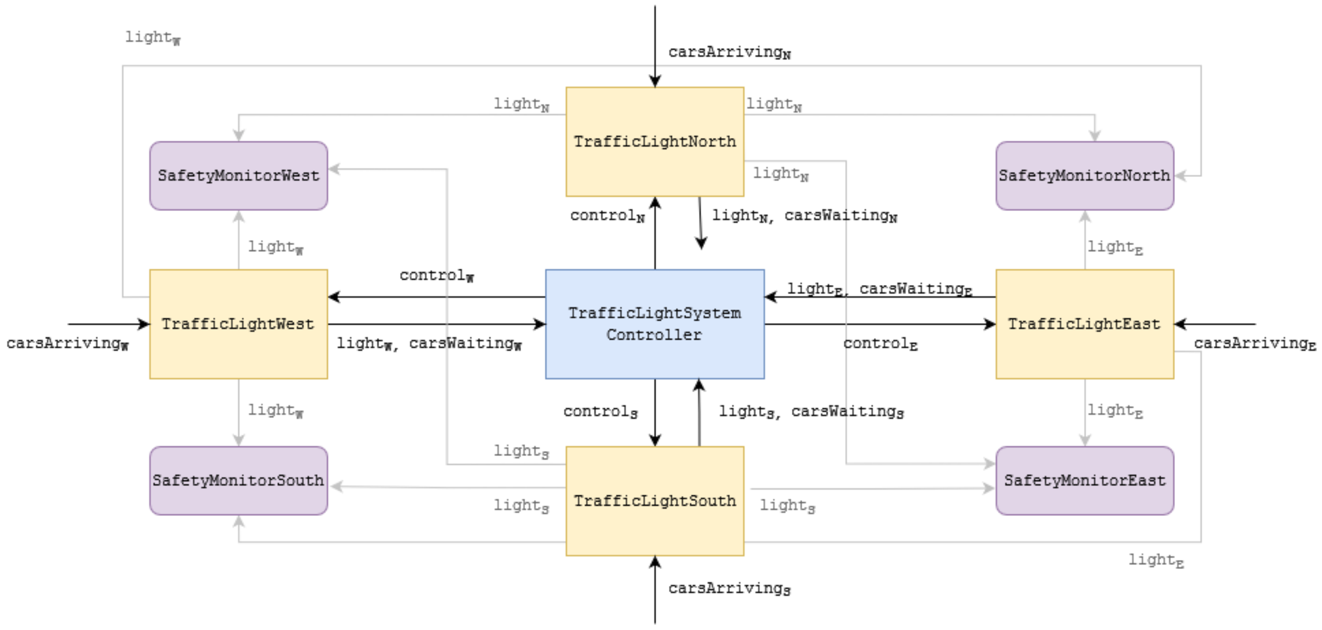
Traffic Light System
Team Members: Tabitha Lee, Rooney Gao

