Université Bordeaux I
U.F.R. Mathématiques et Informatique

Master Informatique 2^{eme} année
Année 2003-2004

Lundi 26 janvier 2004, 8 heures - 9 heures 30
Tous documents autorisés.

Conceptions Formelles (AltaRica)
Alain Griffault

SUJET + CORRIGE

Le système à contrôler

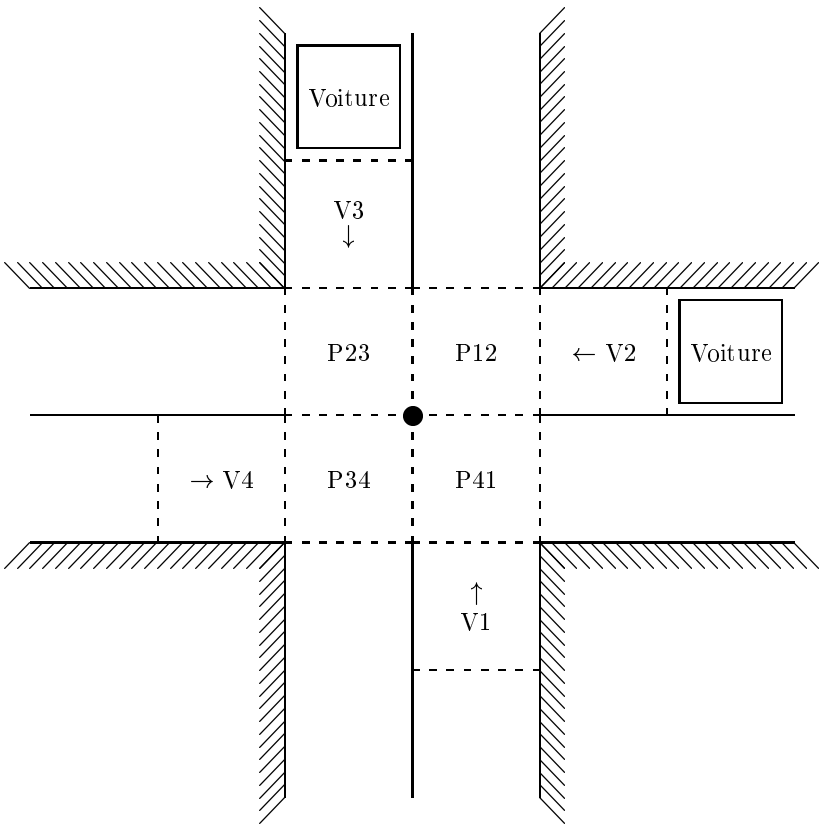

FIG. 1 – Un carrefour routier

Le système à concevoir est le contrôle de la circulation dans un carrefour tel que celui de la figure 1.Les voitures y pénètrent pour soit tourner à droite, soit continuer tout droit, soit tourner à gauche. Pour simplifier le problème, les hypothèses suivantes seront considérées valides :

- Deux véhicules ne peuvent pas être en même temps sur une des huit places *en pointillée*.
- Les routes en sortie de carrefour sont toujours libres.
- Les véhicules avancent par *bonds*. Lorsqu'un véhicule est sur une place, si la place voisine convoitée est libre, alors à l'instant suivant, si le véhicule a bougé, alors la place initiale est libre et la place convoitée est occupée.
- Les véhicules n'ont pas de marche arrière.
- En provenance de la voie $V(i)$, un véhicule doit :
  - pour tourner à droite, obtenir la place $P(i-1,i)$,
  - pour continuer tout droit, obtenir successivement la place $P(i-1,i)$ puis la place $P(i,i+1)$,
  - pour tourner à gauche, obtenir successivement la place $P(i-1,i)$ puis la place $P(i,i+1)$ puis la place $P(i+1,i+2)$.

Les propriétés attendues du système

Le contrôle devra vérifier les propriétés suivantes :

Propriété 1 *Les véhicules ne doivent jamais se gêner mutuellement de telle sorte que plus aucune d'entre elles ne puisse avancer.*

Propriété 2 *Pour des raisons de sûreté, il est interdit aux véhicules d'effectuer une marche arrière.*

Propriété 3 *Pour des raisons de sûreté, le carrefour ne doit pas être considéré comme un rond-point. Il est donc interdit aux véhicules d'utiliser le carrefour pour faire demi-tour.*

Travail à réaliser

Les six questions sont réparties dans le document joint. Vous devez répondre sur le document.
- La première question porte sur la compréhension du modèle initial.
- La deuxième question consiste à écrire les spécifications des propriétés 2 et 3.
- Les troisième, quatrième et cinquième questions portent sur la conception de modèle AltaRica.
- La sixième question porte sur la compréhension de votre modèle.

Chapitre 1

Les composants de base

1.1 Le nœud Voie

Un composant `Voie` permet de modéliser la présence d'un véhicule dans la voie, puis son entrée dans le carrefour avec l'indication de la direction souhaitée.

