTL2, PTL2];

aux\_msg\_2.from = id;

aux\_msg\_2.information = red;

} else if (CTL2 is member of X[0].to){

aux\_msg\_1.to[0]=CTL2;

aux\_msg\_1.from = id;

aux\_msg\_1.information = green;

if (PTL2 is member of X[0].to){

aux\_msg\_1.to.pushback (PTL2);

}

aux\_msg\_2.to= [CTL1, PTL1];

aux\_msg\_2.from = id;

aux\_msg\_2.information = red;

}

output.pushback(aux\_msg\_1);

output.pushback(aux\_msg\_2);

ta = INSTANTENEOUS;

}// Calculates and stores which has to be the state of each light.

δcon(x,s,e){

assert(false && "The definition of the model only allows receiving an instruction to change the state when the coordinator is stable");

}

λ(s){

return output;

} // Send which has to be the state of each light

ta(s){

return ta;

}

## Car Traffic Light atomic model

When instantiating the atomic models, the time in yellow of both cars traffic lights are the same. The time in yellow of both pedestrian traffic lights are also be the same. The delay of each semaphore to change to green is also the same for the four lights.

∀ j ϵ {1, 2}

CTLj = <X, Y, S, δint, δext, δcon, λ, ta>

Where

X = {(T | (T.to[i] ∈ {CTL1, PTL1, CTL2, PTL2}) && (T.from = TLCo) && (T.information ∈ {green, red}))}

Y = {(T | (T.to[i] = model\_output) && (T.from == idj) && (T.information ∈ {red, green, yellow}))}

S = {state ∈ {red, green, yellow}, next\_state ∈ {red, green, yellow, no\_next\_state}, idj, time\_yellow\_car, time\_yellow\_pedestrians, delay\_to\_green}

// idj is the atomic model identification

// time\_yellow\_car is the time the car traffic light will be in yellow

// time\_yellow\_pedestrians is the time the pedestrian traffic light will be in yellow

// delay\_to\_green of the traffic lights to change to green when all the other lights are in red

//Only one instruction per unit time can arrive.

δint(s){

state = next\_state;

if (next\_state == green or next\_state == red){

next\_state = no\_next\_state;

ta = INF;

} else if (next\_state ==yellow){

next\_state = red;

ta = time\_yellow\_car;

}

}

δext(x,s,e){

if (state != x[0].information){

if ( x[0].information == green){

next\_state = green;

ta = time\_yellow\_car + time\_yellow\_pedestrians + delay\_to\_green

}else if ( x[0].information == red){

next\_state = yellow;

ta = time\_yellow\_pedestrians;

}

}

}

δcon (x,s,e) {

assert(false && "The definition of the model only allows receiving an instruction to change the state when it is stable in red or green");

}

λ(s){

Vector <T> output;

T msg;

msg.from = idj;

msg.to[0] = model\_output;

msg.information = next\_state;

output.pushback(msg);

return output;

}

ta(s){

return ta;

}

## Pedestrian Traffic Light atomic model

This model is very similar to the Car Traffic Light atomic model. It only differs in the time transitions of the different states.

When instantiating the atomic models, the time in yellow of both cars traffic lights must be the same. The time in yellow of both pedestrian traffic lights must also be the same. The delay of each semaphore to change to green must also be the same for the four lights.

∀ j ϵ {1, 2}

PTLj = <X, Y, S, δint, δext, δcon, λ, ta>

Where

X = {(T | (T.to[i] ∈ {CTL1, PTL1, CTL2, PTL2}) && (T.from = TLCo) && (T.information ∈ {green, red}))}

Y = {(T | (T.to[i] = model\_output) && (T.from == idj) && (T.information ∈ {red, green, yellow}))}

S = {state ∈ {red, green, yellow}, next\_state ∈ {red, green, yellow, no\_next\_state}, idj, time\_yellow\_car, time\_yellow\_pedestrians, delay\_to\_green}

// idj is the atomic model identification

// time\_yellow\_car is the time the car traffic light will be in yellow

// time\_yellow\_pedestrians is the time the pedestrian traffic light will be in yellow

// delay\_to\_green of the traffic lights to change to green when all the other lights are in red

//Only one instruction per unit time can arrive.

δint(s){

state = next\_state;

if (next\_state == green or next\_state == red){

next\_state = no\_next\_state;

ta = INF;

} else if (next\_state ==yellow){

next\_state = red;

ta = time\_yellow\_pedestrians;

}

}

δext(x,s,e) {

if (state != x[0].information){

if ( x[0].information == green){

next\_state = green;

ta = time\_yellow\_car + time\_yellow\_pedestrians + delay\_to\_green

}else if (x[0].information == red){

next\_state = yellow;

ta = INSTANTANEOUS;

}

}

}

δcon(x,s,e) {

assert(false && "The definition of the model only allows receiving an instruction to change the state when it is stable in red or green");

}

λ(s){

Vector <T> output;

T msg;

msg.from = idj;

msg.to[0] = model\_output;

msg.information = next\_state;

output.pushback(msg);

return output;

}

ta(s) {

return ta;

}

# Testing strategy

The atomic and coupled models will be tested using the “black box” testing method. I will create different test cases with different combinations of input events, run the simulation, and check if the outputs are what we expected.

First I will test every atomic model. Second I will test the coupled models: the traffic light, the car line, the crossroad controller and the crossroad traffic light. Finally I will carry tests doing hierarchical expansion of the model: I will test the pedestrian buttons plus the crossroad controller, then I will add the car lines and finally the crossroad traffic light.

The test cases and the results are explained in the following section.

1. Simulation model

Once the model has been implemented, different tests have been carried to test atomic models, coupled models and the top model.

In the estimation of the output, the time the output is generated it is rounded to the closest integer. In the simulation, this numbers are not integer to handles the delays sending messages and to avoid the problem of the simulator with advanced times equals to 0.

# Atomic model testing

As every atomic model is preceded by a filter, the atomics models will only receive the messages that they are the addressees.

## Filter atomic

The filter can receive any input. It leaves the messages passing if the filter id is a member of x.to. Otherwise it does not pass the message.

To test the filter I have set the filter id as PB1 and I have sent the following input:

5 1 [PB1] model\_input a 1

15 1 [PB2,PB1] TC green 2

20 1 [CT] CC1 red 3

22 1 [PB2] model\_input active 4

22 1 [PB1,CT] CC2 active 5

25 1 [CC1,CC2,TL1,PB1] model\_input passive 6

25 1 [CC1,CC2,TL1] model\_input passive 7

which is translated to the model input format as follow:

event time (5/1)

msg{

msg.to= [PB1]

msg.from = model\_input

msg.information = a

msg.amount = 1

}//msg.time is not relevant

The filter output must be the messages received at time 5, 15, 22 and 25.

The output of the model is:

Creating the filter atomic model

filter input:

5 1 [PB1] model\_input a 1

15 1 [PB2,PB1] TC green 2

20 1 [CT] CC1 red 3

22 1 [PB2] model\_input active 4

22 1 [PB1,CT] CC2 active 5

25 1 [CC1,CC2,TL1,PB1] model\_input passive 6

25 1 [CC1,CC2,TL1] model\_input passive 7

Preparing runner

Starting simulation until time: 500/1seconds

500001/100000 to: [ PB1 ]

from: model\_input

information: a

amount: 1

time:0/1

1500001/100000 to: [ PB2 PB1 ]

from: TC

information: green

amount: 2

time:0/1

2200001/100000 to: [ PB1 CT ]

from: CC2

information: active

amount: 5

time:0/1

2500001/100000 to: [ CC1 CC2 TL1 PB1 ]

from: model\_input

information: passive

amount: 6

time:0/1

Finished simulation with time: infsec

Simulation took: 0.000186165sec

We check that only the proper messages pass the filter.

## Pedestrian Button atomic model

The pedestrian button changes its states according the messages it receives. If it receives a message from the model input it changes its state variable to active, if it receives a messages from the TC it changes its state to passive and if it receives a messages from both model input and TC its state changes to active.

The output of the model is sent to TC. The relevant information from the output structure message is to, from and information (which sends the state of the button). The message is sent to both TC and model output when the state of the pedestrian button or when the pedestrian button receives a message both from the model input and TC.

To test the pedestrian button I have set the pedestrian button id as PB1 and I have sent the following input:

5 1 model\_input

10 1 model\_input

15 1 TC

20 1 model\_input

21 1 model\_input

22 1 model\_input

25 1 model\_input

25 1 TC

30 1 TC

35 1 model\_input

40 1 TC

40 1 model\_input

which is translated to the model input format as follow:

event time (5/1)

msg{

msg.from = model\_input

} //the other msg components are not relevant. The pedestrian button takes its decision based on

// msg.from.

The pedestrian button output must be (time the output is generated - information): 5 – active // 15 – passive // 20 – active // 25 – active // 30 – passive // 35 – active // 40 – active. It must be sent to TC and model output.

The output of the model is:

Creating the pb atomic model

pb input:

5 1 model\_input

10 1 model\_input

15 1 TC

20 1 model\_input

21 1 model\_input

22 1 model\_input

25 1 model\_input

25 1 TC

30 1 TC

35 1 model\_input

40 1 TC

40 1 model\_input

Preparing runner

Starting simulation until time: 500/1seconds

500001/100000 to: [ TC output\_model ]

from: PB1

information: active

amount: 0

time:0/1

1500001/100000 to: [ TC output\_model ]

from: PB1

information: passive

amount: 0

time:0/1

2000001/100000 to: [ TC output\_model ]

from: PB1

information: active

amount: 0

time:0/1

2500001/100000 to: [ TC output\_model ]

from: PB1

information: active

amount: 0

time:0/1

3000001/100000 to: [ TC output\_model ]

from: PB1

information: passive

amount: 0

time:0/1

3500001/100000 to: [ TC output\_model ]

from: PB1

information: active

amount: 0

time:0/1

4000001/100000 to: [ TC output\_model ]

from: PB1

information: active

amount: 0

time:0/1

Finished simulation with time: infsec

Simulation took: 0.000476006sec

The output of the simulation matches to the expected output.

