Université
Bordeaux 1

ANNÉE : 2008/2009

PARCOURS : Master 1 Informatique
UE : INF 471, Conceptions Formelles
Module : Modélisation et Vérification
Date : Mercredi 29 avril 2009
Heure : 14 heures
Durée conseillée : 1 heure 30
Documents : autorisés
Épreuve de M. Alain GRIFFAULT

U. F. R.
Mathématique
Informatique

Code d'anonymat :

Avertissement

- Le barème total est de 14 points.
- Le barème est (approximativement) proportionnel à la difficulté de la question.
- L'espace laissé pour les réponses est suffisant (sauf si vous utilisez ces feuilles comme brouillon, ce qui est fortement déconseillé).

1 Variante du circuit électrique (6 points)

Le circuit électrique représenté par la figure 1 est une simplification de celui étudié en cours.

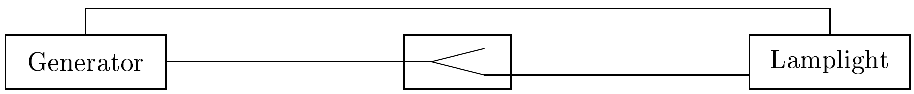

FIG. 1 – Le premier circuit électrique

Le code de l'ampoule est :

```
node Lamplight
  flow    f1, f2 : [0,1];
          light : bool;
  state   ok : bool;
  event   reaction, new;
  trans   ok & ((f1=1) & (f2=1)) |- reaction -> ok := false;
          ~ok                    |- new      -> ok := true;
  assert  light = (ok & (f1 != f2));
  init    ok := true;
edon
```

Question 1.1 (1 point) *Donnez la sémantique de la nouvelle ampoule sous forme de graphe, et expliquez les différences par rapport à la lampe utilisée en cours.*

Nous voulons construire un nouveau circuit électrique représenté par la figure 2

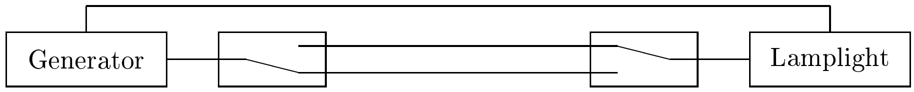

FIG. 2 – Le second circuit éléctrique avec deux interrupteurs va-et-vient

Question 1.2 (1,5 point) *Donnez le code* ALTARICA *d'un interrupteur va-et-vient.*

```
node TwoWaySwitch
```

```
edon
```

Question 1.3 (1,5 point) *Donnez le code* ALTARICA *pour le second circuit électrique.*

```
node CircuitV2
```

```
edon
```

Question 1.4 (2 points) *Donnez les spécifications vous permettant de répondre aux questions suivantes :*

1. *Est-il possible qu'un appui quelconque sur un des interrupteurs laisse la lampe inchangée ?*
2. *Est-il possible que la lampe change de configuration sans aucun appui sur un des interrupteurs ?*

2 Sûreté par redondance (8 points)

L'objectif de l'exercice est la correction de certaines défaillances de fonctions élémentaires par des techniques de redondance. Un équipement est une fonction élémentaire qui prend en entrée une donnée et qui produit en sortie une donnée qui dépend de la donnée en entrée et du mode courant. Le mode courant évolue avec les défaillances : il sera "erroné" après un "bug logiciel" et sera "perdu" après un "bug matériel".
Le système élémentaire abstrait que l'on souhaite sécuriser est décrit par le code ALTARICA suivant :

```
/* L'abstraction d'une fonction élémentaire.
 * Le mode courant dépend de défaillances possibles :
 *   - un "bug logiciel" qui fausse les calculs
 *   - un "bug matériel" qui stoppe la sortie
 * La donnée en sortie est le mode courant.
*/
node Equipement
  state  mode : {ok, erreur, perte} : public;
  event  fail_loss, fail_error : public;
  trans  mode = ok     |- fail_error -> mode := erreur;
         mode != perte |- fail_loss  -> mode := perte;
  init   mode := ok;
edon
```

Question 2.1 (1 point) *Donnez la sémantique du composant* Equipement *sous forme de graphe.*

