UNIVERSITÉ BORDEAUX 1 Sciences Technologies Master 1 Informatique	Année : 2012/2013 Semestre 2 **Parcours :** Master 1 Informatique **UE J1IN8W14 :** Conceptions Formelles **Date :** Vendredi 19 avril 2013 **Heure :** 8h 30 **Durée conseillée :** 1 heure 30 **Documents :** autorisés Épreuve de M. Alain Griffault

Code d'anonymat :

Avertissement

- La plupart des questions sont indépendantes.
- L'espace laissé pour les réponses est suffisant (sauf si vous utilisez ces feuilles comme brouillon, ce qui est fortement déconseillé).

Question	Points	Score
Modélisation en AltaRica	12	
Vérification avec `ARC`	8	
Total:	20	

L'objectif de l'étude est l'optimisation de la disponibilité d'une ressource de type *moteur*. Ce moteur pouvant défaillir, ou bien être arrêter pour des opérations de maintenance, le système étudié est composé de deux moteurs, afin qu'ils puissent mutuellement se remplacer.

Exercice 1 : Modélisation en AltaRica (12 points)

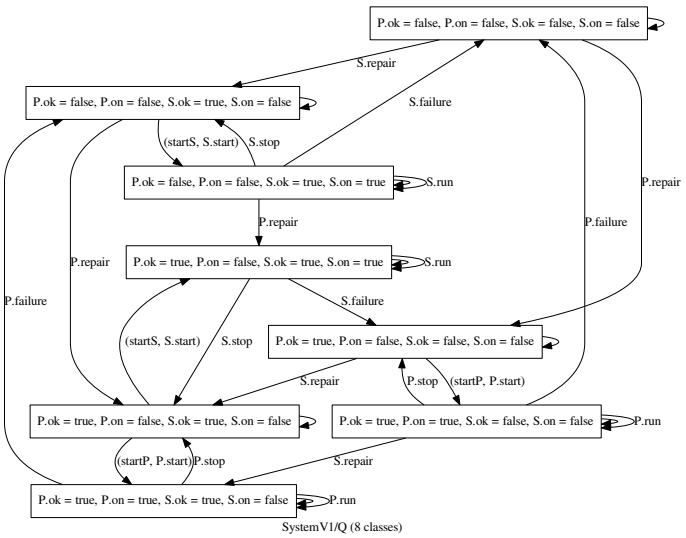
(a) (3 points) Dessinez la sémantique du code AltaRica suivant.

```
node Motor
     state on, ok : bool:public;
     init  on := False, ok := True;
     event start, stop, failure, repair, run;
     trans
       ~on & ok |- start -> on :=true;
       on & ok |- failure -> on :=false, ok := false;
       on & ok |- stop -> on :=false;
       on & ok |- run -> ;
       ~on & ~ok |- repair -> ok := True;
edon
```

(b) (3 points) Complétez le code AltaRica du noeud `SystemV1` afin que la sémantique soit celle du graphe suivant. Dans cette première version :

– **Les deux moteurs ne doivent jamais fonctionner (`on = True`) simultanément.**

Ce contrôle peut être réalisé en AltaRica de plusieurs façons. Votre solution doit utiliser un contrôle par synchronisation d'événements.

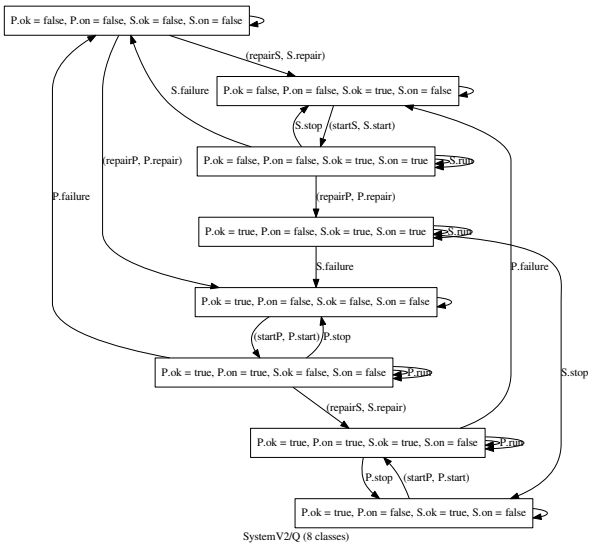



```
node SystemV1
  sub P,S : Motor;
  event
```