## Car Line atomic model

The Car Line atomic models receives the cars arriving and leaving the crossroad and sends to the output of the model how many cars are waiting to cross in the line and the time the first car in the line has been waiting in the intersection each time cars arrives or leaves unless if the queue is empty and this information has already been sent to the output of the model.

To test the car line I have set the car line id as LL1 and I have sent the following input:

5 1 model\_input 1

10 1 model\_input 2

15 1 CC1 1

20 1 model\_input 1

21 1 CC1 1

22 1 CC1 1

23 1 CC1 1

24 1 CC1 1

25 1 CC1 1

25 1 model\_input 3

40 1 CC1 1

40 1 model\_input 2

which is translated to the model input format as follow:

event time (5/1)

msg{

msg.from = model\_input

msg.amount = 1

} //the other msg components are not relevant. The car line takes its decision based on msg.from and

// msg.amount.

The car line output must be (time the output is generated – amount - time): 5 – 1 -0 // 10 – 3 – 5 // 15 – 2 – 5 // 20 – 3 – 10 // 21 – 2 – 11 // 22 – 1 – 2 // 23 – 0 – 0 // 25 – 2 – 0 // 40 – 3 -15. It must be sent to model output.

Amount represents the number of cars in the line and time represents the time the first car in the line has been waiting in the intersection.

The output of the model is:

Creating the car\_line atomic model

carline input:

5 1 model\_input 1

10 1 model\_input 2

15 1 CC1 1

20 1 model\_input 1

21 1 CC1 1

22 1 CC1 1

23 1 CC1 1

24 1 CC1 1

25 1 CC1 1

25 1 model\_input 3

40 1 CC1 1

40 1 model\_input 2

Preparing runner

Starting simulation until time: 500/1seconds

500001/100000 to: [ output\_model ]

from: LL1

information:

amount: 1

time:0/1

1000001/100000 to: [ output\_model ]

from: LL1

information:

amount: 3

time:5/1

1500001/100000 to: [ output\_model ]

from: LL1

information:

amount: 2

time:5/1

2000001/100000 to: [ output\_model ]

from: LL1

information:

amount: 3

time:10/1

2100001/100000 to: [ output\_model ]

from: LL1

information:

amount: 2

time:11/1

2200001/100000 to: [ output\_model ]

from: LL1

information:

amount: 1

time:2/1

2300001/100000 to: [ output\_model ]

from: LL1

information:

amount: 0

time:0/1

2500001/100000 to: [ output\_model ]

from: LL1

information:

amount: 2

time:0/1

4000001/100000 to: [ output\_model ]

from: LL1

information:

amount: 3

time:15/1

Finished simulation with time: infsec

Simulation took: 0.00063207sec

The output of the simulation matches to the expected output.

## Crossed Cars atomic model

The crossed cars atomic model estimates the cars which can be crossing the intersection. Its input is which traffic light will change to green. It generates an output every time a car can be crossing the intersection. The crossed car model does not take into account if the line is empty or not. It only says that in a certain moment the car can be crossing.

To test the crossed car I have set the parameters as follow:

id = CC1

time\_to\_cross = 1

TL\_id\_resonsible = PTL1

car\_line\_ids\_responsible = [LL1, RL1]

in\_yellow\_car = 2

in\_yellow\_pedestrian = 3

delay\_to\_green = 1

I have sent the following input:

5 1 [CC1,CTL1]

15 1 [CC1,CTL2]

25 1 [CC1,CTL1]

35 1 [CC1,CTL2]

45 1 [CC1,CTL1]

which is translated to the model input format as follow:

event time (5/1)

msg{

msg.to = [CC1, CTL1]

} //the other msg components are not relevant. The crossed car takes its decision based on msg.to.

The crossed car output must be:

msg{

msg.to = [LL1, RL1]

msg.from = CC1

msg.amount = 1

} //the other msg components are not relevant

at the following times: 11, 12, 13, 14, 15, 16,17,18, 31, 32, 33, 34, 35, 36, 37, 38.

Amount represents the number of cars that can be crossing the intersection for the car lines under the crossed car control.

The output of the model is:

Creating the crossed\_car atomic model

cc input:

5 1 [CC1,CTL1]

15 1 [CC1,CTL2]

25 1 [CC1,CTL1]

35 1 [CC1,CTL2]

45 1 [CC1,CTL1]

Preparing runner

Starting simulation until time: 50/1seconds

11/1 to: [ LL1 RL1 ]

from: CC1

information:

amount: 1

time:0/1

12/1 to: [ LL1 RL1 ]

from: CC1

information:

amount: 1

time:0/1

13/1 to: [ LL1 RL1 ]

from: CC1

information:

amount: 1

time:0/1

14/1 to: [ LL1 RL1 ]

from: CC1

information:

amount: 1

time:0/1

15/1 to: [ LL1 RL1 ]

from: CC1

information:

amount: 1

time:0/1

16/1 to: [ LL1 RL1 ]

from: CC1

information:

amount: 1

time:0/1

17/1 to: [ LL1 RL1 ]

from: CC1

information:

amount: 1

time:0/1

18/1 to: [ LL1 RL1 ]

from: CC1

information:

amount: 1

time:0/1

31/1 to: [ LL1 RL1 ]

from: CC1

information:

amount: 1

time:0/1

32/1 to: [ LL1 RL1 ]

from: CC1

information:

amount: 1

time:0/1

33/1 to: [ LL1 RL1 ]

from: CC1

information:

amount: 1

time:0/1

34/1 to: [ LL1 RL1 ]

from: CC1

information:

amount: 1

time:0/1

35/1 to: [ LL1 RL1 ]

from: CC1

information:

amount: 1

time:0/1

36/1 to: [ LL1 RL1 ]

from: CC1

information:

amount: 1

time:0/1

37/1 to: [ LL1 RL1 ]

from: CC1

information:

amount: 1

time:0/1

38/1 to: [ LL1 RL1 ]

from: CC1

information:

amount: 1

time:0/1

Finished simulation with time: 51/1sec

Simulation took: 0.000325808sec

The output of the model matches with the expected output.

## Traffic Controller atomic model

The Traffic Controller receives the state of the pedestrian button and sends which traffic lights must switch its color to green to the crossed cars atomic models and to the Crossroad Traffic Light coupled model and send a message to change the state of the pedestrian buttons to passive, when the pedestrian traffic light has to change from green to red. It takes into account the state of the pedestrian buttons, which are the inputs to the model.

To test the traffic controller I have set id to TC and cycle time equal 3 I have sent the following input:

5 1 PB1 active

10 1 PB1 passive

15 1 PB2 active

20 1 PB1 active

25 1 PB1 passive

26 1 PB2 passive

which is translated to the model input format as follow:

event time (5/1)

msg{

msg.from = PB1

msg.information = active

} //the other msg components are not relevant for this atomic model.

The traffic controller atomic model output must be (time the output is generated – to): 0 – [CTL2, CC1, CC2,TLCo] // 3 – [CTL1, CC1, CC2,TLCo] // 6 – [CTL2, CC1, CC2,TLCo] // 9 – [CTL1, PTL1, CC1, CC2,TLCo] // 12 – [CTL2, CC1, CC2,TLCo] // 12 – [PB1] //15 – [CTL1, CC1, CC2,TLCo] // 18 – [CTL2, PTL2, CC1, CC2,TLCo] // 21 – [CTL1, PTL1, CC1, CC2,TLCo] // 21 – [PB2] // 24 – [CTL2, PTL2, CC1, CC2,TLCo] //24 – [PB1] // 27 – [CTL1, CC1, CC2,TLCo] // 27 – [PB2].

The output of the model is:

Creating the traffic\_controller atomic model

tc input:

5 1 PB1 active

10 1 PB1 passive

15 1 PB2 active

20 1 PB1 active

25 1 PB1 passive

26 1 PB2 passive

Preparing runner

Starting simulation until time: 30/1seconds

0/1 to: [ CTL2 CC1 CC2 TLCo ]

from: TC

information: green

amount: 0

time:0/1

3/1 to: [ CTL1 CC1 CC2 TLCo ]

from: TC

information: green

amount: 0

time:0/1

6/1 to: [ CTL2 CC1 CC2 TLCo ]

from: TC

information: green

amount: 0

time:0/1

9/1 to: [ CTL1 CC1 CC2 TLCo PTL1 ]

from: TC

information: green

amount: 0

time:0/1

12/1 to: [ CTL2 CC1 CC2 TLCo ]

from: TC

information: green

amount: 0

time:0/1

12/1 to: [ PB1 ]

from: TC

information:

amount: 0

time:0/1

15/1 to: [ CTL1 CC1 CC2 TLCo ]

from: TC

information: green

amount: 0

time:0/1

18/1 to: [ CTL2 CC1 CC2 TLCo PTL2 ]

from: TC

information: green

amount: 0

time:0/1

21/1 to: [ CTL1 CC1 CC2 TLCo PTL1 ]

from: TC

information: green

amount: 0

time:0/1

21/1 to: [ PB2 ]

from: TC

information:

amount: 0

time:0/1

24/1 to: [ CTL2 CC1 CC2 TLCo PTL2 ]

from: TC

information: green

amount: 0

time:0/1

24/1 to: [ PB1 ]

from: TC

information:

amount: 0

time:0/1

27/1 to: [ CTL1 CC1 CC2 TLCo ]

from: TC

information: green

amount: 0

time:0/1

27/1 to: [ PB2 ]

from: TC

information:

amount: 0

time:0/1

The output of the model matches with the expected output.

