<u>Correction</u> 1 : *Carrefour*

L'état du système est caractérisé par trois booléen qui expriment l'état (allumé ou éteint des chaque couleur du feu).

```
Signal: red, yellow, green: boolean
```

Le passage à vert du feu est fait dans le système français directement à partir du rouge :

```
Go-french:
          red and not yellow  and not green and
     not red' and not yellow' and green
```

Puis, le passage à rouge est fait par un feu jaune :

```
WarnStop:
    not red  and not yellow and green and
    not red' and yellow'    and not green'
Stop:
    not red and yellow      and not green and
       red' and not yellow' and not green'
```

Donc le système final est :

```
TrafficSignal : Signal and (Go-french or WarnStop or Stop)
```

L'invariant est red xor green xor yellow. Un état initial qui satisfait cet invariant est

```
InitRed: red and not yellow and not green
```

Dans le système anglo-saxon, on doit passer par l'état intermédiaire WarningGo :

```
WarnGo:
         red  and  not yellow  and not green and
         red' and  yellow' and not green
Go-gb:
         red and yellow  and not green and
    not red' and not yellow' and green
```

Pour obtenir le carrefour, il suffit de combiner les deux paires de feux afin de les synchroniser.

```
EstWest, NordSud : TrafficSignal
EstWest.red = (NordSud.green or NordSud.yellow) and
(EstWest.green or EstWest.yellow) = NordSud.red and
```

Correction 2 : *Édition concurrente de programmes*

Domaines :

```
[PERSON, DOCUMENT]
```

Les permissions pour changer les documents peuvent être représentées par une relation document et utilisateurs :

```
permission: DOCUMENT <--> PERSON
doug, aki, phil: PERSON
spec, design, code: DOCUMENT
permission = { (spec, doug), (design, doug), (design, aki), (code, aki), (code, phil)
```

L'état du système est une autre relation, qui concerne les mêmes domaines. Toutefois, comme la contrainte principale du système est qu'un document peut être *checked out* par une seule personne, la relation est une fonction, pour chaque document on associe au plus une personne (donc c'est une fonction partielle).

```
Documents
-----
checked_out: DOCUMENT -|-> PERSON
-----
checked_out subseteq permission
```

Le prédicat invariant dit que le documents peuvent uniquement être checked_out par les personnes ayant le droit.

Exemple d'état possible :

```
checked_out = { (design, doug), (spec, doug), (code, phil) }
```

Deux opérations changent l'état, CheckOut et CheckIn.

Le CheckOut a deux paramètres d'entrée, la personne p ? et le document d ?.

```
CheckOut
Delta Documents
```

```
p?: PERSON
d?: DOCUMENT
where
d? notin dom checked_out
(d?, p?) in permission
checked_out' = checked_out \cup {(d?, p?)
```

CheckOut a deux préconditions. Premièrement, le document d ? ne doit pas être déjà checked out, i.e., il ne doit être dans le domaine de checked_out. De plus, la personne qui fait le checking out doit avoir la permission sur le document, i.e., (d ?, p ?) appartient aux permissions. Si les préconditions sont satisfaites, on ajoute la paire (d ?,p ?) à checked_out, ce qui annule tout essai de checking out sur le document.

Si les préconditions ne sont pas satisfaites, il y a deux cas (car deux préconditions). Le premier dit que le document est déjà checked out : $d?indomchecked\_out$. Le deuxième dit que la personne n'a pas la permission : $(d?, p?)notinpermission$. Dans les deux cas, rien n'est checked out et l'état du système ne change pas.

```
Unauthorized
Xi Documents
p?: PERSON
d?: DOCUMENT
where
        d? in dom checked_out
or
(d?,p?) notin permission
```

Le prédicat final T_CheckOut couvre toutes les possibilités.

```
T_CheckOut = CheckOut or CheckedOut or Unauthorized
```