LIVRABLE 1 - CONCEPTION

EN ÉQUIPE

Synchronisation de feux de circulation

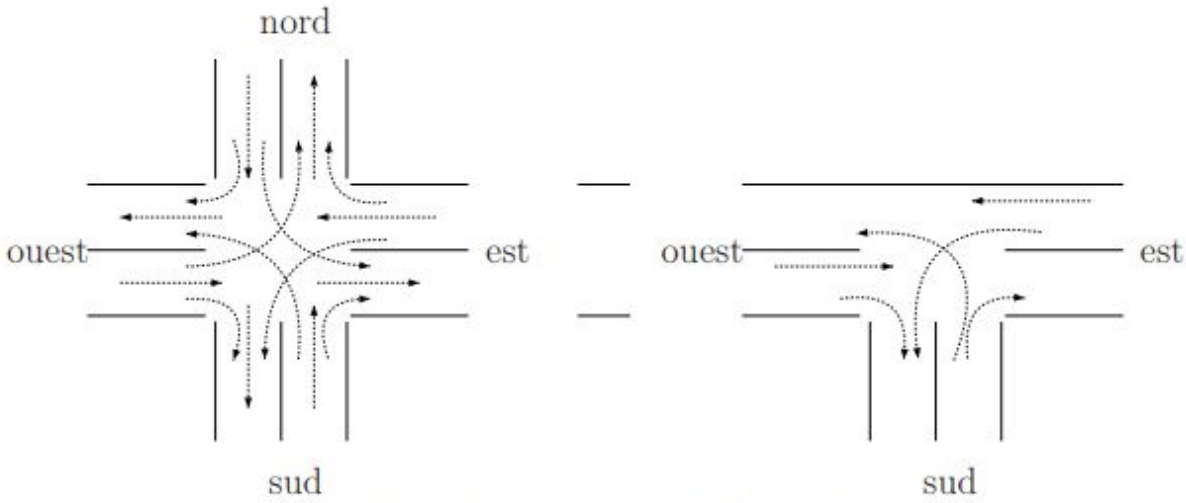


FIGURE 1 – Configuration de l'intersection

Auteurs

Élida Melo - 111 153 228

Ermine Wankpo - 111 067 301

Isabelle Eysseric - 111 189 609

Slim Ben Yahia - 111 150 230

LIVRABLE 1 - CONCEPTION

EN ÉQUIPE

Table des matières

LIVRABLE 1 - CONCEPTION

EN ÉQUIPE

1. Principaux événements, actions, interactions

Les principaux ensembles identifiés sont :

Intersection T

{ est.tourneGauche, sud.tourneGauche, est.continue, ouest.continue, sud.tourneDroite, ouest.tourneDroite }

{ traverse }

{ est.passeauvert, sud.passeauvert, ouest.passeauvert, est.passeaurouge, sud.passeaurouge, ouest.passeaurouge }

Intersection +

{ est2.tourneGauche, sud2.tourneGauche, ouest2.tourneGauche, nord2.tourneGauche, est2.continue, sud2.continue, ouest2.continue, nord2.continue, sud2.tourneDroite, sud2.tourneDroite, ouest2.tourneDroite, nord2.tourneDroite }

{ traverse2 }

{ est2.passeauvert, sud2.passeauvert, ouest2.passeauvert, nord2.passeauvert, est2.passeaurouge, sud2.passeaurouge, ouest2.passeaurouge, nord2.passeaurouge }

Les deux intersections

{ est.tourneGauche, sud.tourneGauche, est2.tourneGauche, sud2.tourneGauche, ouest2.tourneGauche, nord2.tourneGauche, est.continue, ouest.continue, est2.continue, sud2.continue, ouest2.continue, nord2.continue, sud.tourneDroite, ouest.tourneDroite, sud2.tourneDroite, sud2.tourneDroite, ouest2.tourneDroite, nord2.tourneDroite }

{ traverse,traverse2 }

{ est_sync.passeauvert, sud_sync.passeauvert, ouest_sync.passeauvert, nord_sync.passeauvert, est_sync.passeaurouge, sud_sync.passeaurouge, ouest_sync.passeaurouge, nord_sync.passeaurouge }

LIVRABLE 1 - CONCEPTION

2. Entités actives

Les processus qui sont des entités actives et implantés comme threads sont :

- Voitures
- Piétons
- Lumières

Pour ce faire nous utilisons les classes :

CarRunnable, LightRunnable, PedestrianRunnable, LightSyncRunnable, (Sync pour la version synchronisée).

3. Entités passives

Six entités passives partagées sont implantées comme moniteurs :

- TrafficController : Classe qui permet le contrôle du trafic des voitures, piétons et lumières en fonction de l'intersection choisie (T, Croix ou Syc).
- LightController : Classe qui permet les actions de changement de couleur des lumières.
- LightSyncController : Classe qui permet de synchroniser les lumières et les voitures des intersections en T et en Croix en utilisant des barrières. Pour la synchronisation des voitures nous avons utilisé l'approche producteur-consommateur où les producteurs représentent les actions quittant d'une intersection et menant à l'autre. Les consommateurs quant à eux, sont les des directions dans les actions dépendant de l'autre intersection.
- AllLightController : Classe qui permet le contrôle de toutes les lumières dans toutes les directions, incluses les méthodes de changement de couleur, vérification de lumières de la direction opposée, vérification s'il y a de lumières vertes pour empêcher le piéton de traverser, vérification des lumières dans les directions voisines pour ne pas permettre deux lumières adjacentes d'être vertes en même temps.