(c) (3 points) Complétez le code AltaRica du noeud `SystemV2` afin que la sémantique soit celle du graphe suivant. Dans cette seconde version :

- Les deux moteurs ne doivent jamais fonctionner (`on = True`) simultanément.
- **Lorsqu'un moteur tombe en panne, il est plus urgent de démarrer l'autre moteur que de faire des réparations.**
- **Si les deux moteurs sont en état de marche et arrêtés, il est préférable d'utiliser le moteur *Primaire*, au lieu du *secondaire*.**

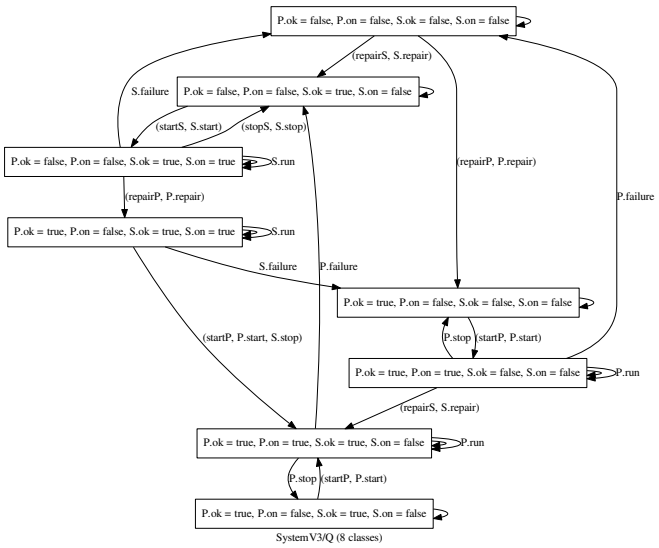
Ce contrôle peut être réalisé en AltaRica de plusieurs façons. Votre solution doit utiliser un contrôle par synchronisation d'événements, et des priorités.



```
node SystemV2
  sub P,S : Motor;
  event
```

(d) (3 points) Complétez le code AltaRica du noeud `SystemV3` afin que la sémantique soit celle du graphe suivant. Dans cette troisième version :
- Les deux moteurs ne doivent jamais fonctionner (`on = True`) simultanément.
- Lorsqu'un moteur tombe en panne, il est plus urgent de démarrer l'autre moteur que de faire des réparations.
- Si les deux moteurs sont en état de marche et arrêtés, il est préférable d'utiliser le moteur *Primaire*, au lieu du *secondaire*.
- **Dès que le moteur *Primaire* est en état de marche, il est préférable de l'utiliser, quitte à arrêter le moteur secondaire.**

Ce contrôle peut être réalisé en AltaRica de plusieurs façons. Votre solution doit utiliser un contrôle par synchronisation d'événements, et des priorités.




```
node SystemV3
  sub P,S : Motor;
  event
```

Exercice 2 : Vérification avec ARC (8 points)

(a) (2 points) Écrivez la propriété qui permet de vérifier que tous ces systèmes sont sans blocage.

```
with SystemV1, SystemV2, SystemV3 do
  deadlock :=
```

(b) (2 points) Écrivez une propriété qui permet de vérifier que les deux moteurs ne sont jamais simultanément en marche.

```
with SystemV1, SystemV2, SystemV3 do
  bothOn :=
```

(c) (2 points) Écrivez des propriétés qui montrent que, lorsque les deux moteurs pourraient démarrer, le moteur `Primaire` est toujours utilisé ; et jamais le `Secondaire`.

```
with SystemV1, SystemV2, SystemV3 do
```

(d) (2 points) Écrivez des propriétés qui montrent que le moteur `Primaire` est utilisé dès qu'il est en état de fonctionnement, et que ce n'est pas le cas pour le moteur `Secondaire`.

```
with SystemV1, SystemV2, SystemV3 do
```