1.1.1 Le source AltaRica

```
node Voie
  state
    libre : bool : public;
  event
    entrer, droite, gauche, enFace;
  trans
    libre  |- entrer -> libre := false;
    ~libre |- droite, gauche, enFace -> libre := true;
  init
    libre := true;
edon
```

1.1.2 La sémantique

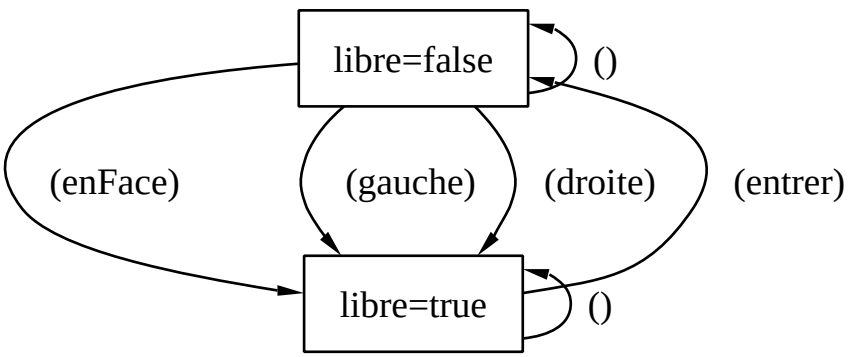

FIG. 1.1 – Le nœud Voie

1.2 Le nœud Place

Un composant `Place` modélise la présence d'un véhicule dans le carrefour et mémorise sa direction souhaitée. La variable de flux `libre` permet d'indiquer à l'environnement si la place est occupée ou non.

1.2.1 Le source AltaRica

```
node Place
  flow
    libre : bool;
  state
    droite, gauche, enFace : bool : public;
  event
    inDroite, inGauche, inEnFace,
    outDroite, outGauche, outEnFace;
  trans
    libre  |- inDroite -> droite := true;
    libre  |- inGauche -> gauche := true;
    libre  |- inEnFace -> enFace := true;
    droite |- outDroite -> droite := false;
    gauche |- outGauche -> gauche := false;
    enFace |- outEnFace -> enFace := false;
  assert
    libre = ~(droite | gauche | enFace);
  init
    droite := false, gauche := false, enFace := false;
edon
```

1.2.2 La sémantique

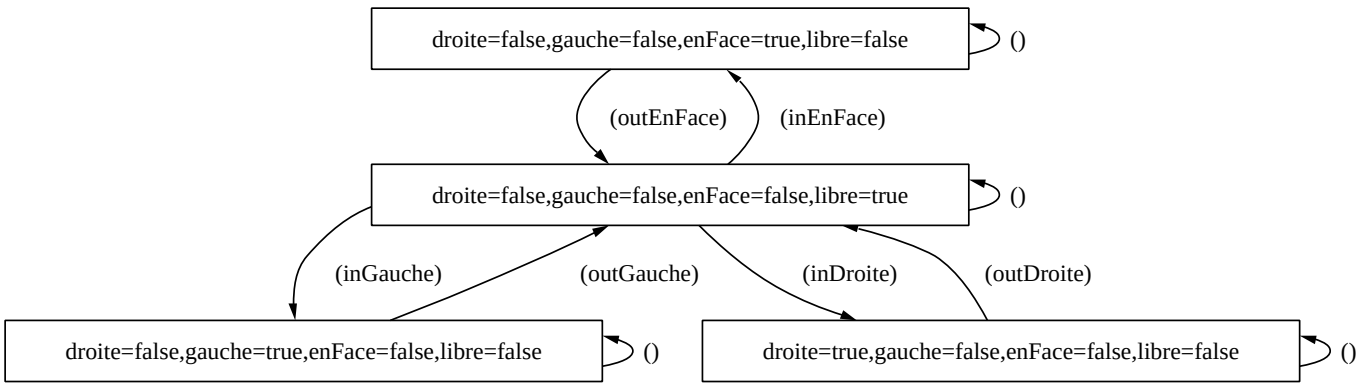

FIG. 1.2 – Le nœud Place

Chapitre 2

Un premier carrefour non contrôlé

Un système est composé :
– de quatre places indiquant la présence de voitures dans le carrefour et leur direction,
– de quatre voies modélisant les quatre files d'attente.

2.1 Le nœud SystemV0

2.1.1 Le source AltaRica

```
node SystemV0
  sub
    V1, V2, V3, V4 : Voie;
    P41, P12, P23, P34 : Place;
  sync
    // les voitures entrent dans le carrefour
    <V1.droite, P41.inDroite>;
    <V1.gauche, P41.inGauche>;
    <V1.enFace, P41.inEnFace>;
    <V2.droite, P12.inDroite>;
    <V2.gauche, P12.inGauche>;
    <V2.enFace, P12.inEnFace>;
    <V3.droite, P23.inDroite>;
    <V3.gauche, P23.inGauche>;
    <V3.enFace, P23.inEnFace>;
    <V4.droite, P34.inDroite>;
    <V4.gauche, P34.inGauche>;
    <V4.enFace, P34.inEnFace>;
    // les voitures changent de position et de direction dans le carrefour
    // pour tourner à gauche il faut ensuite aller en face
    <P41.outGauche, P12.inEnFace>;
    <P12.outGauche, P23.inEnFace>;
    <P23.outGauche, P34.inEnFace>;
    <P34.outGauche, P41.inEnFace>;
    // pour aller en face, il faut tourner à droite !
    <P41.outEnFace, P12.inDroite>;
    <P12.outEnFace, P23.inDroite>;
    <P23.outEnFace, P34.inDroite>;
    <P34.outEnFace, P41.inDroite>;
edon
```