LIVRABLE 1 - CONCEPTION

- AllLightSyncController : c'est l'équivalent de la classe AllLightController, mais appliqué au cas des deux intersections synchronisées.
- PedestrianController : Classe permettant le contrôle du passage des piétons lors de l'exécution du thread PedestrianRunnable.

LIVRABLE 1 - CONCEPTION

4. Environnement interactif d'affichage

Une interface graphique sera utilisée pour l'affichage des lumières et des actions des voitures et des piétons, ainsi que pour l'interaction avec l'utilisateur. Les paramètres d'exécution tels que le type d'intersection choisie, le nombre de voitures et le nombre des piétons doivent être rentrés par l'utilisateur, bien que les fréquences des lumières pour chaque direction .

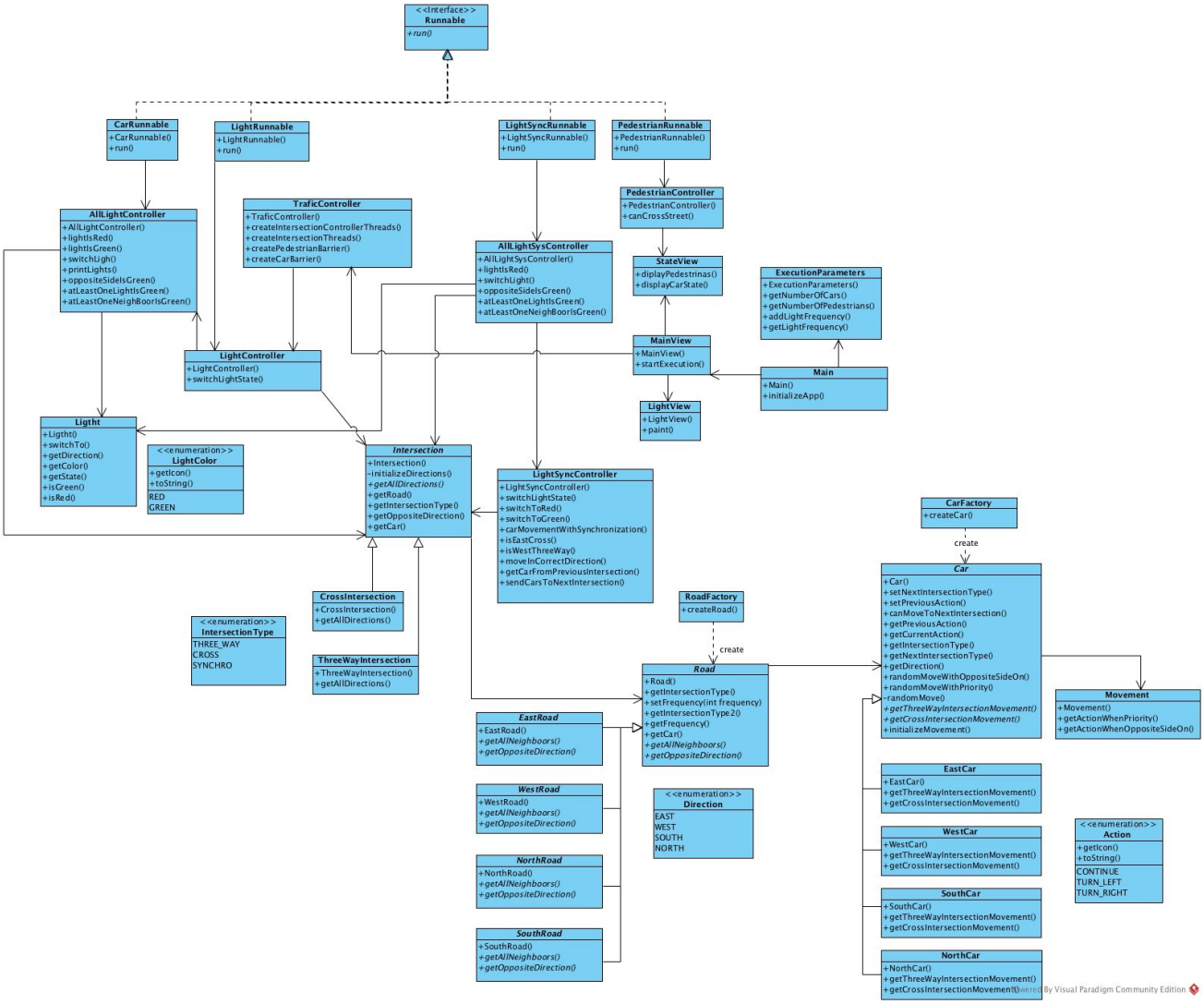
L'utilisateur a aussi l'option de mettre l'exécution en pause, ensuite continuer ou recommencer une nouvelle exécution ou quitter l'application. Pour ce faire les boutons PAUSE, RESUME, RESTART et QUIT sont disponibles dans la barre supérieure de l'interface.

Fig 1 - Interface graphique

LIVRABLE 1 - CONCEPTION

EN ÉQUIPE

5. Structure des classes



LIVRABLE 1 - CONCEPTION

Dans le diagramme de classe ci-dessus nous retrouvons les principales classes et ses associations :

- Intersection : classe abstraite utilisée pour instancier des intersections du type T et Croix.
- Light : classe utilisée pour représenter une lumière et ses actions de changement de couleur.
- Road : classe abstraite utilisée pour instancier des voix selon ses directions (est, ouest, sud, nord).
- Car : classe abstraite utilisée pour instancier une voiture selon sa direction (est, ouest, sud, nord).
- Les classes utilisées pour l'affichage et initialisation des threads : MainView, StateView et LightView.
- Les contrôleurs : TrafficController, LightController, AllLightController LightSyncController, AllLightSyncController et PedestrianController tel qu'indiqués dans la question 3.
- Les classes qui implémentent les threads : CarRunnable, LightRunnable, PedestrianRunnable, LightSyncRunnable, tel qu'indiquées dans la question 2.
- Autres : des classes auxiliaires tel que factories et énumérations.