## Traffic Light Coordinator atomic model

The Traffic Light Coordinator sends the future state (green or red) of the traffic lights based on the information received by the Crossroad Controller. The model checks msg.to and tells the traffic lights in msg.to to change its state to green and the other to change to red. For the pedestrian light, if the car traffic light associated to it has to change to green, but the pedestrian one has not, no message is sent to that light.

To test the traffic light coordinator I have set id to TLCo and I have sent the following input:

5 1 [CTL1,CC1,CC2]

10 1 [CTL2,CC1,CC2,PTL2]

15 1 [CTL1,PTL1CC1,CC2]

20 1 [CTL2,CC1,CC2]

25 1 [CTL1,CC1,CC2]

30 1 [CTL2,PTL2,CC1,CC2]

which is translated to the model input format as follow:

event time (5/1)

msg{

msg.to = [CTL1, CC1, CC2]

} //the other msg components are not relevant for this atomic model.

The traffic light coordinator output must be (time the output is generated – to - information): 5 – [CTL1] – green // 5 – [CTL2, PTL2], red// 10 – [CTL2,PTL2] – green // 10 – [CTL1, PTL1], red// 15 – [CTL1, PTL1] – green // 15 – [CTL2, PTL2], red// 20 – [CTL2] – green // 20 – [CTL1, PTL1], red// 25 – [CTL1] – green // 25 – [CTL2, PTL2], red// 30 – [CTL2,PTL2] – green // 30 – [CTL1, PTL1], red//.

The output of the model is:

Creating the tlco atomic model

tlco input:

5 1 [CTL1,CC1,CC2]

10 1 [CTL2,CC1,CC2,PTL2]

15 1 [CTL1,PTL1CC1,CC2]

20 1 [CTL2,CC1,CC2]

25 1 [CTL1,CC1,CC2] s

30 1 [CTL2,PTL2,CC1,CC2]

Preparing runner

Starting simulation until time: 50/1seconds

500001/100000 to: [ CTL1 ]

from: TLCo

information: green

amount: 0

time:0/1

500001/100000 to: [ CTL2 PTL2 ]

from: TLCo

information: red

amount: 0

time:0/1

1000001/100000 to: [ CTL2 PTL2 ]

from: TLCo

information: green

amount: 0

time:0/1

1000001/100000 to: [ CTL1 PTL1 ]

from: TLCo

information: red

amount: 0

time:0/1

1500001/100000 to: [ CTL1 ]

from: TLCo

information: green

amount: 0

time:0/1

1500001/100000 to: [ CTL2 PTL2 ]

from: TLCo

information: red

amount: 0

time:0/1

2000001/100000 to: [ CTL2 ]

from: TLCo

information: green

amount: 0

time:0/1

2000001/100000 to: [ CTL1 PTL1 ]

from: TLCo

information: red

amount: 0

time:0/1

2500001/100000 to: [ CTL1 ]

from: TLCo

information: green

amount: 0

time:0/1

2500001/100000 to: [ CTL2 PTL2 ]

from: TLCo

information: red

amount: 0

time:0/1

3000001/100000 to: [ CTL2 PTL2 ]

from: TLCo

information: green

amount: 0

time:0/1

3000001/100000 to: [ CTL1 PTL1 ]

from: TLCo

information: red

amount: 0

time:0/1

Finished simulation with time: infsec

Simulation took: 0.00031643sec

The output of the simulation matches to the expected output.

## Car Traffic Light atomic model

The car traffic light models the colors of the semaphore. When it receives an instruction to change to green it changes after a delay which is the sum of several parameters. When it receives an instruction to change to red it changes after a delay equal to the time the pedestrian traffic light has been in yellow. It changes to red after the time the car traffic light has to be in yellow.

To test the car traffic light I have set the parameters as follow:

id = CTL1

in\_yellow\_car = 2

in\_yellow\_pedestrian = 3

delay\_to\_green = 1

I have sent the following input:

5 1 [CTL1,PTL1] green

15 1 [CTL1] red

25 1 [CTL1,PTL1] green

35 1 [CTL1,PTL2] red

45 1 [CTL1,PTL1] green

55 1 [CTL1] red

which is translated to the model input format as follow:

event time (5/1)

msg{

msg.to = [CTL1, PTL1]

msg.information = green

} //the other msg components are not relevant for this atomic model.

The car traffic light output must be (time the output is generated – information): 11 – green // 18 -yellow // 20 – red // 31 – green // 38 -yellow // 40 – red // 51 – green // 58 -yellow // 60 – red. All the messages must be to model output.

Creating the ctl atomic model

ctl input:

5 1 [CTL1,PTL1] green

15 1 [CTL1] red

25 1 [CTL1,PTL1] green

35 1 [CTL1,PTL2] red

45 1 [CTL1,PTL1] green

55 1 [CTL1] red

Preparing runner

Starting simulation until time: 70/1seconds

11/1 to: [ model\_output ]

from: CTL1

information: green

amount: 0

time:0/1

18/1 to: [ model\_output ]

from: CTL1

information: yellow

amount: 0

time:0/1

20/1 to: [ model\_output ]

from: CTL1

information: red

amount: 0

time:0/1

31/1 to: [ model\_output ]

from: CTL1

information: green

amount: 0

time:0/1

38/1 to: [ model\_output ]

from: CTL1

information: yellow

amount: 0

time:0/1

40/1 to: [ model\_output ]

from: CTL1

information: red

amount: 0

time:0/1

51/1 to: [ model\_output ]

from: CTL1

information: green

amount: 0

time:0/1

58/1 to: [ model\_output ]

from: CTL1

information: yellow

amount: 0

time:0/1

60/1 to: [ model\_output ]

from: CTL1

information: red

amount: 0

time:0/1

Finished simulation with time: infsec

Simulation took: 0.000251224sec

The output of the simulation matches to the expected output.

## Pedestrian Traffic Light atomic model

The pedestrian traffic light models the colors of the semaphore. When it receives an instruction to change to green it changes after a delay which is the sum of several parameters. When it receives an instruction to change to red it changes immediately to yellow. It changes to red after the time the pedestrian traffic light has to be in yellow.

To test the pedestrian traffic light I have set the parameters as follow:

id = PTL1

in\_yellow\_car = 2

in\_yellow\_pedestrian = 3

delay\_to\_green = 1

I have sent the following input:

5 1 [PTL1] green

15 1 [PTL1] red

25 1 [CTL1,PTL1] green

35 1 [PTL1,PTL2] red

45 1 [CTL1,PTL1] green

55 1 [PTL1] red

which is translated to the model input format as follow:

event time (5/1)

msg{

msg.to = [PTL1]

msg.information = green

} //the other msg components are not relevant for this atomic model.

The pedestrian traffic light output must be (time the output is generated – information): 11 – green // 15 -yellow // 18 – red // 31 – green // 35 -yellow // 38 – red // 51 – green // 55 -yellow // 58 – red. All the messages must be to model output.

Creating the ptl atomic model

ptl input:

5 1 [PTL1] green

15 1 [PTL1] red

25 1 [CTL1,PTL1] green

35 1 [PTL1,PTL2] red

45 1 [CTL1,PTL1] green

55 1 [PTL1] red

Preparing runner

Starting simulation until time: 70/1seconds

11/1 to: [ model\_output ]

from: PTL1

information: green

amount: 0

time:0/1

1500001/100000 to: [ model\_output ]

from: PTL1

information: yellow

amount: 0

time:0/1

1800001/100000 to: [ model\_output ]

from: PTL1

information: red

amount: 0

time:0/1

31/1 to: [ model\_output ]

from: PTL1

information: green

amount: 0

time:0/1

3500001/100000 to: [ model\_output ]

from: PTL1

information: yellow

amount: 0

time:0/1

3800001/100000 to: [ model\_output ]

from: PTL1

information: red

amount: 0

time:0/1

51/1 to: [ model\_output ]

from: PTL1

information: green

amount: 0

time:0/1

5500001/100000 to: [ model\_output ]

from: PTL1

information: yellow

amount: 0

time:0/1

5800001/100000 to: [ model\_output ]

from: PTL1

information: red

amount: 0

time:0/1

Finished simulation with time: infsec

Simulation took: 0.000252607sec

The output of the simulation matches to the expected output.

# Coupled model testing

## Filter & Atomic model

All the atomics models have been tested. I will only test as example a filter & atomic coupled model.

I will test the filter & pedestrian button atomic model.

To test the filter & pedestrian button I have set the pedestrian button id as PB1 and the filter id also as PB1. I have sent the following input:

5 1 [PB1, PB2] model\_input

10 1 [PB1]model\_input

15 1 [PB1,PB2] TC

18 1 [PB2] TC

20 1 [PB1, PB2] model\_input

21 1 [PB1]model\_input

22 1 [PB1] model\_input

25 1 [PB1,PB2] model\_input

25 1 [PB1,PB2] TC

27 1 [PB2] model\_input

30 1 [PB1,PB2] TC

35 1 [PB1,PB2] model\_input

40 1 [PB1,PB2] TC

40 1 [PB1,PB2] model\_input

which is translated to the model input format as follow:

event time (5/1)

msg{

msg.to = [PB1,PB2]

msg.from = model\_input

} //the other msg components are not relevant. The pedestrian button takes its decision based on

// msg.from; msg.to information is relevant for the filter.