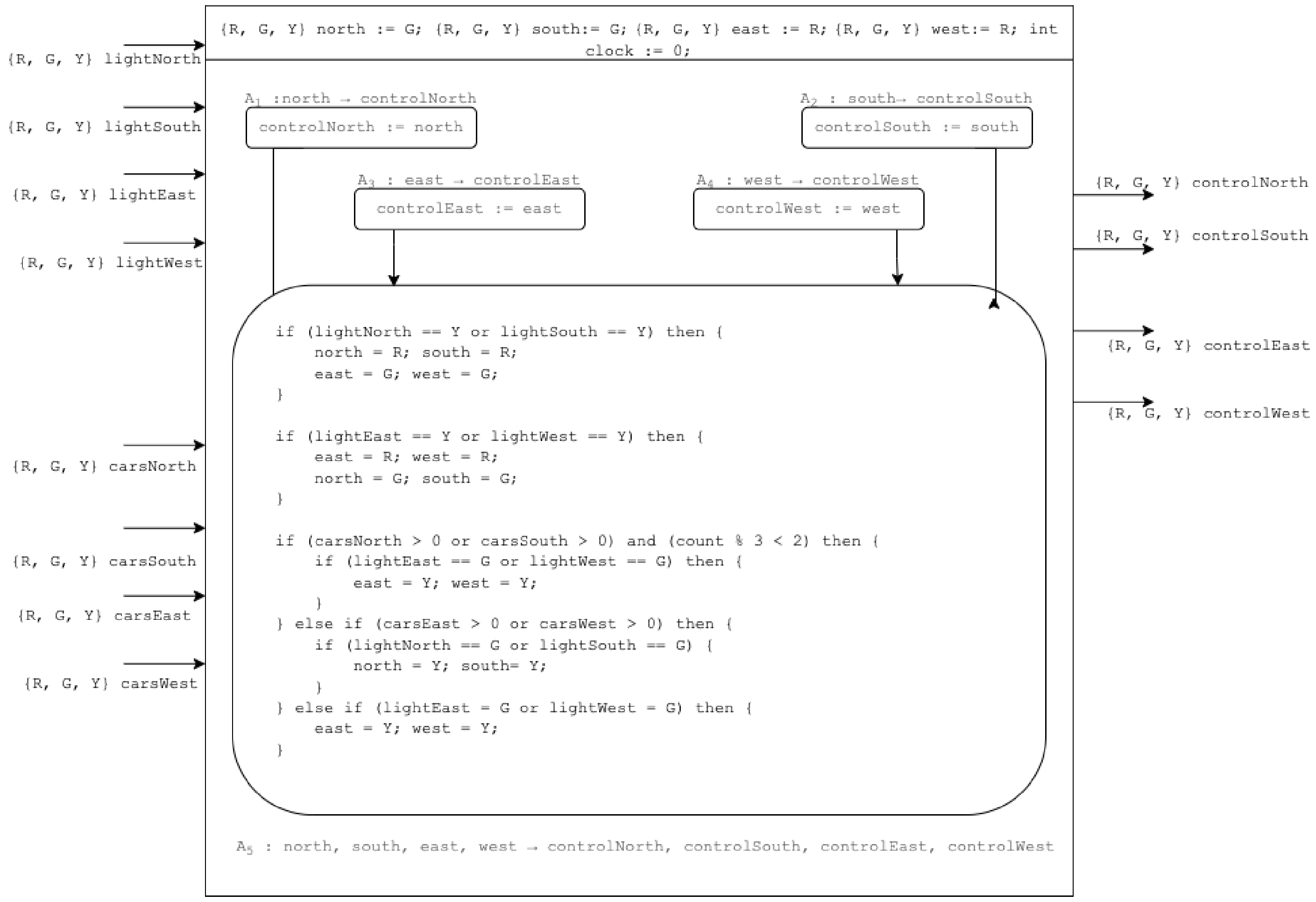
Direction



We plan on implementing a traffic light system that is composed of five main components: 4 traffic light components (TLNorth, TLSouth, TLEast, TLWest) and 1 controller component. At the moment, we do not plan on implementing left-turns in the intersection, but the intersection will consist of two lanes of traffic from the four cardinal directions.