6. Annexe

Partie 1 -CODE FSP

```
//----------------------------------Intersection T-----------------------------------------

set DirectionAlphabet = {est, ouest, sud}


// Q1-------------------------------------------------------------------------------------
VOITURE = ( {continue, tourneDroite, tourneGauche} -> VOITURE ).


// Q2-------------------------------------------------------------------------------------
||VOITURES = (DirectionAlphabet:VOITURE) \ {est.tourneDroite, sud.continue, ouest.tourneGauche}.


// Q3-------------------------------------------------------------------------------------
PIETONS = ( traverse -> PIETONS ).


// Q4-------------------------------------------------------------------------------------
LUMIERE = ( passeauvert -> passeaurouge -> LUMIERE ).

||LUMIERES = ( DirectionAlphabet:LUMIERE ).

// Controles qui empechent deux lumieres adjacentes vertes en meme temps:
CONTROLE_LUMIERES_SUD_EST = ( sud.passeauvert -> sud.passeaurouge -> CONTROLE_LUMIERES_SUD_EST
                        | est.passeauvert -> est.passeaurouge -> CONTROLE_LUMIERES_SUD_EST).

CONTROLE_LUMIERES_SUD_OUEST = ( sud.passeauvert -> sud.passeaurouge -> CONTROLE_LUMIERES_SUD_OUEST
                        | ouest.passeauvert -> ouest.passeaurouge -> CONTROLE_LUMIERES_SUD_OUEST).

// Controle qui va permettre aux lumieres opposees d'etre vertes en meme temps ou bien une verte et
// l'autre rouge, tout en respectant la condition d'adjacence :
||CONTROLE_LUMIERES = ( CONTROLE_LUMIERES_SUD_EST || CONTROLE_LUMIERES_SUD_OUEST ).
```

LIVRABLE 1 - CONCEPTION

```
// Q5-------------------------------------------------------------------------------------------

// Controle pour garantir qu'un pieton ne traverse que quand tous le lumieres sont rouges, c-a-d
// synchronisation sur l'action traverse pour chaque lumiere
CONTROLE_PIETONS_LUMIERE  = ( traverse -> CONTROLE_PIETONS_LUMIERE
                            | passeauvert -> passeaurouge -> CONTROLE_PIETONS_LUMIERE).

||CONTROLE_PASSAGEPIETONS = ( DirectionAlphabet: CONTROLE_PIETONS_LUMIERE)
                            /{traverse/{DirectionAlphabet}.traverse}.

// Controle pour garantir que les actions d'une voiture respectent les conditions de la lumiere
// (face est ou face ouest) en  prennent en compte l'etat de la lumiere opposee:
CONTROLE_VOITURE_EST_ROUGE_OUEST_ROUGE = ( est.passeauvert -> CONTROLE_VOITURE_EST_VERT_OUEST_ROUGE
                                         | ouest.passeauvert -> CONTROLE_VOITURE_EST_ROUGE_OUEST_VERT),
CONTROLE_VOITURE_EST_ROUGE_OUEST_VERT  = ( est.passeauvert -> CONTROLE_VOITURE_EST_VERT_OUEST_VERT
                                         | ouest.passeaurouge-> CONTROLE_VOITURE_EST_ROUGE_OUEST_ROUGE
                                         | ouest.{continue, tourneDroite} -> CONTROLE_VOITURE_EST_ROUGE_OUEST_VERT),
CONTROLE_VOITURE_EST_VERT_OUEST_ROUGE  = ( est.passeaurouge -> CONTROLE_VOITURE_EST_ROUGE_OUEST_ROUGE
                                         | ouest.passeauvert-> CONTROLE_VOITURE_EST_VERT_OUEST_VERT
                                         | est.{continue, tourneGauche} -> CONTROLE_VOITURE_EST_VERT_OUEST_ROUGE),
CONTROLE_VOITURE_EST_VERT_OUEST_VERT   = ( est.passeaurouge -> CONTROLE_VOITURE_EST_ROUGE_OUEST_VERT
                                         | ouest.passeaurouge-> CONTROLE_VOITURE_EST_VERT_OUEST_ROUGE
                                         | {est.continue, ouest.{continue, tourneDroite}}
                                           -> CONTROLE_VOITURE_EST_VERT_OUEST_VERT).

//Controle pour garantir que les actions d'une voiture respectent les conditions de la lumiere sud :
CONTROLE_VOITURE_SUD_ROUGE = ( sud.passeauvert -> CONTROLE_VOITURE_SUD_VERT),
CONTROLE_VOITURE_SUD_VERT  = ( sud.{tourneDroite, tourneGauche} -> CONTROLE_VOITURE_SUD_VERT
                             | sud.passeaurouge -> CONTROLE_VOITURE_SUD_ROUGE).

||CONTROLE_VOITURES = ( CONTROLE_VOITURE_EST_ROUGE_OUEST_ROUGE || CONTROLE_VOITURE_SUD_ROUGE).

//Processus qui controle le passage des voitures, des pietons et le changement des lumieres :
||CONTROLE_INTERSECTION = ( CONTROLE_PASSAGEPIETONS  || CONTROLE_VOITURES || CONTROLE_LUMIERES).

||INTERSECTION = (LUMIERES || PIETONS || VOITURES || CONTROLE_INTERSECTION).
```