The model has to ignore the messages sent at time 18 and 27 and give the same output as the test taken for the pedestrian button atomic model.

The output of the simulation is:

filter\_pb input:

5 1 [PB1,PB2] model\_input

10 1 [PB1]model\_input

15 1 [PB1,PB2] TC

18 1 [PB2] TC

20 1 [PB1,PB2] model\_input

21 1 [PB1]model\_input

22 1 [PB1] model\_input

25 1 [PB1,PB2] model\_input

25 1 [PB1,PB2] TC

27 1 [PB2] model\_input

30 1 [PB1,PB2] TC

35 1 [PB1,PB2] model\_input

40 1 [PB1,PB2] TC

40 1 [PB1,PB2] model\_input

Preparing runner

Starting simulation until time: 500/1seconds

250001/50000 to: [ TC output\_model ]

from: PB1

information: active

amount: 0

time:0/1

750001/50000 to: [ TC output\_model ]

from: PB1

information: passive

amount: 0

time:0/1

1000001/50000 to: [ TC output\_model ]

from: PB1

information: active

amount: 0

time:0/1

1250001/50000 to: [ TC output\_model ]

from: PB1

information: active

amount: 0

time:0/1

1500001/50000 to: [ TC output\_model ]

from: PB1

information: passive

amount: 0

time:0/1

1750001/50000 to: [ TC output\_model ]

from: PB1

information: active

amount: 0

time:0/1

2000001/50000 to: [ TC output\_model ]

from: PB1

information: active

amount: 0

time:0/1

Finished simulation with time: infsec

Simulation took: 0.000442872sec

The output of the simulation matches to the expected output.

## Car Line coupled model

To test the car line coupled model I have set the id of the car line atomic models to LL1 and RL1 respectively and I have give the same id to the filter of each atomic model. I have sent the following input:

5 1 [LL1,RL1] model\_input 1

10 1 [RL1] model\_input 2

15 1 [LL1,RL1] CC1 1

20 1 [RL1] model\_input 1

21 1 [LL1,RL1] CC1 1

22 1 [LL1,RL1] CC1 1

23 1 [LL1,RL1] CC1 1

24 1 [LL1,RL1] CC1 1

25 1 [LL1,RL1] CC1 1

25 1 [LL1] model\_input 3

40 1 [LL1,RL1] CC1 1

40 1 [LL1,RL1] model\_input 2

which is translated to the model input format as follow:

event time (5/1)

msg{

msg.to = [LL1, RL1]

msg.from = model\_input

msg.amount = 1

} //the other msg components are not relevant.

The car line coupled model output must be (time the output is generated – from - amount - time): 5 – LL1 - 1 -0 // 5 – RL1 - 1 -0 // 10 –RL1 - 3 – 5 // 15 – RL1 - 2 – 5 // 15 – LL1 - 0 – 0 // 20 – RL1 – 3 – 10 // 21 – RL1 – 2 – 11 // 22 – RL1 – 1 – 2 // 23 – RL1 – 0 – 0 // 25 - LL1 - 2 – 0 // 40 - LL1 – 3 -15 // 40 - RL1 – 1 -0. It must be sent to model output.

Amount represents the number of cars in the line and time represents the time the first car in the line has been waiting in the intersection.

The output of the simulation is:

car\_line\_coupled input:

5 1 [LL1,RL1] model\_input 1

10 1 [RL1] model\_input 2

15 1 [LL1,RL1] CC1 1

20 1 [RL1] model\_input 1

21 1 [LL1,RL1] CC1 1

22 1 [LL1,RL1] CC1 1

23 1 [LL1,RL1] CC1 1

24 1 [LL1,RL1] CC1 1

25 1 [LL1,RL1] CC1 1

25 1 [LL1] model\_input 3

40 1 [LL1,RL1] CC1 1

40 1 [LL1,RL1] model\_input 2

Preparing runner

Starting simulation until time: 50/1seconds

250001/50000 to: [ output\_model ]

from: LL1

information:

amount: 1

time:0/1

250001/50000 to: [ output\_model ]

from: RL1

information:

amount: 1

time:0/1

500001/50000 to: [ output\_model ]

from: RL1

information:

amount: 3

time:5/1

750001/50000 to: [ output\_model ]

from: LL1

information:

amount: 0

time:0/1

750001/50000 to: [ output\_model ]

from: RL1

information:

amount: 2

time:5/1

1000001/50000 to: [ output\_model ]

from: RL1

information:

amount: 3

time:10/1

1050001/50000 to: [ output\_model ]

from: RL1

information:

amount: 2

time:11/1

1100001/50000 to: [ output\_model ]

from: RL1

information:

amount: 1

time:2/1

1150001/50000 to: [ output\_model ]

from: RL1

information:

amount: 0

time:0/1

1250001/50000 to: [ output\_model ]

from: LL1

information:

amount: 2

time:0/1

2000001/50000 to: [ output\_model ]

from: LL1

information:

amount: 3

time:15/1

2000001/50000 to: [ output\_model ]

from: RL1

information:

amount: 1

time:0/1

Finished simulation with time: infsec

Simulation took: 0.00104007sec

The output of the simulation matches to the expected output.

## Traffic Light coupled model

To test the traffic light coupled model I have set the following parameters:

car traffic light:

id = CTL1

in\_yellow\_car = 2

in\_yellow\_pedestrian = 3

delay\_to\_green = 1

pedestrian traffic light:

id = PTL1

in\_yellow\_car = 2

in\_yellow\_pedestrian = 3

delay\_to\_green = 1

car filter: id = CTL1

pedestrian filter id = PTL1

I have sent the following input:

5 1 [CTL1,PTL1] green

15 1 [CTL1,PTL1] red

25 1 [CTL1] green

35 1 [CTL1,PTL2] red

45 1 [CTL1,PTL1] green

55 1 [CTL1,PTL1] red

which is translated to the model input format as follow:

event time (5/1)

msg{

msg.to = [CTL1, PTL1]

msg.information = green

} //the other msg components are not relevant for this atomic model.

The traffic light coupled model output must be (time the output is generated – from - information): 11 – CTL1 - green // 11 – PTL1 - green // 15 -PTL1 -yellow// 18 -CTL1 -yellow // 18 – PTL1 -red// 20 – CTL1 -red // 31 – CTL1 -green // 38 -CTL1 -yellow // 40 –CTL1 - red // 51 – CTL1 -green // 51 – PTL1 -green // 55 -PTL1 -yellow // 58 -CTL1 -yellow // 58 – PTL1 -red// 60 – CTL1 -red. All the messages must be to model output.

The output of the simulation is:

Creating the traffic light coupled model

traffic light coupled input:

5 1 [CTL1,PTL1] green

15 1 [CTL1,PTL1] red

25 1 [CTL1] green

35 1 [CTL1,PTL2] red

45 1 [CTL1,PTL1] green

55 1 [CTL1,PTL1] red

Preparing runner

Starting simulation until time: 70/1seconds

1100001/100000 to: [ model\_output ]

from: PTL1

information: green

amount: 0

time:0/1

1100001/100000 to: [ model\_output ]

from: CTL1

information: green

amount: 0

time:0/1

750001/50000 to: [ model\_output ]

from: PTL1

information: yellow

amount: 0

time:0/1

1800001/100000 to: [ model\_output ]

from: CTL1

information: yellow

amount: 0

time:0/1

900001/50000 to: [ model\_output ]

from: PTL1

information: red

amount: 0

time:0/1

2000001/100000 to: [ model\_output ]

from: CTL1

information: red

amount: 0

time:0/1

3100001/100000 to: [ model\_output ]

from: CTL1

information: green

amount: 0

time:0/1

3800001/100000 to: [ model\_output ]

from: CTL1

information: yellow

amount: 0

time:0/1

4000001/100000 to: [ model\_output ]

from: CTL1

information: red

amount: 0

time:0/1

5100001/100000 to: [ model\_output ]

from: PTL1

information: green

amount: 0

time:0/1

5100001/100000 to: [ model\_output ]

from: CTL1

information: green

amount: 0

time:0/1

2750001/50000 to: [ model\_output ]

from: PTL1

information: yellow

amount: 0

time:0/1

5800001/100000 to: [ model\_output ]

from: CTL1

information: yellow

amount: 0

time:0/1

2900001/50000 to: [ model\_output ]

from: PTL1

information: red

amount: 0

time:0/1

6000001/100000 to: [ model\_output ]

from: CTL1

information: red

amount: 0

time:0/1

Finished simulation with time: infsec

Simulation took: 0.000916538sec

The output of the simulation matches to the expected output.

## Crossroad Traffic Light coupled model

To test the crossroad traffic light coupled model I have set the following parameters:

car traffic light 1:

id = CTL1

in\_yellow\_car = 2

in\_yellow\_pedestrian = 3

delay\_to\_green = 1

car traffic light 2:

id = CTL2

in\_yellow\_car = 2

in\_yellow\_pedestrian = 3

delay\_to\_green = 1

pedestrian traffic light 1:

id = PTL1

in\_yellow\_car = 2

in\_yellow\_pedestrian = 3

delay\_to\_green = 1

pedestrian traffic light 2:

id = PTL2

in\_yellow\_car = 2

in\_yellow\_pedestrian = 3

delay\_to\_green = 1

car filter 1: id = CTL1

car filter 2: id = CTL2

pedestrian filter 1 id = PTL1

pedestrian filter 2 id = PTL2

traffic coordinator id = TLCo

traffic coordinated filter id = TLCo

I have sent the following input:

5 1 [TLCo,CTL1,PTL1] green

8 1 [PB1] red

15 1 [TLCo,CTL2] green

25 1 [TLCo,CTL1] green

35 1 [TLCo,CTL2,PTL2] green

45 1 [TLCo,CTL1,PTL1] green

which is translated to the model input format as follow:

event time (5/1)

msg{

msg.to = [TLCo, CTL1, PTL1]

msg.information = green

} //the other msg components are not relevant for this atomic model.

