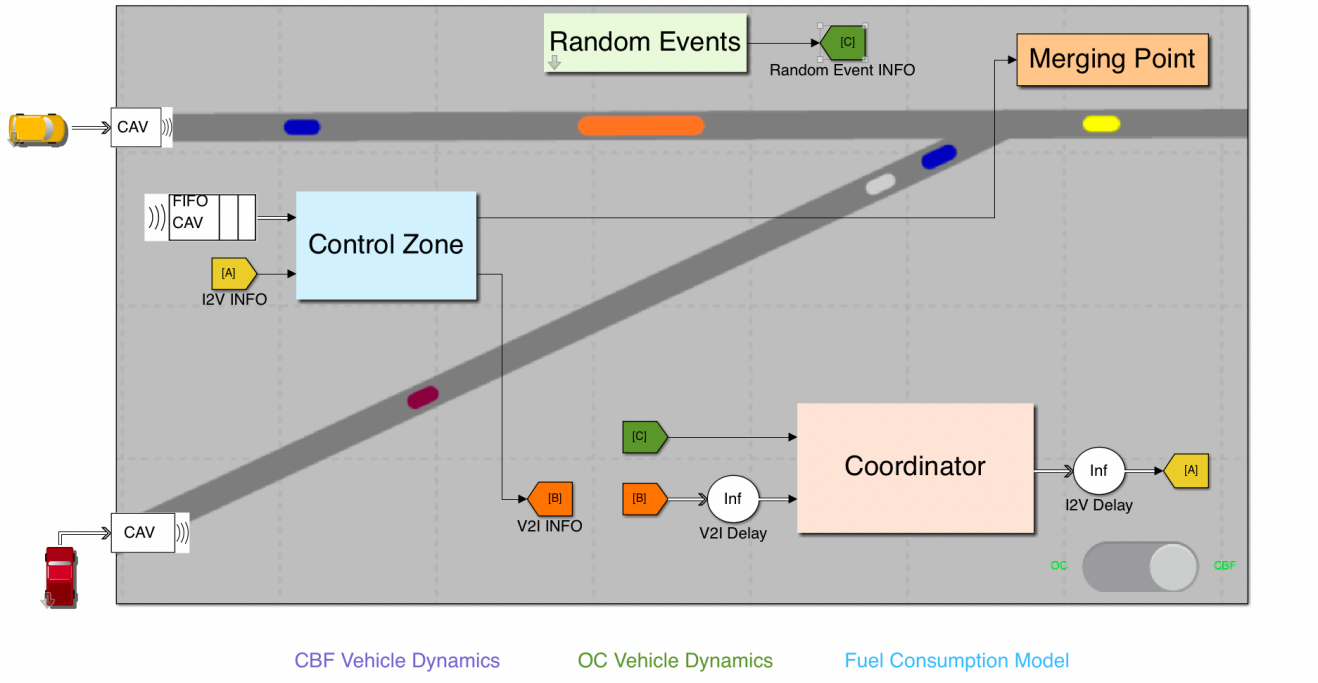
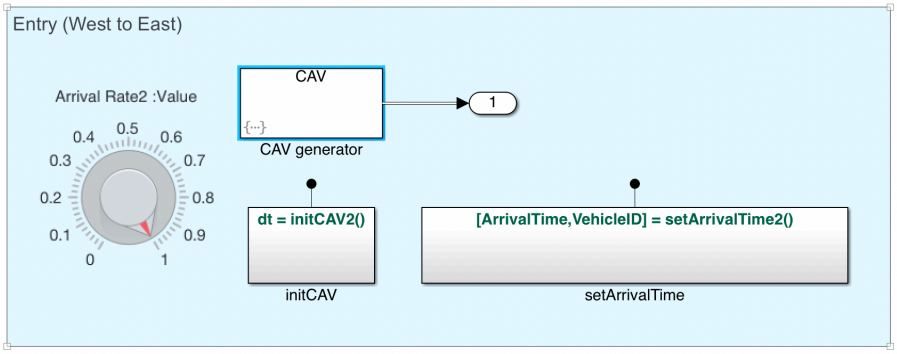
# Merging代码分析