LIVRABLE 1 - CONCEPTION

```
// Q6-----------------------------------------------------------------------------------------------------------------

const False = 0
const True = 1
range Bool = False..True

property TRAVERSE_LUMIERES_ROUGE = TRAVERSE_LUMIERES_ROUGE_PARAMETRE[False][False][False],
TRAVERSE_LUMIERES_ROUGE_PARAMETRE[estvert:Bool][sudvert:Bool][ouestvert:Bool] =
        ( when(estvert==False && sudvert==False && ouestvert==False) traverse
          -> TRAVERSE_LUMIERES_ROUGE_PARAMETRE[estvert][sudvert][ouestvert]
          // l'action traverse se fait seulement dans cette condition

        | when(estvert==False) est.passeauvert  -> TRAVERSE_LUMIERES_ROUGE_PARAMETRE[True][sudvert][ouestvert]
        | when(estvert==True) est.passeaurouge -> TRAVERSE_LUMIERES_ROUGE_PARAMETRE[False][sudvert][ouestvert]
        | when(sudvert==False) sud.passeauvert  -> TRAVERSE_LUMIERES_ROUGE_PARAMETRE[estvert][True][ouestvert]
        | when(sudvert==True) sud.passeaurouge -> TRAVERSE_LUMIERES_ROUGE_PARAMETRE[estvert][False][ouestvert]
        | when(ouestvert==False) ouest.passeauvert  -> TRAVERSE_LUMIERES_ROUGE_PARAMETRE[estvert][sudvert][True]
        | when(ouestvert==True) ouest.passeaurouge -> TRAVERSE_LUMIERES_ROUGE_PARAMETRE[estvert][sudvert][False]).

||TEST1 = ( INTERSECTION || TRAVERSE_LUMIERES_ROUGE ).



// Q8-----------------------------------------------------------------------------------------------------------------

// Oui, il faut spécifier des singletons car on veut veérifier que chacune des actions peut etre executée dans
// toutes les composantes fortement connexes. Si on spécifie par exemple :
// progress EST_TOURNE = {est.tourneDroite, est.tourneGauche}, on va vérifier qu'au moins une des deux actions
// peut etreexecutée dans toute les composantes fortemenet connexes et ce n'est pas ce qu'on veut faire

//progress PIETONS_TRAVERSE = {traverse}
//progress EST_CONTINUE = {est.continue}
//progress EST_TOURNE_GAUCHE = {est.tourneGauche}
//progress SUD_TOURNE_DROITE = {sud.tourneDroite}
//progress SUD_TOURNE_GAUCHE = {sud.tourneGauche}
//progress OUEST_CONTINUE = {ouest.continue}
//progress OUEST_TOURNE_DROITE = {ouest.tourneDroite}


// Q7-----------------------------------------------------------------------------------------------------------------

// Proprietes pour verifier l'absence de collision selon l'etat de la lumiere, en permettant seulement les
// voitures de realiser les actions qui sont en accord avec cet etat et aussi en prenant compte les lumieres
// adjacentes et face opposee

property ABSENCE_COLLISION_EST = ABSENCE_COLLISION_EST_PARAMETRE[False][False][False],
ABSENCE_COLLISION_EST_PARAMETRE[estvert:Bool][sudvert:Bool][ouestvert:Bool] =
        ( when(estvert==True && sudvert==False && ouestvert==False) est.tourneGauche
          -> ABSENCE_COLLISION_EST_PARAMETRE[estvert][sudvert][ouestvert]
        | when(estvert==True && sudvert==False) est.continue -> ABSENCE_COLLISION_EST_PARAMETRE[estvert][sudvert][ouestvert]
        | when(estvert==False) est.passeauvert -> ABSENCE_COLLISION_EST_PARAMETRE[True][sudvert][ouestvert]
        | when(estvert==True) est.passeaurouge -> ABSENCE_COLLISION_EST_PARAMETRE[False][sudvert][ouestvert]
        | when(sudvert==False) sud.passeauvert -> ABSENCE_COLLISION_EST_PARAMETRE[estvert][True][ouestvert]
        | when(sudvert==True) sud.passeaurouge -> ABSENCE_COLLISION_EST_PARAMETRE[estvert][False][ouestvert]
        | when(ouestvert==False) ouest.passeauvert -> ABSENCE_COLLISION_EST_PARAMETRE[estvert][sudvert][True]
        | when(ouestvert==True) ouest.passeaurouge -> ABSENCE_COLLISION_EST_PARAMETRE[estvert][sudvert][False]).

property ABSENCE_COLLISION_SUD = ABSENCE_COLLISION_SUD_PARAMETRE[False][False][False],
ABSENCE_COLLISION_SUD_PARAMETRE[estvert:Bool][sudvert:Bool][ouestvert:Bool] =
        ( when(estvert==False && sudvert==True && ouestvert==False) sud.{tourneDroite, tourneGauche}
          -> ABSENCE_COLLISION_SUD_PARAMETRE[estvert][sudvert][ouestvert]
        | when(estvert==False) est.passeauvert -> ABSENCE_COLLISION_SUD_PARAMETRE[True][sudvert][ouestvert]
        | when(estvert==True) est.passeaurouge -> ABSENCE_COLLISION_SUD_PARAMETRE[False][sudvert][ouestvert]
        | when(sudvert==False) sud.passeauvert -> ABSENCE_COLLISION_SUD_PARAMETRE[estvert][True][ouestvert]
        | when(sudvert==True) sud.passeaurouge -> ABSENCE_COLLISION_SUD_PARAMETRE[estvert][False][ouestvert]
        | when(ouestvert==False) ouest.passeauvert -> ABSENCE_COLLISION_SUD_PARAMETRE[estvert][sudvert][True]
        | when(ouestvert==True) ouest.passeaurouge -> ABSENCE_COLLISION_SUD_PARAMETRE[estvert][sudvert][False]).

property ABSENCE_COLLISION_OUEST = ABSENCE_COLLISION_OUEST_PARAMETRE[False][False],
ABSENCE_COLLISION_OUEST_PARAMETRE[sudvert:Bool][ouestvert:Bool] =
        ( when(sudvert==False  && ouestvert==True) ouest.{continue, tourneDroite}
          -> ABSENCE_COLLISION_OUEST_PARAMETRE[sudvert][ouestvert]
        | when(sudvert==False) sud.passeauvert -> ABSENCE_COLLISION_OUEST_PARAMETRE[True][ouestvert]
        | when(sudvert==True) sud.passeaurouge -> ABSENCE_COLLISION_OUEST_PARAMETRE[False][ouestvert]
        | when(ouestvert==False) ouest.passeauvert -> ABSENCE_COLLISION_OUEST_PARAMETRE[sudvert][True]
        | when(ouestvert==True) ouest.passeaurouge -> ABSENCE_COLLISION_OUEST_PARAMETRE[sudvert][False]).

||TEST2 = ( INTERSECTION || ABSENCE_COLLISION_EST || ABSENCE_COLLISION_SUD || ABSENCE_COLLISION_OUEST).
```