The crossroad traffic light coupled model output must be (time the output is generated – from - information): 11 – CTL1 - green // 11 – PTL1 - green // 15 -PTL1 -yellow // 18 -CTL1 -yellow // 18 – PTL1 -red // 20 – CTL1 -red // 21 – CTL2 - green // 28 -CTL2 -yellow// 30 – CTL2 -red // 31 – CTL1 -green // 38 -CTL1 -yellow // 40 –CTL1 - red // 41 – CTL2 - green // 41 – PTL2 - green // 45 -PTL2 -yellow // 48 -CTL2 -yellow // 48 – PTL2 -red // 50 – CTL2 -red // 51 – CTL1 -green // 51 – PTL1 -green . All the messages must be to model output.

The output of the simulation is:

Creating the crossroad traffic light coupled model

crossroad traffic light coupled input:

5 1 [TLCo,CTL1,PTL1] green

8 1 [PB1] red

15 1 [TLCo,CTL2] green

25 1 [TLCo,CTL1] green

35 1 [TLCo,CTL2,PTL2] green

45 1 [TLCo,CTL1,PTL1] green

Preparing runner

Starting simulation until time: 70/1seconds

1100003/100000 to: [ model\_output ]

from: PTL1

information: green

amount: 0

time:0/1

1100003/100000 to: [ model\_output ]

from: CTL1

information: green

amount: 0

time:0/1

375001/25000 to: [ model\_output ]

from: PTL1

information: yellow

amount: 0

time:0/1

1800003/100000 to: [ model\_output ]

from: CTL1

information: yellow

amount: 0

time:0/1

450001/25000 to: [ model\_output ]

from: PTL1

information: red

amount: 0

time:0/1

2000003/100000 to: [ model\_output ]

from: CTL1

information: red

amount: 0

time:0/1

2100003/100000 to: [ model\_output ]

from: CTL2

information: green

amount: 0

time:0/1

2800003/100000 to: [ model\_output ]

from: CTL2

information: yellow

amount: 0

time:0/1

3000003/100000 to: [ model\_output ]

from: CTL2

information: red

amount: 0

time:0/1

3100003/100000 to: [ model\_output ]

from: CTL1

information: green

amount: 0

time:0/1

3800003/100000 to: [ model\_output ]

from: CTL1

information: yellow

amount: 0

time:0/1

4000003/100000 to: [ model\_output ]

from: CTL1

information: red

amount: 0

time:0/1

4100003/100000 to: [ model\_output ]

from: PTL2

information: green

amount: 0

time:0/1

4100003/100000 to: [ model\_output ]

from: CTL2

information: green

amount: 0

time:0/1

1125001/25000 to: [ model\_output ]

from: PTL2

information: yellow

amount: 0

time:0/1

4800003/100000 to: [ model\_output ]

from: CTL2

information: yellow

amount: 0

time:0/1

1200001/25000 to: [ model\_output ]

from: PTL2

information: red

amount: 0

time:0/1

5000003/100000 to: [ model\_output ]

from: CTL2

information: red

amount: 0

time:0/1

5100003/100000 to: [ model\_output ]

from: PTL1

information: green

amount: 0

time:0/1

5100003/100000 to: [ model\_output ]

from: CTL1

information: green

amount: 0

time:0/1

Finished simulation with time: infsec

Simulation took: 0.00153786sec

The output of the simulation matches to the expected output.

## Crossroad Controller coupled model

To test the crossroad controller coupled model I have set the following parameters:

Crossed cars 1

id = CC1

time\_to\_cross = 1

TL\_id\_resonsible = PTL1

car\_line\_ids\_responsible = [LL1, RL1]

in\_yellow\_car = 2

in\_yellow\_pedestrian = 3

delay\_to\_green = 1

Crossed cars 2

id = CC2

time\_to\_cross = 1

TL\_id\_resonsible = PTL2

car\_line\_ids\_responsible = [LL2, RL2]

in\_yellow\_car = 2

in\_yellow\_pedestrian = 3

delay\_to\_green = 1

Traffic controller

id =TC

cicle\_time = 10

crossed car filter 1 id = CC1

crossed car filter 2 id = CC2

traffic controller id = TC

I have sent the following input:

9 1 [TC] PB1 active

21 1 [TC] PB1 passive

25 1 [TC] PB2 active

33 1 [AY] PB2 active

which is translated to the model input format as follow:

event time (5/1)

msg{

msg.to = [TC]

msg.from = PB1

msg.information = active

} //the other msg components are not relevant for this atomic model.

The crossroad controller output must be the combination of the following ones:

Traffic Controller output must be (time the output is generated – to): 0 – [CTL2, CC1, CC2,TLCo] // 10 – [CTL1, CC1, CC2,TLCo, PTL1] // 20 – [CTL2, CC1, CC2,TLCo] // 20 – [PB1] //30 – [CTL1, CC1, CC2,TLCo] // 40 – [CTL2, PTL2, CC1, CC2,TLCo] //50 – [CTL1, CC1, CC2,TLCo] // 50 – [PB2]

The crossed car 1 output must be:

msg{

msg.to = [LL1, RL1]

msg.from = CC1

msg.amount = 1

} //the other msg components are not relevant

at the following times: 16,17,18,19,20,21,22,23,36, 37, 38,40,41,42,43.

The crossed car 2 output must be:

msg{

msg.to = [LL2, RL2]

msg.from = CC2

msg.amount = 1

} //the other msg components are not relevant

at the following times: 6,7,8,9,10,11,12,13,26,27,28,29,30,31,32,33,46,47,48,49,50.

The output of the simulation is:

Creating the crossroad\_controller coupled model

crossroad controller input:

9 1 [TC] PB1 active

21 1 [TC] PB1 passive

25 1 [TC] PB2 active

33 1 [AY] PB2 active

Preparing runner

Starting simulation until time: 51/1seconds

0/1 to: [ CTL2 CC1 CC2 TLCo ]

from: TC

information: green

amount: 0

time:0/1

600001/100000 to: [ LL2 RL2 ]

from: CC2

information:

amount: 1

time:0/1

700001/100000 to: [ LL2 RL2 ]

from: CC2

information:

amount: 1

time:0/1

800001/100000 to: [ LL2 RL2 ]

from: CC2

information:

amount: 1

time:0/1

900001/100000 to: [ LL2 RL2 ]

from: CC2

information:

amount: 1

time:0/1

10/1 to: [ CTL1 CC1 CC2 TLCo PTL1 ]

from: TC

information: green

amount: 0

time:0/1

1000001/100000 to: [ LL2 RL2 ]

from: CC2

information:

amount: 1

time:0/1

1100001/100000 to: [ LL2 RL2 ]

from: CC2

information:

amount: 1

time:0/1

1200001/100000 to: [ LL2 RL2 ]

from: CC2

information:

amount: 1

time:0/1

1300001/100000 to: [ LL2 RL2 ]

from: CC2

information:

amount: 1

time:0/1

1600001/100000 to: [ LL1 RL1 ]

from: CC1

information:

amount: 1

time:0/1

1700001/100000 to: [ LL1 RL1 ]

from: CC1

information:

amount: 1

time:0/1

1800001/100000 to: [ LL1 RL1 ]

from: CC1

information:

amount: 1

time:0/1

1900001/100000 to: [ LL1 RL1 ]

from: CC1

information:

amount: 1

time:0/1

20/1 to: [ CTL2 CC1 CC2 TLCo ]

from: TC

information: green

amount: 0

time:0/1

20/1 to: [ PB1 ]

from: TC

information:

amount: 0

time:0/1

2000001/100000 to: [ LL1 RL1 ]

from: CC1

information:

amount: 1

time:0/1

2100001/100000 to: [ LL1 RL1 ]

from: CC1

information:

amount: 1

time:0/1

2200001/100000 to: [ LL1 RL1 ]

from: CC1

information:

amount: 1

time:0/1

2300001/100000 to: [ LL1 RL1 ]

from: CC1

information:

amount: 1

time:0/1

2600001/100000 to: [ LL2 RL2 ]

from: CC2

information:

amount: 1

time:0/1

2700001/100000 to: [ LL2 RL2 ]

from: CC2

information:

amount: 1

time:0/1

2800001/100000 to: [ LL2 RL2 ]

from: CC2

information:

amount: 1

time:0/1

2900001/100000 to: [ LL2 RL2 ]

from: CC2

information:

amount: 1

time:0/1

30/1 to: [ CTL1 CC1 CC2 TLCo ]

from: TC

information: green

amount: 0

time:0/1

3000001/100000 to: [ LL2 RL2 ]

from: CC2

information:

amount: 1

time:0/1

3100001/100000 to: [ LL2 RL2 ]

from: CC2

information:

amount: 1

time:0/1

3200001/100000 to: [ LL2 RL2 ]

from: CC2

information:

amount: 1

time:0/1

3300001/100000 to: [ LL2 RL2 ]

from: CC2

information:

amount: 1

time:0/1

3600001/100000 to: [ LL1 RL1 ]

from: CC1

information:

amount: 1

time:0/1

3700001/100000 to: [ LL1 RL1 ]

from: CC1

information:

amount: 1

time:0/1

3800001/100000 to: [ LL1 RL1 ]

from: CC1

information:

amount: 1

time:0/1

3900001/100000 to: [ LL1 RL1 ]

from: CC1

information:

amount: 1

time:0/1

40/1 to: [ CTL2 CC1 CC2 TLCo PTL2 ]

from: TC

information: green

amount: 0

time:0/1

4000001/100000 to: [ LL1 RL1 ]

from: CC1

information:

amount: 1

time:0/1

4100001/100000 to: [ LL1 RL1 ]

from: CC1

information:

amount: 1

time:0/1

4200001/100000 to: [ LL1 RL1 ]

from: CC1

information:

amount: 1

time:0/1

4300001/100000 to: [ LL1 RL1 ]

from: CC1

information:

amount: 1

time:0/1

4600001/100000 to: [ LL2 RL2 ]

from: CC2

information:

amount: 1

time:0/1

4700001/100000 to: [ LL2 RL2 ]

from: CC2

information:

amount: 1

time:0/1

4800001/100000 to: [ LL2 RL2 ]

from: CC2

information:

amount: 1

time:0/1

4900001/100000 to: [ LL2 RL2 ]

from: CC2

information:

amount: 1

time:0/1

50/1 to: [ CTL1 CC1 CC2 TLCo ]

from: TC

information: green

amount: 0

time:0/1

50/1 to: [ PB2 ]

from: TC

information:

amount: 0

time:0/1

5000001/100000 to: [ LL2 RL2 ]

from: CC2

information:

amount: 1

time:0/1

Finished simulation with time: 5100001/100000sec

Simulation took: 0.00249282sec

The output of the simulation matches to the expected output.

# Integration testing

Once all the atomic and coupled models have been tested, I will carry the testing integration.

## Pedestrian buttons and Crossroad Controller coupled model

When I defined the Crossroad Controller atomic model I set up INSTANTANEOUS equal to 0 instead of a small delay to check its behavior easily. At this point, I had remembered that the simulator have problems dealing with advance times equal to 0, so I have changed this value to a very small delay.

To run the simulation I have instantiated the atomic models as follows:

Crossed cars 1

id = CC1

time\_to\_cross = 1

TL\_id\_resonsible = PTL1

car\_line\_ids\_responsible = [LL1, RL1]

in\_yellow\_car = 2

in\_yellow\_pedestrian = 3

delay\_to\_green = 1

Crossed cars 2

id = CC2

time\_to\_cross = 1

TL\_id\_resonsible = PTL2

car\_line\_ids\_responsible = [LL2, RL2]

in\_yellow\_car = 2

in\_yellow\_pedestrian = 3

delay\_to\_green = 1

Traffic controller

id =TC

cicle\_time = 10

crossed car filter 1 id = CC1

crossed car filter 2 id = CC2

traffic controller id = TC

pedestrian button 1 id = PB1

pedestrian button 2 id = PB2

pedestrian button 1 filter id = PB1

pedestrian button 2 filter id = PB2

I have sent the following input:

5 1 [PB1] model\_input

10 1 [PB2,PB1] model\_input

19 1 [PB1] model\_input

25 1 [PB2] model\_input

35 1 [PB1,PB2] model\_input

40 1 [PB2] model\_input

which is translated to the model input format as follow:

event time (5/1)

msg{

msg.to = [PB1]

msg.from = model\_input

} //the other msg components are not relevant for the input.

The output must be the combination of the following ones: (the times are rounded not to deal with the INSTANTANEOUS advance while examining the expected output of the model)

Traffic Controller output must be (time the output is generated – to): 0 – [CTL2, CC1, CC2,TLCo] // 10 – [CTL1, CC1, CC2,TLCo, PTL1] // 20 – [CTL2, CC1, CC2,TLCo, PTL2] // 20 – [PB1] //30 – [CTL1, CC1, CC2,TLCo] // 30 – [PB2] // 40 – [CTL2, PTL2, CC1, CC2,TLCo] //50 – [CTL1, PTL1, CC1, CC2,TLCo] // 50 – [PB2]

The crossed car 1 output must be:

msg{

msg.to = [LL1, RL1]

msg.from = CC1

msg.amount = 1

} //the other msg components are not relevant

at the following times: 16,17,18,19,20,21,22,23,24,36, 37, 38,40,41,42,43,44.

The crossed car 2 output must be:

msg{

msg.to = [LL2, RL2]

msg.from = CC2

msg.amount = 1

} //the other msg components are not relevant

at the following times: 6,7,8,9,10,11,12,13,14,26,27,28,29,30,31,32,33,34,46,47,48,49,50.

The pedestrian button 1 output must be (time the output is generated – information): 5 -active // 20 – passive // 35 – active.

The pedestrian button 2 output must be (time the output is generated – information): 10 -active // 30 – passive // 35 – active // 50 -passive.

The output of the simulation is:

Creating pedestrian button and crossroad controller coupled model

model input:

5 1 [PB1] model\_input

10 1 [PB2,PB1] model\_input

20 1 [PB1] model\_input

25 1 [PB2] model\_input

35 1 [PB1,PB2] model\_input

40 1 [PB2] model\_input

Preparing runner

Starting simulation until time: 51/1seconds

1/100000 to: [ CTL2 CC1 CC2 TLCo ]

from: TC

information: green

amount: 0

time:0/1

250001/50000 to: [ TC output\_model ]

from: PB1

information: active

amount: 0

time:0/1

300001/50000 to: [ LL2 RL2 ]

from: CC2

information:

amount: 1

time:0/1

350001/50000 to: [ LL2 RL2 ]

from: CC2

information:

amount: 1

time:0/1

400001/50000 to: [ LL2 RL2 ]

from: CC2

information:

amount: 1

time:0/1

450001/50000 to: [ LL2 RL2 ]

from: CC2

information:

amount: 1

time:0/1

500001/50000 to: [ CTL1 CC1 CC2 TLCo PTL1 ]

from: TC

information: green

amount: 0

time:0/1

500001/50000 to: [ LL2 RL2 ]

from: CC2

information:

amount: 1

time:0/1

500001/50000 to: [ TC output\_model ]

from: PB2

information: active

amount: 0

time:0/1

550001/50000 to: [ LL2 RL2 ]

from: CC2

information:

amount: 1

time:0/1

600001/50000 to: [ LL2 RL2 ]

from: CC2

information:

amount: 1

time:0/1

650001/50000 to: [ LL2 RL2 ]

from: CC2

information:

amount: 1

time:0/1

700001/50000 to: [ LL2 RL2 ]

from: CC2

information:

amount: 1

time:0/1

1600003/100000 to: [ LL1 RL1 ]

from: CC1

information:

amount: 1

time:0/1

1700003/100000 to: [ LL1 RL1 ]

from: CC1

information:

amount: 1

time:0/1

1800003/100000 to: [ LL1 RL1 ]

from: CC1

information:

amount: 1

time:0/1

1900003/100000 to: [ LL1 RL1 ]

from: CC1

information:

amount: 1

time:0/1

2000003/100000 to: [ CTL2 CC1 CC2 TLCo PTL2 ]

from: TC

information: green

amount: 0

time:0/1

2000003/100000 to: [ PB1 ]

from: TC

information:

amount: 0

time:0/1

2000003/100000 to: [ LL1 RL1 ]

from: CC1

information:

amount: 1

time:0/1

400001/20000 to: [ TC output\_model ]

from: PB1

information: passive

amount: 0

time:0/1

2100003/100000 to: [ LL1 RL1 ]

from: CC1

information:

amount: 1

time:0/1

2200003/100000 to: [ LL1 RL1 ]

from: CC1

information:

amount: 1

time:0/1

2300003/100000 to: [ LL1 RL1 ]

from: CC1

information:

amount: 1

time:0/1

2400003/100000 to: [ LL1 RL1 ]

from: CC1

information:

amount: 1

time:0/1

650001/25000 to: [ LL2 RL2 ]

from: CC2

information:

amount: 1

time:0/1

675001/25000 to: [ LL2 RL2 ]

from: CC2

information:

amount: 1

time:0/1

700001/25000 to: [ LL2 RL2 ]

from: CC2

information:

amount: 1

time:0/1

725001/25000 to: [ LL2 RL2 ]

from: CC2

information:

amount: 1

time:0/1

750001/25000 to: [ CTL1 CC1 CC2 TLCo ]

from: TC

information: green

amount: 0

time:0/1

750001/25000 to: [ PB2 ]

from: TC

information:

amount: 0

time:0/1

750001/25000 to: [ LL2 RL2 ]

from: CC2

information:

amount: 1

time:0/1

1500003/50000 to: [ TC output\_model ]

from: PB2

information: passive

amount: 0

time:0/1

775001/25000 to: [ LL2 RL2 ]

from: CC2

information:

amount: 1

time:0/1

800001/25000 to: [ LL2 RL2 ]

from: CC2

information:

amount: 1

time:0/1

825001/25000 to: [ LL2 RL2 ]

from: CC2

information:

amount: 1

time:0/1

850001/25000 to: [ LL2 RL2 ]

from: CC2

information:

amount: 1

time:0/1

1750001/50000 to: [ TC output\_model ]

from: PB1

information: active

amount: 0

time:0/1

1750001/50000 to: [ TC output\_model ]

from: PB2

information: active

amount: 0

time:0/1

720001/20000 to: [ LL1 RL1 ]

from: CC1

information:

amount: 1

time:0/1

740001/20000 to: [ LL1 RL1 ]