## Simulink部分

其包括6个子系统，分别为两个车辆生成子系统，一个control zone子系统，一个coordinator子系统、一个random events子系统（实际并没有使用）和一个merging point子系统。



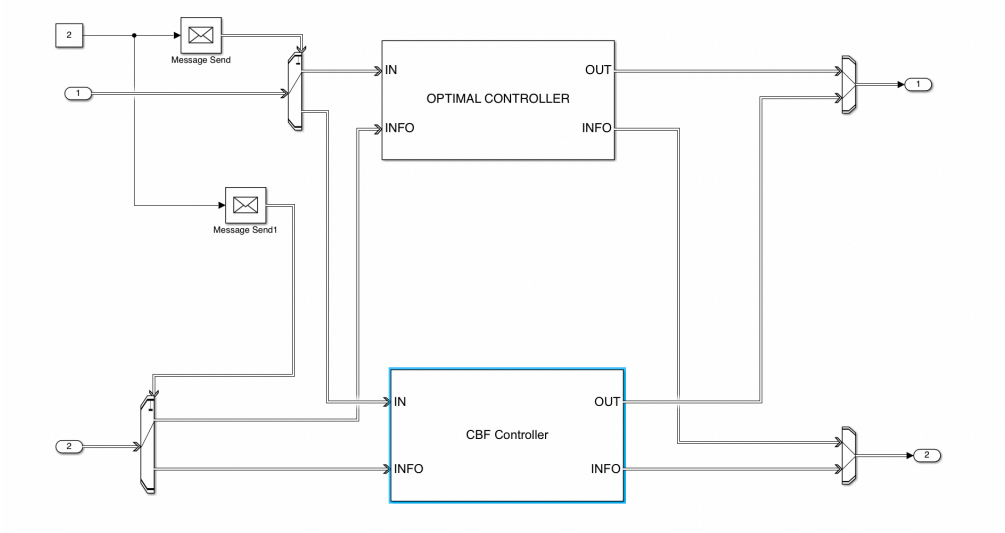

其中车辆生成子系统主要部分为entity generator，其中event action为一些车辆属性的初始化。



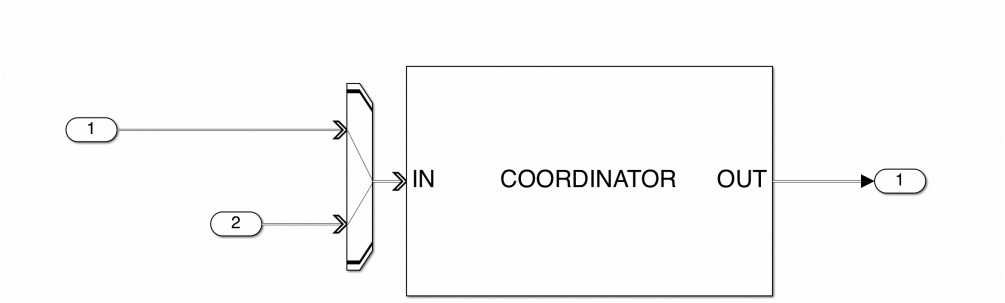

```
entity.FinalSpeed = 30;
entity.distance = 0;
[entity.ArrivalTime,...
    entity.ID] = setArrivalTime2();
%entity.Speed = 9 + 3 * rand();
entity.Speed = 20;
entity.TravelTime = 0;
entity.FuelConsumption = 0;
entity.Lane = 2;
entity.Position = 0;
entity.coe = [0, 0, entity.Speed, ...
    -entity.Speed * entity.ArrivalTime];
```

```
dest = [1, 3, 4];
idx = ceil(3 * rand());
entity.Destination = 3;%dest(idx);
% 0: go straight 1 : right turn 2: left turn
```