LIVRABLE 1 - CONCEPTION

```
// Q9-------------------------------------------------------------------------------------------------------------------

//Dans cette question vous trouverez les versions des questions precedentes incluant la lumiere Nord2

||VOITURES_2 = ( DirectionAlphabet2:VOITURE ).

PIETONS_2 = ( traverse2 -> PIETONS_2 ).

||LUMIERES_2 = ( DirectionAlphabet2:LUMIERE ).

CONTROLE_LUMIERES_SUD_EST_2   =  ( sud2.passeauvert -> sud2.passeaurouge -> CONTROLE_LUMIERES_SUD_EST_2
                                 | est2.passeauvert -> est2.passeaurouge -> CONTROLE_LUMIERES_SUD_EST_2).

CONTROLE_LUMIERES_SUD_OUEST_2 =  ( sud2.passeauvert   -> sud2.passeaurouge   -> CONTROLE_LUMIERES_SUD_OUEST_2
                                 | ouest2.passeauvert -> ouest2.passeaurouge -> CONTROLE_LUMIERES_SUD_OUEST_2).

CONTROLE_LUMIERES_NORD_EST_2  =  ( nord2.passeauvert -> nord2.passeaurouge -> CONTROLE_LUMIERES_NORD_EST_2
                                 | est2.passeauvert  -> est2.passeaurouge  -> CONTROLE_LUMIERES_NORD_EST_2).

CONTROLE_LUMIERES_NORD_OUEST_2 = ( nord2.passeauvert  -> nord2.passeaurouge  -> CONTROLE_LUMIERES_NORD_OUEST_2
                                 | ouest2.passeauvert -> ouest2.passeaurouge -> CONTROLE_LUMIERES_NORD_OUEST_2).

||CONTROLE_LUMIERES_2 = ( CONTROLE_LUMIERES_SUD_EST_2  || CONTROLE_LUMIERES_SUD_OUEST_2
                       || CONTROLE_LUMIERES_NORD_EST_2 || CONTROLE_LUMIERES_NORD_OUEST_2).


CONTROLE_PIETONS_LUMIERE2   = ( traverse2 -> CONTROLE_PIETONS_LUMIERE2
                              | passeauvert -> passeaurouge -> CONTROLE_PIETONS_LUMIERE2).

||CONTROLE_PASSAGEPIETONS_2 = ( DirectionAlphabet2: CONTROLE_PIETONS_LUMIERE2)
                             /{traverse2/{DirectionAlphabet2}.traverse2}.


CONTROLE_VOITURE_EST_ROUGE_OUEST_ROUGE_2 = ( est2.passeauvert   -> CONTROLE_VOITURE_EST_VERT_OUEST_ROUGE_2
                                           | ouest2.passeauvert -> CONTROLE_VOITURE_EST_ROUGE_OUEST_VERT_2),
CONTROLE_VOITURE_EST_ROUGE_OUEST_VERT_2  = ( est2.passeauvert   -> CONTROLE_VOITURE_EST_VERT_OUEST_VERT_2
                                           | ouest2.passeaurouge-> CONTROLE_VOITURE_EST_ROUGE_OUEST_ROUGE_2
                                           | ouest2.{continue, tourneDroite, tourneGauche} -> CONTROLE_VOITURE_EST_ROUGE_OUEST_VERT_2),
CONTROLE_VOITURE_EST_VERT_OUEST_ROUGE_2  = ( est2.passeaurouge  -> CONTROLE_VOITURE_EST_ROUGE_OUEST_ROUGE_2
                                           | ouest2.passeauvert -> CONTROLE_VOITURE_EST_VERT_OUEST_VERT_2
                                           | est2.{continue, tourneDroite, tourneGauche} -> CONTROLE_VOITURE_EST_VERT_OUEST_ROUGE_2),
CONTROLE_VOITURE_EST_VERT_OUEST_VERT_2   = ( est2.passeaurouge  -> CONTROLE_VOITURE_EST_ROUGE_OUEST_VERT_2
                                           | ouest2.passeaurouge-> CONTROLE_VOITURE_EST_VERT_OUEST_ROUGE_2
                                           | {est2.{continue, tourneDroite}, ouest2.{continue, tourneDroite}}
                                             -> CONTROLE_VOITURE_EST_VERT_OUEST_VERT_2).

CONTROLE_VOITURE_NORD_ROUGE_SUD_ROUGE_2  = ( nord2.passeauvert -> CONTROLE_VOITURE_NORD_VERT_SUD_ROUGE_2
                                           | sud2.passeauvert -> CONTROLE_VOITURE_NORD_ROUGE_SUD_VERT_2),
CONTROLE_VOITURE_NORD_ROUGE_SUD_VERT_2   = ( nord2.passeauvert -> CONTROLE_VOITURE_NORD_VERT_SUD_VERT_2
                                           | sud2.passeaurouge-> CONTROLE_VOITURE_NORD_ROUGE_SUD_ROUGE_2
                                           | sud2.{continue, tourneDroite, tourneGauche} -> CONTROLE_VOITURE_NORD_ROUGE_SUD_VERT_2),
CONTROLE_VOITURE_NORD_VERT_SUD_ROUGE_2   = ( nord2.passeaurouge -> CONTROLE_VOITURE_NORD_ROUGE_SUD_ROUGE_2
                                           | sud2.passeauvert-> CONTROLE_VOITURE_NORD_VERT_SUD_VERT_2
                                           | nord2.{continue, tourneDroite, tourneGauche} -> CONTROLE_VOITURE_NORD_VERT_SUD_ROUGE_2),
CONTROLE_VOITURE_NORD_VERT_SUD_VERT_2    = ( nord2.passeaurouge -> CONTROLE_VOITURE_NORD_ROUGE_SUD_VERT_2
                                           | sud2.passeaurouge-> CONTROLE_VOITURE_NORD_VERT_SUD_ROUGE_2
                                           | {nord2.{continue, tourneDroite}, sud2.{continue, tourneDroite}}
                                             -> CONTROLE_VOITURE_NORD_VERT_SUD_VERT_2).

||CONTROLE_VOITURES_2 = ( CONTROLE_VOITURE_EST_ROUGE_OUEST_ROUGE_2 || CONTROLE_VOITURE_NORD_ROUGE_SUD_ROUGE_2).

||CONTROLE_INTERSECTION_2 = ( CONTROLE_PASSAGEPIETONS_2  || CONTROLE_VOITURES_2 || CONTROLE_LUMIERES_2).

||INTERSECTION_2 = (LUMIERES_2 || PIETONS_2 || VOITURES_2 || CONTROLE_INTERSECTION_2).
```