Question 1 (1 point) *Expliquez le blocage de la figure 2.1.*

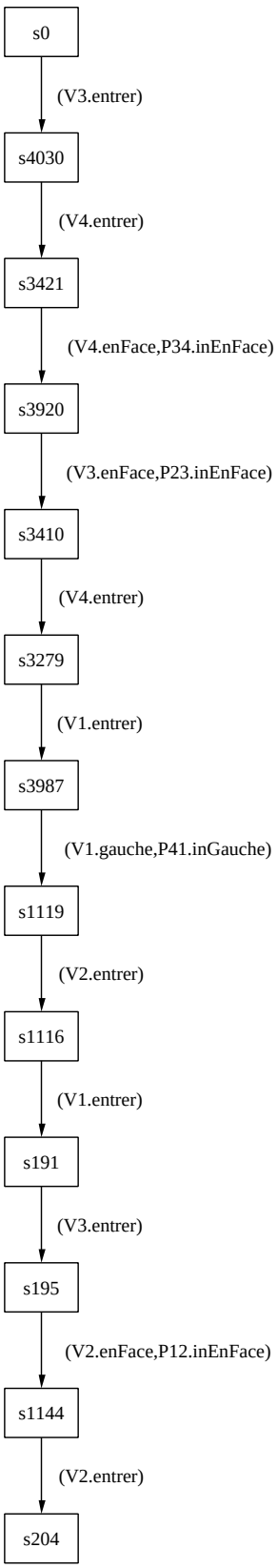

FIG. 2.1 – Une situation d'inter-blocage

2.1.2 Les propriétés

Les spécifications

Question 2 (3 points) *Complétez les spécifications* ACHECK *(*`marcheArriere` *et* `demiTour`*) du composant* `SystemV0`*, afin d'obtenir les résultats suivants.*

```
TEST(any_s=0) [FAILED] actual size = 4096
TEST(any_t=0) [FAILED] actual size = 24576
TEST(notCFC=0) [FAILED] actual size = 1024
TEST(dead=0) [FAILED] actual size = 16
TEST(deadTime=0) [FAILED] actual size = 16
TEST(unavDeadTime=0) [FAILED] actual size = 256
TEST(sControlable=0) [FAILED] actual size = 3840
TEST(tControlable=0) [FAILED] actual size = 23296
TEST(marcheArriere=0) [PASSED]
TEST(demiTour=0) [PASSED]
```

```
with SystemV0 do
 //quot()                    >  '$NODENAME.dot';
 dead                      := any_s - src(any_t - self_epsilon);
 time                      := self_epsilon;
 mustTime                  := src(time) - src(any_t -time);
 loopTime                  := loop(rsrc(mustTime)&rtgt(mustTime),
                                   rsrc(mustTime)&rtgt(mustTime));
 deadTime                  := dead | src(loopTime);
 unavDeadTime              := unav(any_t -self_epsilon, deadTime);
 sControlable              := any_s - unavDeadTime;
 tControlable              := any_t - rtgt(unavDeadTime);
 notCFC                    := any_t - loop(any_t,any_t);
 traceDead                 := trace(initial,any_t, dead);
 dot(src(traceDead)|tgt(traceDead), traceDead) > '$NODENAME.Dead.dot';
 // des calculs auxiliaires (P41,P12,P23,P34) avec o:occupe, l:libre, x:quelconque
 olxx := [~P41.libre] & [P12.libre];
 loxx := [P41.libre] & [~P12.libre];
 xolx := [~P12.libre] & [P23.libre];
 xlox := [P12.libre] & [~P23.libre];
 xxol := [~P23.libre] & [P34.libre];
 xxlo := [P23.libre] & [~P34.libre];
 lxxo := [~P34.libre] & [P41.libre];
 oxxl := [P34.libre] & [~P41.libre];
 // une voiture va de P41 à P12
 P41toP12 := rsrc(olxx) & rtgt(loxx);
 // les voitures en marche arriere !
 marcheArriere := (rtgt(olxx) & rsrc(loxx)) |
                  (rtgt(xolx) & rsrc(xlox)) |
                  (rtgt(xxol) & rsrc(xxlo)) |
                  (rtgt(lxxo) & rsrc(oxxl)) ;
 // demi-tour P41,P12,P23,P34
 demiTour := rsrc(reach(tgt(rsrc(reach(tgt(rsrc(olxx)&rtgt(loxx)),
                                       rsrc([~P12.libre])&rtgt([~P12.libre]))
                                   & xolx)
                               & rtgt(xlox)),
                        rsrc([~P23.libre])&rtgt([~P23.libre]))
                  & xxol)
             & rtgt(xxlo);
 show(all)                 >  '$NODENAME.prop';
 test(any_s,0)             >  '$NODENAME.res';
 test(any_t,0)             >> '$NODENAME.res';
```

```
 test(notCFC,0)           >> '$NODENAME.res';
 test(dead,0)             >> '$NODENAME.res';
 test(deadTime,0)         >> '$NODENAME.res';
 test(unavDeadTime,0)     >> '$NODENAME.res';
 test(sControlable,0)     >> '$NODENAME.res';
 test(tControlable,0)     >> '$NODENAME.res';
 test(marcheArriere,0)    >> '$NODENAME.res';
 test(demiTour,0)         >> '$NODENAME.res';
done
```