from: CC1

information:

amount: 1

time:0/1

760001/20000 to: [ LL1 RL1 ]

from: CC1

information:

amount: 1

time:0/1

780001/20000 to: [ LL1 RL1 ]

from: CC1

information:

amount: 1

time:0/1

800001/20000 to: [ CTL2 CC1 CC2 TLCo PTL2 ]

from: TC

information: green

amount: 0

time:0/1

800001/20000 to: [ LL1 RL1 ]

from: CC1

information:

amount: 1

time:0/1

820001/20000 to: [ LL1 RL1 ]

from: CC1

information:

amount: 1

time:0/1

840001/20000 to: [ LL1 RL1 ]

from: CC1

information:

amount: 1

time:0/1

860001/20000 to: [ LL1 RL1 ]

from: CC1

information:

amount: 1

time:0/1

880001/20000 to: [ LL1 RL1 ]

from: CC1

information:

amount: 1

time:0/1

2300003/50000 to: [ LL2 RL2 ]

from: CC2

information:

amount: 1

time:0/1

2350003/50000 to: [ LL2 RL2 ]

from: CC2

information:

amount: 1

time:0/1

2400003/50000 to: [ LL2 RL2 ]

from: CC2

information:

amount: 1

time:0/1

2450003/50000 to: [ LL2 RL2 ]

from: CC2

information:

amount: 1

time:0/1

2500003/50000 to: [ CTL1 CC1 CC2 TLCo PTL1 ]

from: TC

information: green

amount: 0

time:0/1

2500003/50000 to: [ PB2 ]

from: TC

information:

amount: 0

time:0/1

2500003/50000 to: [ LL2 RL2 ]

from: CC2

information:

amount: 1

time:0/1

625001/12500 to: [ TC output\_model ]

from: PB2

information: passive

amount: 0

time:0/1

Finished simulation with time: 2550003/50000sec

Simulation took: 0.00175681sec

The output of the simulation matches to the expected output.

## Pedestrian buttons, Crossroad Controller and car lines coupled model

To do this test integration, I have added car line 1 and car line 2 to the previous integration.

I have sent the following input:

5 1 [PB1] model\_input 1

5 1 [LL1,RL1] model\_input 1

10 1 [RL1] model\_input 2

10 1 [PB2,PB1] model\_input

15 1 [LL2] model\_input 3

19 1 [PB1] model\_input 2

20 1 [RL1] model\_input 1

25 1 [PB2] model\_input 2

25 1 [LL1] model\_input 3

30 1 [RL2] model\_input 4

35 1 [PB1,PB2] model\_input 3

40 1 [PB2] model\_input 1

40 1 [LL1,RL1] model\_input 2

which is translated to the model input format as follow:

event time (5/1)

msg{

msg.to = [PB1]

msg.from = model\_input

msg.amount = 1

} //the other msg components are not relevant for the input.

The output for both pedestrian buttons must be the same as in the previous step. The output for the car lines must be:

From LL1: (time the output is generated – amount - time): 5 – 1 -0 // 16 – 0 – 0 // 25 – 3 -0 // 36 – 2 -11 // 37 – 1- 12 // 38 -0 -0 // 40 – 2 – 0 // 40.000x – 1 – 0 // 41 – 0 – 0. It must be sent to model output.

From RL1: (time the output is generated – amount - time): 5 – 1 -0 // 10 – 3 – 5 // 16 – 2 – 6 // 17 – 1 – 7 // 18 -0 -0 // 20 – 1 - 0 // 20.0000x – 0 – 0 // 40 – 2 – 0 // 40.000x – 1 – 0 // 41 – 0 – 0. It must be sent to model output.

From LL2: (time the output is generated – amount - time): 15 – 3 -0 // 26 – 2 – 11 // 27 – 1 - 12 // 28 – 0 - 0. It must be sent to model output.

From RL2: (time the output is generated – amount - time): 30 – 4 -0 // 30.0000x – 3 – 0 // 31 – 2 - 1 // 32 – 1 – 2 // 33 – 0 - 0. It must be sent to model output.

The output of the simulation is:

Creating pedestrian button and crossroad controller coupled model

model input:

5 1 [PB1] model\_input 1

5 1 [LL1,RL1] model\_input 1

10 1 [RL1] model\_input 2

10 1 [PB2,PB1] model\_input

15 1 [LL2] model\_input 3

19 1 [PB1] model\_input 2

20 1 [RL1] model\_input 1

25 1 [PB2] model\_input 2

25 1 [LL1] model\_input 3

30 1 [RL2] model\_input 4

35 1 [PB1,PB2] model\_input 3

40 1 [PB2] model\_input 1

40 1 [LL1,RL1] model\_input 2

Preparing runner

Starting simulation until time: 51/1seconds

250001/50000 to: [ TC output\_model ]

from: PB1

information: active

amount: 0

time:0/1

250001/50000 to: [ output\_model ]

from: LL1

information:

amount: 1

time:0/1

250001/50000 to: [ output\_model ]

from: RL1

information:

amount: 1

time:0/1

500001/50000 to: [ TC output\_model ]

from: PB2

information: active

amount: 0

time:0/1

500001/50000 to: [ output\_model ]

from: RL1

information:

amount: 3

time:5/1

750001/50000 to: [ output\_model ]

from: LL2

information:

amount: 3

time:0/1

320001/20000 to: [ output\_model ]

from: LL1

information:

amount: 0

time:0/1

320001/20000 to: [ output\_model ]

from: RL1

information:

amount: 2

time:600003/100000

340001/20000 to: [ output\_model ]

from: RL1

information:

amount: 1

time:700003/100000

360001/20000 to: [ output\_model ]

from: RL1

information:

amount: 0

time:0/1

1000001/50000 to: [ output\_model ]

from: RL1

information:

amount: 1

time:0/1

400001/20000 to: [ TC output\_model ]

from: PB1

information: passive

amount: 0

time:0/1

400001/20000 to: [ output\_model ]

from: RL1

information:

amount: 0

time:0/1

1250001/50000 to: [ output\_model ]

from: LL1

information:

amount: 3

time:0/1

1300003/50000 to: [ output\_model ]

from: LL2

information:

amount: 2

time:275001/25000

1350003/50000 to: [ output\_model ]

from: LL2

information:

amount: 1

time:300001/25000

1400003/50000 to: [ output\_model ]

from: LL2

information:

amount: 0

time:0/1

1500001/50000 to: [ output\_model ]

from: RL2

information:

amount: 4

time:0/1

1500003/50000 to: [ TC output\_model ]

from: PB2

information: passive

amount: 0

time:0/1

1500003/50000 to: [ output\_model ]

from: RL2

information:

amount: 3

time:1/25000

1550003/50000 to: [ output\_model ]

from: RL2

information:

amount: 2

time:25001/25000

1600003/50000 to: [ output\_model ]

from: RL2

information:

amount: 1

time:50001/25000

1650003/50000 to: [ output\_model ]

from: RL2

information:

amount: 0

time:0/1

1750001/50000 to: [ TC output\_model ]

from: PB1

information: active

amount: 0

time:0/1

1750001/50000 to: [ TC output\_model ]

from: PB2

information: active

amount: 0

time:0/1

3600007/100000 to: [ output\_model ]

from: LL1

information:

amount: 2

time:220001/20000

3700007/100000 to: [ output\_model ]

from: LL1

information:

amount: 1

time:240001/20000

3800007/100000 to: [ output\_model ]

from: LL1

information:

amount: 0

time:0/1

2000001/50000 to: [ output\_model ]

from: LL1

information:

amount: 2

time:0/1

2000001/50000 to: [ output\_model ]

from: RL1

information:

amount: 2

time:0/1

4000007/100000 to: [ output\_model ]

from: LL1

information:

amount: 1

time:1/20000

4000007/100000 to: [ output\_model ]

from: RL1

information:

amount: 1

time:1/20000

4100007/100000 to: [ output\_model ]

from: LL1

information:

amount: 0

time:0/1

4100007/100000 to: [ output\_model ]

from: RL1

information:

amount: 0

time:0/1

625001/12500 to: [ TC output\_model ]

from: PB2

information: passive

amount: 0

time:0/1

Finished simulation with time: 2550003/50000sec

Simulation took: 0.00456706sec

The output of the simulation matches to the expected output.

## Crossroad top model

To test the crossroad top model I have set the following model parameters:

cicle\_time = 10;

in\_yellow\_car = 2;

in\_yellow\_pedestrian = 3;

delay\_to\_green = 1;

time\_to\_cross = 1;

I have performed two test cases in the crossroad top model. The simplest one, which is not having any input (test 1) and another one sending the following input (test 2):

5 1 [PB1] model\_input 1

5 1 [LL1,RL1] model\_input 1

10 1 [RL1] model\_input 2

10 1 [PB2,PB1] model\_input

15 1 [LL2] model\_input 3

19 1 [PB1] model\_input 2

20 1 [RL1] model\_input 1

25 1 [PB2] model\_input 2

25 1 [LL1] model\_input 3

30 1 [RL2] model\_input 4

35 1 [PB1,PB2] model\_input 3

40 1 [PB2] model\_input 1

40 1 [LL1,RL1] model\_input 2

Test 1 output is:

Creating Crossroad top model

model input:

Preparing runner

Starting simulation until time: 51/1seconds

150001/25000 to: [ model\_output ]

from: CTL2

information: green

amount: 0

time:0/1

260001/20000 to: [ model\_output ]

from: CTL2

information: yellow

amount: 0

time:0/1

300001/20000 to: [ model\_output ]

from: CTL2

information: red

amount: 0

time:0/1

320001/20000 to: [ model\_output ]

from: CTL1

information: green

amount: 0

time:0/1

1150003/50000 to: [ model\_output ]

from: CTL1

information: yellow

amount: 0

time:0/1

1250003/50000 to: [ model\_output ]

from: CTL1

information: red

amount: 0

time:0/1

1300003/50000 to: [ model\_output ]

from: CTL2

information: green

amount: 0

time:0/1

3300007/100000 to: [ model\_output ]

from: CTL2

information: yellow

amount: 0

time:0/1

3500007/100000 to: [ model\_output ]

from: CTL2

information: red

amount: 0

time:0/1

3600007/100000 to: [ model\_output ]

from: CTL1

information: green

amount: 0

time:0/1

537501/12500 to: [ model\_output ]

from: CTL1

information: yellow

amount: 0

time:0/1

562501/12500 to: [ model\_output ]

from: CTL1

information: red

amount: 0

time:0/1

575001/12500 to: [ model\_output ]

from: CTL2

information: green

amount: 0

time:0/1

Finished simulation with time: 2550003/50000sec

Simulation took: 0.00335751sec

This output is just the car traffic lights changing their color periodically as no pedestrian press the button to cross and no cars are arriving, so neither leaving, the crossroad.