LIVRABLE 1 - CONCEPTION

EN ÉQUIPE

```
property TRAVERSE_LUMIERES_ROUGE_2 = TRAVERSE_LUMIERES_ROUGE_PARAMETRE_2[False][False][False][False],
TRAVERSE_LUMIERES_ROUGE_PARAMETRE_2[estvert:Bool][sudvert:Bool][ouestvert:Bool][nordvert:Bool] =
        ( when(estvert==False && sudvert==False && ouestvert==False && nordvert==False) traverse2
          -> TRAVERSE_LUMIERES_ROUGE_PARAMETRE_2[estvert][sudvert][ouestvert][nordvert]
        | when(estvert==False) est2.passeauvert  -> TRAVERSE_LUMIERES_ROUGE_PARAMETRE_2[True][sudvert][ouestvert][nordvert]
        | when(estvert==True) est2.passeaurouge -> TRAVERSE_LUMIERES_ROUGE_PARAMETRE_2[False][sudvert][ouestvert][nordvert]
        | when(sudvert==False) sud2.passeauvert  -> TRAVERSE_LUMIERES_ROUGE_PARAMETRE_2[estvert][True][ouestvert][nordvert]
        | when(sudvert==True) sud2.passeaurouge -> TRAVERSE_LUMIERES_ROUGE_PARAMETRE_2[estvert][False][ouestvert][nordvert]
        | when(ouestvert==False) ouest2.passeauvert  -> TRAVERSE_LUMIERES_ROUGE_PARAMETRE_2[estvert][sudvert][True][nordvert]
        | when(ouestvert==True) ouest2.passeaurouge -> TRAVERSE_LUMIERES_ROUGE_PARAMETRE_2[estvert][sudvert][False][nordvert]
        | when(nordvert==False) nord2.passeauvert  -> TRAVERSE_LUMIERES_ROUGE_PARAMETRE_2[estvert][sudvert][ouestvert][True]
        | when(nordvert==True) nord2.passeaurouge -> TRAVERSE_LUMIERES_ROUGE_PARAMETRE_2[estvert][sudvert][ouestvert][False]).

||TEST3 = ( INTERSECTION_2 || TRAVERSE_LUMIERES_ROUGE_2 ).

progress PIETONS_TRAVERSE_2    = {traverse2}
progress EST_CONTINUE_2        = {est2.continue}
progress EST_TOURNE_DROITE_2   = {est2.tourneDroite}
progress EST_TOURNE_GAUCHE_2   = {est2.tourneGauche}
progress OUEST_CONTINUE_2      = {ouest2.continue}
progress OUEST_TOURNE_DROITE_2 = {ouest2.tourneDroite}
progress OUEST_TOURNE_GAUCHE_2 = {ouest2.tourneGauche}
progress SUD_CONTINUE_2        = {sud2.continue}
progress SUD_TOURNE_DROITE_2   = {sud2.tourneDroite}
progress SUD_TOURNE_GAUCHE_2   = {sud2.tourneGauche}
progress NORD_CONTINUE_2       = {nord2.continue}
progress NORD_TOURNE_DROITE_2  = {nord2.tourneDroite}
progress NORD_TOURNE_GAUCHE_2  = {nord2.tourneGauche}

property ABSENCE_COLLISION_EST_2 = ABSENCE_COLLISION_EST_PARAMETRE_2[False][False][False][False],
ABSENCE_COLLISION_EST_PARAMETRE_2[estvert:Bool][sudvert:Bool][ouestvert:Bool][nordvert:Bool] =
        ( when(estvert==True && sudvert==False && ouestvert==False && nordvert==False) est2.tourneGauche
          -> ABSENCE_COLLISION_EST_PARAMETRE_2[estvert][sudvert][ouestvert][nordvert]
        | when(estvert==True && sudvert==False && nordvert==False) est2.{continue, tourneDroite}