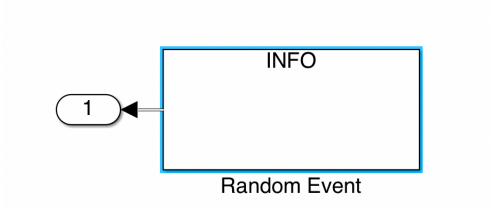
control zone子系统可以选择车辆的控制方式是OC还是CBF Control。他们分别有两个输入，一个是CAV传来的entity，一个是coordinator传来的entity。



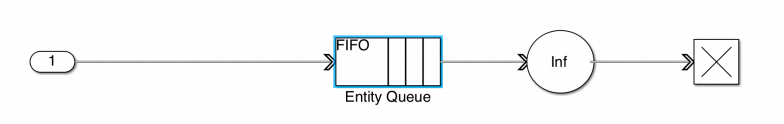

Coordinator的输入为controller的INFO entity和Random event的输入，输出为INFO entity（实际只有INFO一个输入）




Random event是一个entity generator，产生时间是dt = 1000




Merging point子系统为一个FIFO的队列，输入为CAV的信息，server为30/final_speed，即车辆最终的速度除以merging point处的长度，得到通行时间。车辆通过merging point后，entity被销毁。



## CBF control代码部分

作者定义了四个storage，都是queueFIFO。index 1代表control zone中的车辆实例队列，index 2代merging point之后的车辆实例队列，index 3代表地图区域外的车辆队列（还没有生成车辆），index 4信息输入的队列。

```
% Input queue for control zone #容量相同，都为100
storageSpec(1) = obj.queueFIFO('CAV', obj.capacity);
% Input queue after merging point
storageSpec(2) = obj.queueFIFO('CAV', obj.capacity);
% Input queue beyond mission space
storageSpec(3) = obj.queueFIFO('CAV', obj.capacity); %reserved
% Input queue for INFO
storageSpec(4) = obj.queueFIFO('INFO', obj.capacity);%reserved
```

对于storage的index为4的情况，当一个INFO实例进入，更新新到达车辆的ID。

当车辆到达时，将目前的entity传送到output port 2中，延时为0.

```
function [entity, events] = INFOEntryImpl(obj, storage, entity, tag)
  if storage == 4
    obj.newArrival = entity.data.VehicleID;
    events = obj.eventDestroy();
  end
end

function [entity, events] = INFOGenerateImpl(obj, storage, entity, tag)
  switch tag
  case 'arrival'
    entity.data.VehicleID = 0;
    events = obj.eventForward('output', 2, 0);
  end
end
```

当一个CAV实例进入后，根据storage的类型选择控制的方法。若storage index为1，那么其就在控制区域，采用CBF来控制。若storage index为2，那么其就通过了merging point，采用cruise方法，即保持速度不变。如果车辆还没进入地图，那么就采用defalt方法，什么也不做。

对于storage为1的情况，若两个lane的车同时到达控制区域，作者将其中一辆车后移0.1。

```
        function [entity, events] = CAVGenerateImpl(obj, storage, entity, tag)
            switch tag
                case 're-generate'
                    events = obj.eventForward('output', 1, 0);
            end
        end

        function [entity, events] = CAVEntryImpl(obj, storage, entity, ~)
            %Entry a vehicle
            events = [];
```

```
            if storage == 1 % apply CBF control
                obj.numVehicles = obj.numVehicles + 1;
                entity.data.distance = obj.distance;
                if(obj.two_events == 1) % two events at the same time, i.e., two CAVs
arrive at the same time
                    entity.data.distance = 0.1;
                    entity.data.Position = -0.1;
                end
                if entity.data.ID == 1
                    events = obj.eventTimer('CZ',obj.step);
                end
                obj.two_events = 1; % events count

            elseif storage == 2
                if entity.data.ID == 1
                    events = obj.eventTimer('ACZ',obj.step);
                end
            else
                if entity.data.ID == 1
                    events = obj.eventTimer('default',obj.step);
                end
            end
        end
```