Chapitre 3

Un carrefour contrôlé par des feux

Un carrefour contrôlé est composé :
- de quatre places indiquant la présence de voitures dans le carrefour et leur direction,
- de quatre voies modélisant les quatre files d'attente.
- de quatre feux autorisant l'entrée des véhicules dans le carrefour, de telle sorte que les propriétés 1, 2 et 3 soient satisfaites.

3.1 Le nœud Feu

3.1.1 La sémantique

Question 3 (1 point) *Dessinez l'automate de la sémantique souhaitée du composant* `Feu`

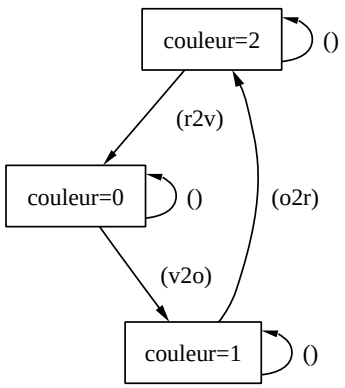

FIG. 3.1 – Le nœud Feu

3.1.2 Le source AltaRica

Question 4 (1 point) *Donnez le source* ALTARICA *du composant* `Feu`

```
node Feu
  state
    couleur : [0,2] : public;
  event
    v2o, o2r, r2v;
  trans
    couleur = 0 |- v2o -> couleur := 1; // vert à orange
    couleur = 1 |- o2r -> couleur := 2; // orange à rouge
    couleur = 2 |- r2v -> couleur := 0; // rouge à vert
edon
```

3.2 Le nœud SystemV1

3.2.1 Le source AltaRica

Question 5 (3 points) *Complétez le source* AltaRica *du composant* `SystemV1`

Voici plusieurs contrôleurs possibles basés sur des abstractions de la vie réelle.

V1 : Un feu vert à la fois : le contrôleur est très strict, un seul feu à la fois, et il ne passe vert que si le carrefour est totalement libre de véhicule. Un ordonnancement circulaire assure qu'il n'y aura pas d'attente infinie.

```
node SystemV1
  sub
    F1, F2, F3, F4 : Feu;
    V1, V2, V3, V4 : Voie;
    P41, P12, P23, P34 : Place;
  flow
    libre : bool;
  state
    feu : [1,4];
  event
    v1Entrer, v2Entrer, v3Entrer, v4Entrer;
    f1ToVert, f1ToOrange, f1ToRouge;
    f2ToVert, f2ToOrange, f2ToRouge;
    f3ToVert, f3ToOrange, f3ToRouge;
    f4ToVert, f4ToOrange, f4ToRouge;
  trans
    // les entrées des voitures dans le carrefour
    F1.couleur=0 |- v1Entrer -> ;
    F2.couleur=0 |- v2Entrer -> ;
    F3.couleur=0 |- v3Entrer -> ;
    F4.couleur=0 |- v4Entrer -> ;
    // l'ordonnanceur des feux
    feu = 1 & libre |- f1ToVert -> ;
    feu = 1         |- f1ToOrange -> ;
    feu = 1         |- f1ToRouge -> feu := 2;
    feu = 2 & libre |- f2ToVert -> ;
    feu = 2         |- f2ToOrange -> ;
    feu = 2         |- f2ToRouge -> feu := 3;
    feu = 3 & libre |- f3ToVert -> ;
    feu = 3         |- f3ToOrange -> ;
    feu = 3         |- f3ToRouge -> feu := 4;
    feu = 4 & libre |- f4ToVert -> ;
    feu = 4         |- f4ToOrange -> ;
    feu = 4         |- f4ToRouge -> feu := 1;
  sync
    // Evolution des feux
    <f1ToVert, F1.r2v>; <f1ToOrange, F1.v2o>; <f1ToRouge, F1.o2r>;
    <f2ToVert, F2.r2v>; <f2ToOrange, F2.v2o>; <f2ToRouge, F2.o2r>;
    <f3ToVert, F3.r2v>; <f3ToOrange, F3.v2o>; <f3ToRouge, F3.o2r>;
    <f4ToVert, F4.r2v>; <f4ToOrange, F4.v2o>; <f4ToRouge, F4.o2r>;
    // les voitures entrent dans le carrefour
    <v1Entrer, V1.droite, P41.inDroite>;
    <v1Entrer, V1.gauche, P41.inGauche>;
    <v1Entrer, V1.enFace, P41.inEnFace>;
    <v2Entrer, V2.droite, P12.inDroite>;
    <v2Entrer, V2.gauche, P12.inGauche>;
    <v2Entrer, V2.enFace, P12.inEnFace>;
    <v3Entrer, V3.droite, P23.inDroite>;
    <v3Entrer, V3.gauche, P23.inGauche>;
```

```
    <v3Entrer, V3.enFace, P23.inEnFace>;
    <v4Entrer, V4.droite, P34.inDroite>;
    <v4Entrer, V4.gauche, P34.inGauche>;
    <v4Entrer, V4.enFace, P34.inEnFace>;
    // les voitures changent de position et de direction dans le carrefour
    // pour tourner à gauche il faut ensuite aller en face
    <P41.outGauche, P12.inEnFace>;
    <P12.outGauche, P23.inEnFace>;
    <P23.outGauche, P34.inEnFace>;
    <P34.outGauche, P41.inEnFace>;
    // pour aller en face, il faut tourner à droite !
    <P41.outEnFace, P12.inDroite>;
    <P12.outEnFace, P23.inDroite>;
    <P23.outEnFace, P34.inDroite>;
    <P34.outEnFace, P41.inDroite>;
  assert
    libre = (P41.libre & P12.libre & P23.libre & P34.libre);
  init
    feu := 1,
    F1.couleur:=2,
    F2.couleur:=2,
    F3.couleur:=2,
    F4.couleur:=2;
edon
```