          -> ABSENCE_COLLISION_EST_PARAMETRE_2[estvert][sudvert][ouestvert][nordvert]
        | when(estvert==False) est2.passeauvert  -> ABSENCE_COLLISION_EST_PARAMETRE_2[True][sudvert][ouestvert][nordvert]
        | when(estvert==True) est2.passeaurouge  -> ABSENCE_COLLISION_EST_PARAMETRE_2[False][sudvert][ouestvert][nordvert]
        | when(sudvert==False) sud2.passeauvert  -> ABSENCE_COLLISION_EST_PARAMETRE_2[estvert][True][ouestvert][nordvert]
        | when(sudvert==True) sud2.passeaurouge  -> ABSENCE_COLLISION_EST_PARAMETRE_2[estvert][False][ouestvert][nordvert]
        | when(ouestvert==False) ouest2.passeauvert  -> ABSENCE_COLLISION_EST_PARAMETRE_2[estvert][sudvert][True][nordvert]
        | when(ouestvert==True) ouest2.passeaurouge  -> ABSENCE_COLLISION_EST_PARAMETRE_2[estvert][sudvert][False][nordvert]
        | when(nordvert==False) nord2.passeauvert    -> ABSENCE_COLLISION_EST_PARAMETRE_2[estvert][sudvert][ouestvert][True]
        | when(nordvert==True) nord2.passeaurouge    -> ABSENCE_COLLISION_EST_PARAMETRE_2[estvert][sudvert][ouestvert][False]).

property ABSENCE_COLLISION_OUEST_2 = ABSENCE_COLLISION_OUEST_PARAMETRE_2[False][False][False][False],
ABSENCE_COLLISION_OUEST_PARAMETRE_2[estvert:Bool][sudvert:Bool][ouestvert:Bool][nordvert:Bool] =
        ( when(estvert==False && sudvert==False && ouestvert==True && nordvert==False) ouest2.tourneGauche
          -> ABSENCE_COLLISION_OUEST_PARAMETRE_2[estvert][sudvert][ouestvert][nordvert]
        | when(sudvert==False && ouestvert==True && nordvert==False) ouest2.{continue, tourneDroite}
          -> ABSENCE_COLLISION_OUEST_PARAMETRE_2[estvert][sudvert][ouestvert][nordvert]
        | when(estvert==False) est2.passeauvert  -> ABSENCE_COLLISION_OUEST_PARAMETRE_2[True][sudvert][ouestvert][nordvert]
        | when(estvert==True) est2.passeaurouge  -> ABSENCE_COLLISION_OUEST_PARAMETRE_2[False][sudvert][ouestvert][nordvert]
        | when(sudvert==False) sud2.passeauvert  -> ABSENCE_COLLISION_OUEST_PARAMETRE_2[estvert][True][ouestvert][nordvert]
        | when(sudvert==True) sud2.passeaurouge  -> ABSENCE_COLLISION_OUEST_PARAMETRE_2[estvert][False][ouestvert][nordvert]
        | when(ouestvert==False) ouest2.passeauvert -> ABSENCE_COLLISION_OUEST_PARAMETRE_2[estvert][sudvert][True][nordvert]
        | when(ouestvert==True) ouest2.passeaurouge -> ABSENCE_COLLISION_OUEST_PARAMETRE_2[estvert][sudvert][False][nordvert]
        | when(nordvert==False) nord2.passeauvert   -> ABSENCE_COLLISION_OUEST_PARAMETRE_2[estvert][sudvert][ouestvert][True]
        | when(nordvert==True) nord2.passeaurouge   -> ABSENCE_COLLISION_OUEST_PARAMETRE_2[estvert][sudvert][ouestvert][False]).
```