Components

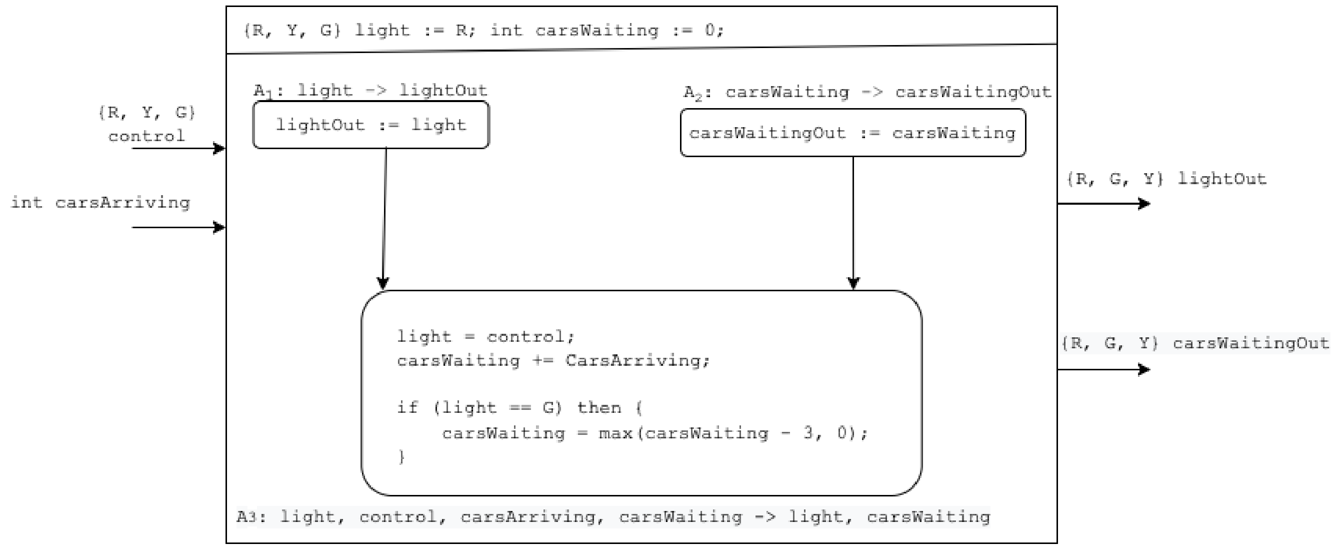
TrafficLightSystemController:
The **`TrafficLightSystemController`** component has the following inputs, states, and outputs:
I: {lightNorth, lightSouth, lightWest, lightEast, carsNorth, carsSouth, carsEast, carsWest}
S: {north, south, east, west}
O: {controlNorth, controlSouth, controlEast, controlWest}
The TrafficLightSystemController component takes in the current light and number of cars at each traffic light and issues a control signal to each traffic light. For this system, we assume that if the light is green, **1 clock cycle** allows **3 cars** to pass through the intersection. Therefore, to fulfill our liveness requirement, while we give the North and South roads priority, we prevent it from spending more than 3 clock cycles in “green” to allow the other roads a chance to turn green as well.



```
{R, G, Y} north := G; {R, G, Y} south:= G; {R, G, Y} east := R; {R, G, Y} west:= R; int
clock := 0;

A1 :north → controlNorth
controlNorth := north

A2 : south→ controlSouth
controlSouth := south

A3 : east → controlEast
controlEast := east

A4 : west → controlWest
controlWest := west

if (lightNorth == Y or lightSouth == Y) then {
    north = R; south = R;
    east = G; west = G;
}

if (lightEast == Y or lightWest == Y) then {
    east = R; west = R;
    north = G; south = G;
}

if (carsNorth > 0 or carsSouth > 0) and (count % 3 < 2) then {
    if (lightEast == G or lightWest == G) then {
        east = Y; west = Y;
    }
} else if (carsEast > 0 or carsWest > 0) then {
    if (lightNorth == G or lightSouth == G) {
        north = Y; south= Y;
    }
} else if (lightEast = G or lightWest = G) then {
    east = Y; west = Y;
}

A5 : north, south, east, west → controlNorth, controlSouth, controlEast, controlWest
```