作者接着在函数`function [entity, events, next] = CAVIterateImpl(obj, storage, entity, tag, status)`中对storage中的车辆信息进行迭代更新。

若对于其中一个entity用cbf来控制，则根据论文中的几种情况进行讨论：

若entity的ID为0，即队列中的第一辆车，则x1（前面的一辆车位于不同lane）的状态设为空，x0（当前车）的最后两个状态设为0。

```
x1 = []; %i_p vehicle state
obj.two_events = 0; % events reset, no two events
if(entity.data.ID == obj.firstID) % first CAV in the CZ queue
  x0 = [entity.data.Position,entity.data.Speed,entity.data.distance,0,0];

[entity.data.Position,entity.data.Speed,entity.data.distance,entity.data.Acceleration]
...
                                =
CBF_Control(x0,x1,obj.Vd,obj.Vmax,obj.Vmin,obj.Umax,obj.Umin,obj.step,1,1);
```

若车辆不是第一辆车，则区分前车是在同一个lane还是不同的lane

若在同一个lane，则x1仍然为空。

若在不同的lane，则x1的状态包括其位置、速度、加速度。

```
x0 =
[entity.data.Position,entity.data.Speed,entity.data.distance,obj.formerSpeed,obj.former
Acc];
if (entity.data.Lane == obj.formerLane) % front vehicle in the same lane

[entity.data.Position,entity.data.Speed,entity.data.distance,entity.data.Acceleration]
...
   = CBF_Control(x0,x1,obj.Vd,obj.Vmax,obj.Vmin,obj.Umax,obj.Umin,obj.step,1,2);
else  %front vehicle in the different lane
   x1 = []; % vehicle ip state
   if(entity.data.Lane == 1)  %main lane
    if(obj.main_exist == 1) % ip exists
      x1 = [obj.main_pos,obj.main_speed,obj.main_acc];
    end
   else %merging lane
    if(obj.merg_exist == 1)% ip exists
      x1 = [obj.merg_pos,obj.merg_speed,obj.merg_acc];
    end
   end
 [entity.data.Position,entity.data.Speed,entity.data.distance,entity.data.Acceleration]
...
 = CBF_Control(x0,x1,obj.Vd,obj.Vmax,obj.Vmin,obj.Umax,obj.Umin,obj.step,1,3);
end
```

计算好新的entity的位置、速度、距离、加速度后，进行参数更新

```
%update parameters
if(entity.data.Lane == 1) % update ip for the main lane queue
  obj.main_pos = entity.data.Position;
  obj.main_speed = entity.data.Speed;
  obj.main_acc = entity.data.Acceleration;
  obj.main_exist = 1;
else % update ip for the merging lane queue
  obj.merg_pos = entity.data.Position;
  obj.merg_speed = entity.data.Speed;
  obj.merg_acc = entity.data.Acceleration;
  obj.merg_exist = 1;
end
if(obj.numVehicles == entity.data.ID)
  obj.distance = entity.data.Position;  % the last CAV position is the distance of new
arrival CAV
  obj.main_exist = 0; % next loop, starting from the first CAV in the CZ queue, reset
  obj.merg_exist = 0;
end
```