V2 : Deux feux verts opposés à la fois : les feux opposés passent verts simultanément et à condition que le carrefour soit libre.

```
node SystemV2
  sub
    F1, F2, F3, F4 : Feu;
    V1, V2, V3, V4 : Voie;
    P41, P12, P23, P34 : Place;
  flow
    libre : bool;
  state
    feu : [1,2];
  event
    v1Entrer, v2Entrer, v3Entrer, v4Entrer;
    f13ToVert, f13ToOrange, f13ToRouge;
    f24ToVert, f24ToOrange, f24ToRouge;
  trans
    // les entrées des voitures dans le carrefour
    F1.couleur=0 |- v1Entrer -> ;
    F2.couleur=0 |- v2Entrer -> ;
    F3.couleur=0 |- v3Entrer -> ;
    F4.couleur=0 |- v4Entrer -> ;
    // l'ordonnanceur des feux
    feu = 1 & libre |- f13ToVert -> ;
    feu = 1         |- f13ToOrange -> ;
    feu = 1         |- f13ToRouge -> feu := 2;
    feu = 2 & libre |- f24ToVert -> ;
    feu = 2         |- f24ToOrange -> ;
    feu = 2         |- f24ToRouge -> feu := 1;
  sync
    // Evolution des feux
    <f13ToVert,   F1.r2v, F3.r2v>;
    <f13ToOrange, F1.v2o, F3.v2o>;
    <f13ToRouge,  F1.o2r, F3.o2r>;
```

```
    <f24ToVert,   F2.r2v, F4.r2v>;
    <f24ToOrange, F2.v2o, F4.v2o>;
    <f24ToRouge,  F2.o2r, F4.o2r>;
    // les voitures entrent dans le carrefour
    <v1Entrer, V1.droite, P41.inDroite>;
    <v1Entrer, V1.gauche, P41.inGauche>;
    <v1Entrer, V1.enFace, P41.inEnFace>;
    <v2Entrer, V2.droite, P12.inDroite>;
    <v2Entrer, V2.gauche, P12.inGauche>;
    <v2Entrer, V2.enFace, P12.inEnFace>;
    <v3Entrer, V3.droite, P23.inDroite>;
    <v3Entrer, V3.gauche, P23.inGauche>;
    <v3Entrer, V3.enFace, P23.inEnFace>;
    <v4Entrer, V4.droite, P34.inDroite>;
    <v4Entrer, V4.gauche, P34.inGauche>;
    <v4Entrer, V4.enFace, P34.inEnFace>;
    // les voitures changent de position et de direction dans le carrefour
    // pour tourner à gauche il faut ensuite aller en face
    <P41.outGauche, P12.inEnFace>;
    <P12.outGauche, P23.inEnFace>;
    <P23.outGauche, P34.inEnFace>;
    <P34.outGauche, P41.inEnFace>;
    // pour aller en face, il faut tourner à droite !
    <P41.outEnFace, P12.inDroite>;
    <P12.outEnFace, P23.inDroite>;
    <P23.outEnFace, P34.inDroite>;
    <P34.outEnFace, P41.inDroite>;
  assert
    libre = (P41.libre & P12.libre & P23.libre & P34.libre);
  init
    feu := 1,
    F1.couleur:=2,
    F2.couleur:=2,
    F3.couleur:=2,
    F4.couleur:=2;
edon
```