LIVRABLE 1 - CONCEPTION

```
property ABSENCE_COLLISION_SUD_2 = ABSENCE_COLLISION_SUD_PARAMETRE_2[False][False][False][False],
ABSENCE_COLLISION_SUD_PARAMETRE_2[estvert:Bool][sudvert:Bool][ouestvert:Bool][nordvert:Bool] =
        ( when(estvert==False && sudvert==True && ouestvert==False && nordvert==False) sud2.tourneGauche
          -> ABSENCE_COLLISION_SUD_PARAMETRE_2[estvert][sudvert][ouestvert][nordvert]
        | when(estvert==False &&  sudvert==True && ouestvert==False) sud2.{continue, tourneDroite}
          -> ABSENCE_COLLISION_SUD_PARAMETRE_2[estvert][sudvert][ouestvert][nordvert]
        | when(estvert==False) est2.passeauvert -> ABSENCE_COLLISION_SUD_PARAMETRE_2[True][sudvert][ouestvert][nordvert]
        | when(estvert==True) est2.passeaurouge -> ABSENCE_COLLISION_SUD_PARAMETRE_2[False][sudvert][ouestvert][nordvert]
        | when(sudvert==False) sud2.passeauvert -> ABSENCE_COLLISION_SUD_PARAMETRE_2[estvert][True][ouestvert][nordvert]
        | when(sudvert==True) sud2.passeaurouge -> ABSENCE_COLLISION_SUD_PARAMETRE_2[estvert][False][ouestvert][nordvert]
        | when(ouestvert==False) ouest2.passeauvert -> ABSENCE_COLLISION_SUD_PARAMETRE_2[estvert][sudvert][True][nordvert]
        | when(ouestvert==True) ouest2.passeaurouge -> ABSENCE_COLLISION_SUD_PARAMETRE_2[estvert][sudvert][False][nordvert]
        | when(nordvert==False) nord2.passeauvert   -> ABSENCE_COLLISION_SUD_PARAMETRE_2[estvert][sudvert][ouestvert][True]
        | when(nordvert==True) nord2.passeaurouge   -> ABSENCE_COLLISION_SUD_PARAMETRE_2[estvert][sudvert][ouestvert][False]).

property ABSENCE_COLLISION_NORD_2 = ABSENCE_COLLISION_NORD_PARAMETRE_2[False][False][False][False],
ABSENCE_COLLISION_NORD_PARAMETRE_2[estvert:Bool][sudvert:Bool][ouestvert:Bool][nordvert:Bool] =
        ( when(estvert==False && sudvert==False && ouestvert==False && nordvert==True) nord2.tourneGauche
          -> ABSENCE_COLLISION_NORD_PARAMETRE_2[estvert][sudvert][ouestvert][nordvert]
        | when(estvert==False && ouestvert==False && nordvert==True) nord2.{continue, tourneDroite}
          -> ABSENCE_COLLISION_NORD_PARAMETRE_2[estvert][sudvert][ouestvert][nordvert]
        | when(estvert==False) est2.passeauvert -> ABSENCE_COLLISION_NORD_PARAMETRE_2[True][sudvert][ouestvert][nordvert]
        | when(estvert==True) est2.passeaurouge -> ABSENCE_COLLISION_NORD_PARAMETRE_2[False][sudvert][ouestvert][nordvert]
        | when(sudvert==False) sud2.passeauvert -> ABSENCE_COLLISION_NORD_PARAMETRE_2[estvert][True][ouestvert][nordvert]
        | when(sudvert==True) sud2.passeaurouge -> ABSENCE_COLLISION_NORD_PARAMETRE_2[estvert][False][ouestvert][nordvert]
        | when(ouestvert==False) ouest2.passeauvert -> ABSENCE_COLLISION_NORD_PARAMETRE_2[estvert][sudvert][True][nordvert]
        | when(ouestvert==True) ouest2.passeaurouge -> ABSENCE_COLLISION_NORD_PARAMETRE_2[estvert][sudvert][False][nordvert]
        | when(nordvert==False) nord2.passeauvert   -> ABSENCE_COLLISION_NORD_PARAMETRE_2[estvert][sudvert][ouestvert][True]
        | when(nordvert==True) nord2.passeaurouge   -> ABSENCE_COLLISION_NORD_PARAMETRE_2[estvert][sudvert][ouestvert][False]).

||TEST4 = ( INTERSECTION_2 || ABSENCE_COLLISION_EST_2 || ABSENCE_COLLISION_OUEST_2
         || ABSENCE_COLLISION_SUD_2 || ABSENCE_COLLISION_NORD_2).

// Q10-------------------------------------------------------------------------------------------------------------

// Processus VOITURE_SYNC tel quel suggere dans l'enonce

VOITURE_SYNC = ({approcheC, approcheG, approcheD} -> VSYNC ),
VSYNC = (continue -> finitC -> VOITURE_SYNC |
tourneGauche -> finitG -> VOITURE_SYNC |
tourneDroite -> finitD -> VOITURE_SYNC |
attend -> VSYNC ).

|| VOITURES_SYNC1 = DirectionAlphabet:VOITURE_SYNC.
|| VOITURES_SYNC2 = DirectionAlphabet2:VOITURE_SYNC.
||VOITURES_SYNC = (VOITURES_SYNC1 || VOITURES_SYNC2).


// Parallelisation des intersections, lumieres, pietons, les controles de traverse en lumiere rouge,
// prorietes de securite et renommage sur les actions passeauvert et passeaurouge de chaque direction
// pour la sychronisation sur ces actions

||SYNC_INTERSECTION = (TEST1 || TEST2 || TEST3 || TEST4) /{
                            est_sync.passeauvert/{est, est2}.passeauvert,
                            est_sync.passeaurouge/{est, est2}.passeaurouge,
                            ouest_sync.passeauvert/{ouest, ouest2}.passeauvert,
                            ouest_sync.passeaurouge/{ouest, ouest2}.passeaurouge,
                            sud_sync.passeauvert/{sud, sud2}.passeauvert,
                            sud_sync.passeaurouge/{sud, sud2}.passeaurouge,
                            nord_sync.passeauvert/nord2.passeauvert,
                            nord_sync.passeaurouge/nord2.passeaurouge
                            }.

||TEST5 = (VOITURES_SYNC || SYNC_INTERSECTION) / {
                            est2.approcheC/est.finitC,
                            est2.approcheG/sud.finitG,
                            ouest.approcheC/ouest2.finitC,
                            ouest.approcheD/sud2.finitD,
                            ouest.approcheG/nord2.finitG
                            }.
```