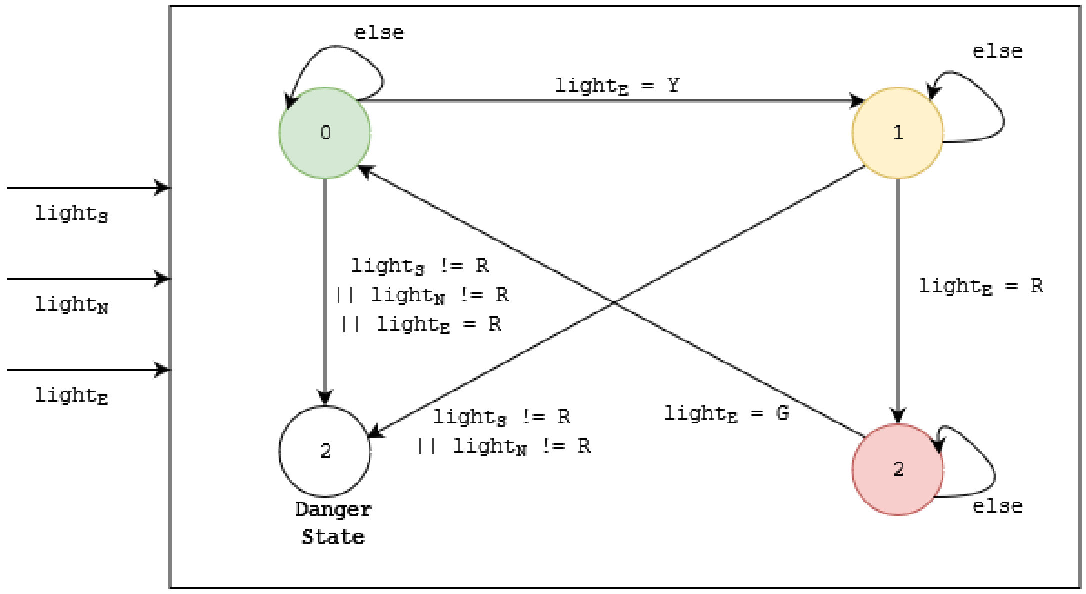
TrafficLightSystemController
This controller takes in the state of each traffic light, computes the control signals for each traffic light, stores them in internal state variables, and outputs them in the next clock cycle.

TrafficLight (North, South, East, West):
Each **`TrafficLight`** (North, South, East, West) component has the following inputs, states, and outputs:
I: {control, carsArriving}
S: {light, carsWaiting}
O: {lightOut, carsWaitingOut}
Essentially, each TrafficLight component maintains and updates its current light and number of cars waiting based on inputs and updates its current state to the TrafficLightSystemController component.


{R, Y, G} light := R; int carsWaiting := 0;
A1: light -> lightOut
lightOut := light
A2: carsWaiting -> carsWaitingOut
carsWaitingOut := carsWaiting
{R, Y, G} control
int carsArriving
{R, G, Y} lightOut
{R, G, Y} carsWaitingOut
light = control;
carsWaiting += CarsArriving;
if (light == G) then {
carsWaiting = max(carsWaiting - 3, 0);
}
A3: light, control, carsArriving, carsWaiting -> light, carsWaiting

TrafficLight(North, South, East, West) (Each TrafficLight component keeps track of the current number of cars and updates that based on the cars arriving, and whose output lightOut is the current light of the traffic light, as well as what is passed to the Controller).

SafetyMonitor (North, South, East, West):
The below implementation is specifically for TrafficLightEast as an example. The monitor will take in the current traffic light and the traffic lights from the two colliding directions.



SafetyMonitor for TrafficLight_E
(Each TrafficLight has its own Monitor that watches the opposite direction that may cause a collison. For example, TrafficLight_E looks at its own light and that of the North/South. TrafficLight_N would look at its own and that of the East/West. There are two cases where we enter the danger state: if there is a signal that the colliding direction's traffic light is green or yellow at the same time as the current traffic light is green or yellow, *or* if the current light is green and receives a signal to turn red, since it should always turn yellow first.)

Outcome

Our goal is to have a traffic light system that will fulfill all safety and liveness requirements, namely that the traffic light system will not lead to a collision of cars, and that cars will get an evenly distributed chance to pass through the intersection.

After we have designed the system, we will demonstrate through formal verification that the system fulfills both safety and liveness requirements, and also conduct various simulations on the Python implementation of the traffic light system and display their results as part of our final report.

Safety Requirement: the traffic light system will not lead to a collision of cars
ALWAYS NOT ((Light_w == Green OR Light_e == Green OR Light_w == Yellow OR Light_e == Yellow) AND (Light_n == Green OR Light_s == Green OR Light_s == Yellow OR Light_n == Yellow)

Liveness Requirement: While we prioritize the North and South Roads (i.e. they will always stay green if there are no cars at the intersection) we want to ensure that the other side will eventually turn green as well.
Always [(carsNorth > 0 or carsSouth > 0) -> Eventually(lightNorth = Green and lightSouth = Green)]
Always [(carsEast > 0 or carsWest> 0) -> Eventually(lightEast = Green and lightWest = Green)]