V3 : Deux feux verts successifs à la fois : les feux opposés passent vert et rouge simultanément et à condition que la place de la file passant au rouge soit libre.

```
node SystemV3
  sub
    F1, F2, F3, F4 : Feu;
    V1, V2, V3, V4 : Voie;
    P41, P12, P23, P34 : Place;
  state
    feu : [1,4];
  event
    v1Entrer, v2Entrer, v3Entrer, v4Entrer;
    f1ToOrange, f2ToOrange, f3ToOrange, f4ToOrange;
    f13ToRV, f24ToRV, f31ToRV, f42ToRV;
  trans
    // les entrées des voitures dans le carrefour
    F1.couleur=0 |- v1Entrer -> ;
    F2.couleur=0 |- v2Entrer -> ;
    F3.couleur=0 |- v3Entrer -> ;
    F4.couleur=0 |- v4Entrer -> ;
    // l'ordonnanceur des feux
    feu = 1             |- f1ToOrange -> ;
```

```
    feu = 1 & P41.libre |- f13ToRV    -> feu := 2 ;
    feu = 2             |- f2ToOrange -> ;
    feu = 2 & P12.libre |- f24ToRV    -> feu := 3 ;
    feu = 3             |- f3ToOrange -> ;
    feu = 3 & P23.libre |- f31ToRV    -> feu := 4 ;
    feu = 4             |- f4ToOrange -> ;
    feu = 4 & P34.libre |- f42ToRV    -> feu := 1 ;
  sync
    // Evolution des feux
    <f1ToOrange, F1.v2o>;
    <f2ToOrange, F2.v2o>;
    <f3ToOrange, F3.v2o>;
    <f4ToOrange, F4.v2o>;
    <f13ToRV,  F1.o2r, F3.r2v>;
    <f24ToRV,  F2.o2r, F4.r2v>;
    <f31ToRV,  F3.o2r, F1.r2v>;
    <f42ToRV,  F4.o2r, F2.r2v>;
    // les voitures entrent dans le carrefour
    <v1Entrer, V1.droite, P41.inDroite>;
    <v1Entrer, V1.gauche, P41.inGauche>;
    <v1Entrer, V1.enFace, P41.inEnFace>;
    <v2Entrer, V2.droite, P12.inDroite>;
    <v2Entrer, V2.gauche, P12.inGauche>;
    <v2Entrer, V2.enFace, P12.inEnFace>;
    <v3Entrer, V3.droite, P23.inDroite>;
    <v3Entrer, V3.gauche, P23.inGauche>;
    <v3Entrer, V3.enFace, P23.inEnFace>;
    <v4Entrer, V4.droite, P34.inDroite>;
    <v4Entrer, V4.gauche, P34.inGauche>;
    <v4Entrer, V4.enFace, P34.inEnFace>;
    // les voitures changent de position et de direction dans le carrefour
    // pour tourner à gauche il faut ensuite aller en face
    <P41.outGauche, P12.inEnFace>;
    <P12.outGauche, P23.inEnFace>;
    <P23.outGauche, P34.inEnFace>;
    <P34.outGauche, P41.inEnFace>;
    // pour aller en face, il faut tourner à droite !
    <P41.outEnFace, P12.inDroite>;
    <P12.outEnFace, P23.inDroite>;
    <P23.outEnFace, P34.inDroite>;
    <P34.outEnFace, P41.inDroite>;
  init
    feu := 1,
    F1.couleur:=0,
    F2.couleur:=0,
    F3.couleur:=2,
    F4.couleur:=2;
edon
```

V4 : carrefour australien : les véhicules présents dans le carrefour sont prioritaires. Une voiture ne peux entrer que si la place à sa *gauche* est libre.

```
node SystemV4
  sub
    V1, V2, V3, V4 : Voie;
    P41, P12, P23, P34 : Place;
  event
    v1Entrer, v2Entrer, v3Entrer, v4Entrer;
  trans
```

```
    // les entrées des voitures dans le carrefour
    P34.libre |- v1Entrer -> ;
    P41.libre |- v2Entrer -> ;
    P12.libre |- v3Entrer -> ;
    P23.libre |- v4Entrer -> ;
  sync
    // les voitures entrent dans le carrefour
    <v1Entrer, V1.droite, P41.inDroite>;
    <v1Entrer, V1.gauche, P41.inGauche>;
    <v1Entrer, V1.enFace, P41.inEnFace>;
    <v2Entrer, V2.droite, P12.inDroite>;
    <v2Entrer, V2.gauche, P12.inGauche>;
    <v2Entrer, V2.enFace, P12.inEnFace>;
    <v3Entrer, V3.droite, P23.inDroite>;
    <v3Entrer, V3.gauche, P23.inGauche>;
    <v3Entrer, V3.enFace, P23.inEnFace>;
    <v4Entrer, V4.droite, P34.inDroite>;
    <v4Entrer, V4.gauche, P34.inGauche>;
    <v4Entrer, V4.enFace, P34.inEnFace>;
    // les voitures changent de position et de direction dans le carrefour
    // pour tourner à gauche il faut ensuite aller en face
    <P41.outGauche, P12.inEnFace>;
    <P12.outGauche, P23.inEnFace>;
    <P23.outGauche, P34.inEnFace>;
    <P34.outGauche, P41.inEnFace>;
    // pour aller en face, il faut tourner à droite !
    <P41.outEnFace, P12.inDroite>;
    <P12.outEnFace, P23.inDroite>;
    <P23.outEnFace, P34.inDroite>;
    <P34.outEnFace, P41.inDroite>;
edon
```