在CBF函数中，根据传入参数type的不同，赋予不同的B(x)：

```
switch type
```

```
        case 1  % first vehicle
            A = [phi1 -1; 1 0; -1 0; A_vmax 0; A_vmin 0];
            b = [-phi0; ca*m*g; cd*m*g; b_vmax; b_vmin];
        case 2  % i-1 is in the same lane
            h = x0(3) - 1.8*x0(2) - 0.5*(v0 - x0(2))^2/(cd*g); %CBF for safety
            Bf = 1/h;
            LfBf = -(m*(v0 - x0(2)) - m*(v0 - x0(2))*dv0/(cd*g) - Fr*((v0 -
x0(2))/(cd*g) - 1.8))*Bf^2/m;
            LgBf = (1.8 - (v0 - x0(2))/(cd*g))*Bf^2/m;
            A = [phi1 -1; LgBf 0; 1 0; -1 0; A_vmax 0; A_vmin 0];
            b = [-phi0; -LfBf + 1/Bf; ca*m*g; cd*m*g; b_vmax; b_vmin];
        case 3  % i-1 is in the different lane
            cdg = cd*g;   %CBF for safe merging between i and i-1
            h = x0(3) - 0.5*(v0 - x0(2))^2/cdg - 1.8*x0(2)*(x0(1) + 0.5*(x0(2)^2 -
v0^2)/cdg)/L;
            Bf = 1/h;
            constant = x0(2) - v0 + 1.8*v0*x0(2)/L + Fr*((v0 - x0(2))/cdg -
1.8*v0*x0(2)/(cdg*L) - 1.8*x0(1)/L - (1.8*x0(2)^2 - 1.8*v0^2)/(cdg*L))/m;
            constant1 = ( (v0-x0(2))*L - 1.8*v0^2 )/(cdg*L);
            LfBf = (constant + constant1*dv0)*Bf^2;
            LgBf = Bf^2*(1.8*v0*x0(2) + 1.8*x0(1)*cdg + (1.8*x0(2)^2 - 1.8*v0^2) - (v0
- x0(2))*L)/(cdg*L*m);
            A_bf = [LgBf 0];
            b_bf = -LfBf+1/Bf;
            if(numel(y1)~= 0) % if ip vehicle exists
                h = y1(1) - x0(1) - 1.8*x0(2) - 0.5*(y1(2) - x0(2))^2/(cd*g); %CBF for
safety between i and ip
                Bf = 1/h;
                LfBf_ip = -(m*(y1(2) - x0(2)) - m*(y1(2) - x0(2))*y1(3)/(cd*g) - Fr*
((y1(2) - x0(2))/(cd*g) - 1.8))*Bf^2/m;
                LgBf_ip = (1.8 - (y1(2) - x0(2))/(cd*g))*Bf^2/m;
                b_ip = -LfBf_ip + 1/Bf;
                A_ip = [LgBf_ip, 0];
            else
                b_ip = [];
                A_ip = [];
            end
            A = [phi1 -1;A_bf;A_ip; 1 0; -1 0; A_vmax 0; A_vmin 0];
            b = [-phi0;b_bf; b_ip; ca*m*g; cd*m*g; b_vmax; b_vmin];
    end
    H = [2/(m^2) 0; 0 2*psc];
    F = [-2*Fr/(m^2); 0];
```

用qp来解上述非线性方程，得到控制输入u。

若没有解，则减速或者速度不变。

```
    options = optimoptions('quadprog',...
        'Algorithm','interior-point-convex','Display','off');
    [u,~,~,~,~] = quadprog(H,F,A,b,[],[],[],[],[],options);
    if (numel(u) == 0)  % if no solution found (usually happens in very heavy traffic, then use minimum deceleration or 0 deceleration)
       u = [-cd*m*g 0]; % minimum deceleration
    end
```

根据控制输入，用ode45求解微分方程，更新位置、速度和距离。

```
    a = (u(1) - Fr)/m;
    t=[0 dt];
    [~,xx]=ode45('acc',t,[x0, u(1), v0]);
    pos = xx(end, 1);
    vel = xx(end, 2);
    dis = xx(end, 3);
```

## coordinator部分

coordinator只负责接收环境和控制区域的信息并传送信息，不负责控制车辆。

虽然simulink中，coordinator有两个输入，但是作者只启用了一个输入，即INFO输入。所以实际上这个subsystem不接受任何环境信息，不起到作用。

```
        function [storageSpec, I, O] = getEntityStorageImpl(obj)
            % Input queue for entities
            storageSpec(1) = obj.queueFIFO('INFO', obj.capacity);
            I = 1;
            O = 1;
        end
```