For test 2, the output of the pedestrian buttons and car lines must be the same and the step 2 of the integration test, and the output coming for the car and pedestrian lights atomic models must be the following ones:

CTL2 (time the output is generated – information): 6 – green // 13 – yellow // 15 – red // 26 – green // 33 – yellow // 35 – red // 46 – green

PTL2 (time the output is generated – information): 26 – green // 30 – yellow // 33 – red // 46 – green // 50 – yellow

CTL1 (time the output is generated – information): 16 – green // 23 – yellow // 25 – red // 36 – green // 43 – yellow // 45 – red

PTL1 (time the output is generated – information): 16 – green // 20 - yellow // 23 – red.

The output of the simulation is:

Creating Crossroad top model

model input:

5 1 [PB1] model\_input 1

5 1 [LL1,RL1] model\_input 1

10 1 [RL1] model\_input 2

10 1 [PB2,PB1] model\_input

15 1 [LL2] model\_input 3

19 1 [PB1] model\_input 2

20 1 [RL1] model\_input 1

25 1 [PB2] model\_input 2

25 1 [LL1] model\_input 3

30 1 [RL2] model\_input 4

35 1 [PB1,PB2] model\_input 3

40 1 [PB2] model\_input 1

40 1 [LL1,RL1] model\_input 2

Preparing runner

Starting simulation until time: 51/1seconds

250001/50000 to: [ TC output\_model ]

from: PB1

information: active

amount: 0

time:0/1

250001/50000 to: [ output\_model ]

from: LL1

information:

amount: 1

time:0/1

250001/50000 to: [ output\_model ]

from: RL1

information:

amount: 1

time:0/1

150001/25000 to: [ model\_output ]

from: CTL2

information: green

amount: 0

time:0/1

500001/50000 to: [ TC output\_model ]

from: PB2

information: active

amount: 0

time:0/1

500001/50000 to: [ output\_model ]

from: RL1

information:

amount: 3

time:5/1

260001/20000 to: [ model\_output ]

from: CTL2

information: yellow

amount: 0

time:0/1

750001/50000 to: [ output\_model ]

from: LL2

information:

amount: 3

time:0/1

300001/20000 to: [ model\_output ]

from: CTL2

information: red

amount: 0

time:0/1

320001/20000 to: [ output\_model ]

from: LL1

information:

amount: 0

time:0/1

320001/20000 to: [ output\_model ]

from: RL1

information:

amount: 2

time:600003/100000

320001/20000 to: [ model\_output ]

from: PTL1

information: green

amount: 0

time:0/1

320001/20000 to: [ model\_output ]

from: CTL1

information: green

amount: 0

time:0/1

340001/20000 to: [ output\_model ]

from: RL1

information:

amount: 1

time:700003/100000

360001/20000 to: [ output\_model ]

from: RL1

information:

amount: 0

time:0/1

1000001/50000 to: [ output\_model ]

from: RL1

information:

amount: 1

time:0/1

400001/20000 to: [ TC output\_model ]

from: PB1

information: passive

amount: 0

time:0/1

400001/20000 to: [ output\_model ]

from: RL1

information:

amount: 0

time:0/1

2000007/100000 to: [ model\_output ]

from: PTL1

information: yellow

amount: 0

time:0/1

1150003/50000 to: [ model\_output ]

from: CTL1

information: yellow

amount: 0

time:0/1

2300007/100000 to: [ model\_output ]

from: PTL1

information: red

amount: 0

time:0/1

1250001/50000 to: [ output\_model ]

from: LL1

information:

amount: 3

time:0/1

1250003/50000 to: [ model\_output ]

from: CTL1

information: red

amount: 0

time:0/1

1300003/50000 to: [ output\_model ]

from: LL2

information:

amount: 2

time:275001/25000

1300003/50000 to: [ model\_output ]

from: PTL2

information: green

amount: 0

time:0/1

1300003/50000 to: [ model\_output ]

from: CTL2

information: green

amount: 0

time:0/1

1350003/50000 to: [ output\_model ]

from: LL2

information:

amount: 1

time:300001/25000

1400003/50000 to: [ output\_model ]

from: LL2

information:

amount: 0

time:0/1

1500001/50000 to: [ output\_model ]

from: RL2

information:

amount: 4

time:0/1

1500003/50000 to: [ TC output\_model ]

from: PB2

information: passive

amount: 0

time:0/1

1500003/50000 to: [ output\_model ]

from: RL2

information:

amount: 3

time:1/25000

375001/12500 to: [ model\_output ]

from: PTL2

information: yellow

amount: 0

time:0/1

1550003/50000 to: [ output\_model ]

from: RL2

information:

amount: 2

time:25001/25000

1600003/50000 to: [ output\_model ]

from: RL2

information:

amount: 1

time:50001/25000

1650003/50000 to: [ output\_model ]

from: RL2

information:

amount: 0

time:0/1

3300007/100000 to: [ model\_output ]

from: CTL2

information: yellow

amount: 0

time:0/1

412501/12500 to: [ model\_output ]

from: PTL2

information: red

amount: 0

time:0/1

1750001/50000 to: [ TC output\_model ]

from: PB1

information: active

amount: 0

time:0/1

1750001/50000 to: [ TC output\_model ]

from: PB2

information: active

amount: 0

time:0/1

3500007/100000 to: [ model\_output ]

from: CTL2

information: red

amount: 0

time:0/1

3600007/100000 to: [ output\_model ]

from: LL1

information:

amount: 2

time:220001/20000

3600007/100000 to: [ model\_output ]

from: CTL1

information: green

amount: 0

time:0/1

3700007/100000 to: [ output\_model ]

from: LL1

information:

amount: 1

time:240001/20000

3800007/100000 to: [ output\_model ]

from: LL1

information:

amount: 0

time:0/1

2000001/50000 to: [ output\_model ]

from: LL1

information:

amount: 2

time:0/1

2000001/50000 to: [ output\_model ]

from: RL1

information:

amount: 2

time:0/1

4000007/100000 to: [ output\_model ]

from: LL1

information:

amount: 1

time:1/20000

4000007/100000 to: [ output\_model ]

from: RL1

information:

amount: 1

time:1/20000

4100007/100000 to: [ output\_model ]

from: LL1

information:

amount: 0

time:0/1

4100007/100000 to: [ output\_model ]

from: RL1

information:

amount: 0

time:0/1

537501/12500 to: [ model\_output ]

from: CTL1

information: yellow

amount: 0

time:0/1

562501/12500 to: [ model\_output ]

from: CTL1

information: red

amount: 0

time:0/1

575001/12500 to: [ model\_output ]

from: PTL2

information: green

amount: 0

time:0/1

575001/12500 to: [ model\_output ]

from: CTL2

information: green

amount: 0

time:0/1

625001/12500 to: [ TC output\_model ]

from: PB2

information: passive

amount: 0

time:0/1

500001/10000 to: [ model\_output ]

from: PTL2

information: yellow

amount: 0

time:0/1

Finished simulation with time: 2550003/50000sec

Simulation took: 0.00419679sec

This output matches with the expected results.

# Other simulations

I have simulated other scenarios that can be run from the scripts: simulation 1 and general simulations.

The input for simulation 1 can be seen from file: crossroad\_simulation1.txt

For that simulation I have set the following parameters:

cicle\_time = 30;

in\_yellow\_car = 4;

in\_yellow\_pedestrian = 6;

delay\_to\_green = 1;

time\_to\_cross = 2;

We observer in all of them that the traffic lights change their colors from red to green to yellow and to read again alternatively. Cars do not cross the crossroad when the traffic light is red. We also have the information about if the pedestrian buttons are switched on or off and the number of cars that are waiting to cross each line and how long the first car in the line has been waiting.

The output of the crossroad top model has to be interpreted in the following way:

T {

to; // the atomic models which are addressees of the message. In the final model this is not

// shown. Only shown while testing.

from; // which atomic model sends the message.

amount; // It only has meaning if from ∈ {LL1, LL2, RL1, RL2}. It represents the numbers of

// cars waiting in the line.

Information; // It only has meaning if from ∈ {PB1, PB2,CTL1, CTL2, PTL1, PTL2}.

// It represents the state of the device. For lights their color and for buttons active or

// passive.

timer; // It only has meaning if from ∈ {LL1, LL2, RL1, RL2}.

//It represents how long the first car in the line has been waiting.

}