V5 : contrôle global sur les places : un véhicule ne peut entrer que si il y a deux places libres dans le carrefour (celle qu'il va prendre et une autre).

```
node SystemV5
  sub
    V1, V2, V3, V4 : Voie;
    P41, P12, P23, P34 : Place;
  sync
    // les voitures entrent dans le carrefour
    <V1.droite, P41.inDroite>;
    <V1.gauche, P41.inGauche>;
    <V1.enFace, P41.inEnFace>;
    <V2.droite, P12.inDroite>;
    <V2.gauche, P12.inGauche>;
    <V2.enFace, P12.inEnFace>;
    <V3.droite, P23.inDroite>;
    <V3.gauche, P23.inGauche>;
    <V3.enFace, P23.inEnFace>;
    <V4.droite, P34.inDroite>;
    <V4.gauche, P34.inGauche>;
    <V4.enFace, P34.inEnFace>;
    // les voitures changent de position et de direction dans le carrefour
    // pour tourner à gauche il faut ensuite aller en face
    <P41.outGauche, P12.inEnFace>;
    <P12.outGauche, P23.inEnFace>;
    <P23.outGauche, P34.inEnFace>;
    <P34.outGauche, P41.inEnFace>;
```

```
    // pour aller en face, il faut tourner à droite !
    <P41.outEnFace, P12.inDroite>;
    <P12.outEnFace, P23.inDroite>;
    <P23.outEnFace, P34.inDroite>;
    <P34.outEnFace, P41.inDroite>;
  assert
    P41.libre | P12.libre | P23.libre | P34.libre;
edon
```

V6 : carrefour australien et confiance dans les clignotants : les véhicules présents dans le carrefour sont prioritaires. Une voiture ne peux entrer que si la place à sa *gauche* est soit libre, soit utilisée par une voiture ayant ses clignotants indiquant sa volonté de sortie à droite.

```
node SystemV6
  sub
    V1, V2, V3, V4 : Voie;
    P41, P12, P23, P34 : Place;
  event
    v1Entrer, v2Entrer, v3Entrer, v4Entrer;
  trans
    // les entrées des voitures dans le carrefour
    P34.libre | P34.droite |- v1Entrer -> ;
    P41.libre | P41.droite |- v2Entrer -> ;
    P12.libre | P12.droite |- v3Entrer -> ;
    P23.libre | P23.droite |- v4Entrer -> ;
  sync
    // les voitures entrent dans le carrefour
    <v1Entrer, V1.droite, P41.inDroite>;
    <v1Entrer, V1.gauche, P41.inGauche>;
    <v1Entrer, V1.enFace, P41.inEnFace>;
    <v2Entrer, V2.droite, P12.inDroite>;
    <v2Entrer, V2.gauche, P12.inGauche>;
    <v2Entrer, V2.enFace, P12.inEnFace>;
    <v3Entrer, V3.droite, P23.inDroite>;
    <v3Entrer, V3.gauche, P23.inGauche>;
    <v3Entrer, V3.enFace, P23.inEnFace>;
    <v4Entrer, V4.droite, P34.inDroite>;
    <v4Entrer, V4.gauche, P34.inGauche>;
    <v4Entrer, V4.enFace, P34.inEnFace>;
    // les voitures changent de position et de direction dans le carrefour
    // pour tourner à gauche il faut ensuite aller en face
    <P41.outGauche, P12.inEnFace>;
    <P12.outGauche, P23.inEnFace>;
    <P23.outGauche, P34.inEnFace>;
    <P34.outGauche, P41.inEnFace>;
    // pour aller en face, il faut tourner à droite !
    <P41.outEnFace, P12.inDroite>;
    <P12.outEnFace, P23.inDroite>;
    <P23.outEnFace, P34.inDroite>;
    <P34.outEnFace, P41.inDroite>;
edon
```

V7 : contrôle global sur les places et les clignotants : un véhicule ne peut entrer que si il y a deux places libres dans le carrefour (celle qu'il va prendre et une autre), ou bien si une des voitures du carrefour peut en sortir.

```
node SystemV7
  sub
    V1, V2, V3, V4 : Voie;
    P41, P12, P23, P34 : Place;
  sync
```

```
    // les voitures entrent dans le carrefour
    <V1.droite, P41.inDroite>;
    <V1.gauche, P41.inGauche>;
    <V1.enFace, P41.inEnFace>;
    <V2.droite, P12.inDroite>;
    <V2.gauche, P12.inGauche>;
    <V2.enFace, P12.inEnFace>;
    <V3.droite, P23.inDroite>;
    <V3.gauche, P23.inGauche>;
    <V3.enFace, P23.inEnFace>;
    <V4.droite, P34.inDroite>;
    <V4.gauche, P34.inGauche>;
    <V4.enFace, P34.inEnFace>;
    // les voitures changent de position et de direction dans le carrefour
    // pour tourner à gauche il faut ensuite aller en face
    <P41.outGauche, P12.inEnFace>;
    <P12.outGauche, P23.inEnFace>;
    <P23.outGauche, P34.inEnFace>;
    <P34.outGauche, P41.inEnFace>;
    // pour aller en face, il faut tourner à droite !
    <P41.outEnFace, P12.inDroite>;
    <P12.outEnFace, P23.inDroite>;
    <P23.outEnFace, P34.inDroite>;
    <P34.outEnFace, P41.inDroite>;
  assert
    P41.libre | P12.libre | P23.libre | P34.libre |
    P41.droite | P12.droite | P23.droite | P34.droite;
edon
```

3.2.2 Les propriétés

Question 6 (1 point) *En fonction de la confiance que vous avez dans votre conception ; indiquez (en entourant la bonne réponse) si votre composant* `SystemV1` *satisfait les propriétés du cahier des charges.*

V1 : Un feu vert à la fois : les propriétés :

```
TEST(any_s=0) [FAILED] actual size = 4608
TEST(any_t=0) [FAILED] actual size = 24064
TEST(dead=0) [PASSED]
TEST(vraiDead=0) [PASSED]
TEST(notCFC=0) [PASSED]
TEST(marcheArriere=0) [PASSED]
TEST(demiTour=0) [PASSED]
```

Ce contrôleur très strict garantit l'absence blocage.

V2 : Deux feux verts opposés à la fois : les propriétés :

```
TEST(any_s=0) [FAILED] actual size = 13824
TEST(any_t=0) [FAILED] actual size = 77216
TEST(dead=0) [FAILED] actual size = 8
TEST(vraiDead=0) [FAILED] actual size = 24
TEST(notCFC=0) [FAILED] actual size = 1152
TEST(marcheArriere=0) [PASSED]
TEST(demiTour=0) [PASSED]
```

Deux voies entrantes opposées suffisent pour provoquer un inter-blocage !

V3 : Deux feux verts successifs à la fois : les propriétés :

```
TEST(any_s=0) [FAILED] actual size = 12288
TEST(any_t=0) [FAILED] actual size = 73472
TEST(dead=0) [PASSED]
```

```
TEST(vraiDead=0) [PASSED]
TEST(notCFC=0) [PASSED]
TEST(marcheArriere=0) [PASSED]
TEST(demiTour=0) [PASSED]
```

Deux voies entrantes *successives* ne provoquent pas d'inter-blocage, malgré un contrôle en entrée plus souple.

V4 : carrefour australien : les propriétés :

```
TEST(any_s=0) [FAILED] actual size = 2800
TEST(any_t=0) [FAILED] actual size = 14352
TEST(dead=0) [PASSED]
TEST(vraiDead=0) [PASSED]
TEST(notCFC=0) [PASSED]
TEST(marcheArriere=0) [PASSED]
TEST(demiTour=0) [PASSED]
```

Cela explique peut-être la prolifération des ronds-points en France !

V5 : contrôle global sur les places : les propriétés :

```
TEST(any_s=0) [FAILED] actual size = 2800
TEST(any_t=0) [FAILED] actual size = 16368
TEST(dead=0) [PASSED]
TEST(vraiDead=0) [PASSED]
TEST(notCFC=0) [PASSED]
TEST(marcheArriere=0) [PASSED]
TEST(demiTour=0) [PASSED]
```

Pas d'inter-blocage, mais l'implémentation n'est peut-être pas aisée !

V6 : carrefour australien et confiance dans les clignotants : les propriétés :

```
TEST(any_s=0) [FAILED] actual size = 3840
TEST(any_t=0) [FAILED] actual size = 20736
TEST(dead=0) [PASSED]
TEST(vraiDead=0) [PASSED]
TEST(notCFC=0) [PASSED]
TEST(marcheArriere=0) [PASSED]
TEST(demiTour=0) [PASSED]
```

Cela explique peut-être la prolifération des accidents sur les ronds-points en France !

V7 : contrôle global sur les places et les clignotants : les propriétés :

```
TEST(any_s=0) [FAILED] actual size = 3840
TEST(any_t=0) [FAILED] actual size = 23296
TEST(dead=0) [PASSED]
TEST(vraiDead=0) [PASSED]
TEST(notCFC=0) [PASSED]
TEST(marcheArriere=0) [PASSED]
TEST(demiTour=0) [PASSED]
```

Les nombres d'états et de transitions sont les mêmes que les nombres d'états et de transitions contrôlable de la version V0. C'est donc le contrôleur maximal, mais l'implémentation n'est peut-être pas aisée !