Question 2.2 (1,5 point) *Complétez le code* ALTARICA *suivant afin de décrire un système élémentaire avec des entrée et sortie plus concrètes.*

```
/* Une fonction élémentaire.
 * Le mode courant dépend de défaillances possibles :
 *   - un "bug logiciel" qui fausse les calculs
 *   - un "bug matériel" qui stoppe la sortie
 * La donnée en entrée est supposée quelconque
 * La donnée en sortie est
 *   - le mode courant si l'entree est "ok"
 *   - "perte" si l'entree est "perte"
 *   - "perte" si le mode est "perte"
 *   - "erreur" dans les autres cas.
*/
node Equipement_reel
  flow   in, out : {ok, erreur, perte};
  state  mode : {ok, erreur, perte};
  init   mode := ok;
  event  fail_loss, fail_error : public;
  trans  mode = ok     |- fail_error -> mode := erreur;
         mode != perte |- fail_loss  -> mode := perte;
  assert




edon
```

Dans de nombreux systèmes critiques, l'absence de sortie est préférable à une sortie erronée. Une technique de redondance pour éviter les sorties erronées consiste d'une part à dupliquer la fonction élémentaire ; d'autre part à ajouter un comparateur des sorties pour détecter les erreurs.

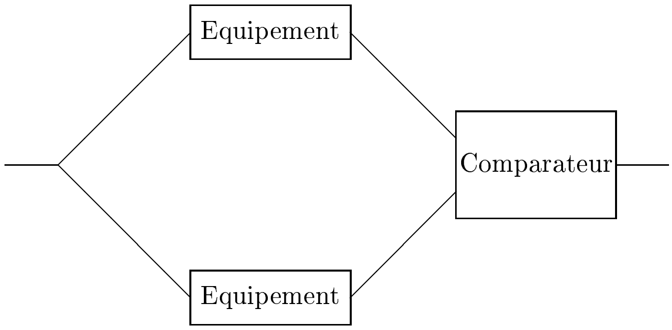

FIG. 3 – La redondance avec comparateur

Le comparateur est décrit par le code ALTARICA suivant :

```
node Comparateur
  flow   in1, in2, out : {ok, erreur, perte};
  state  mode : {ok, different, egal};
  init   mode := ok;
  event  detection : public;
  trans  mode != different |- detection -> mode := different;
         mode  = different |- detection -> mode := egal;
  assert (mode = different) = (in1 != in2);
         (mode  = ok) => (out = in1);
         (mode != ok) => (out = perte);
edon
```

Question 2.3 (1 point) *Donnez la sémantique du composant* `Comparateur` *sous forme de graphe en faisant les hypothèses suivantes :*
– *la variable de flux* `in1` *reste toujours à* `ok`*.*
– *la variable de flux* `in2` *à pour valeur initiale* `ok`*.*

Question 2.4 (1,5 point) *Complétez le code* ALTARICA *suivant qui représente l'architecture de redondance visant à éviter la sortie* `erreur`. *Vous ferez l'hypothèse que les défaillances des deux équipements sont totalement indépendantes, et qu'il est donc impossible qu'elles soient simultanées.*

```
/* Un système de redondance avec comparaison des sorties.
*/
node Redondance_pannes_independantes
  sub    COM, MON : Equipement;
         Comp : Comparateur;
  flow   mode : {ok, erreur, perte}
  assert COM.mode = Comp.in1;
         MON.mode = Comp.in2;
         mode = Comp.out;
  sync







edon
```

Question 2.5 (1 point) *Complétez le code* ALTARICA *suivant qui représente l'architecture de redondance visant à éviter la sortie* `erreur`. *Vous ferez l'hypothèse que les défaillances logicielles des deux équipements sont dépendantes, et qu'il est donc possible (mais pas obligatoire) qu'elles soient simultanées.*

```
/* Un système de redondance avec comparaison des sorties.
*/
node Redondance_pannes_dependantes
  sub    COM, MON : Equipement;
         Comp : Comparateur;
  flow   mode : {ok, erreur, perte}
  assert COM.mode = Comp.in1;
         MON.mode = Comp.in2;
         mode = Comp.out;
  sync







edon
```

Question 2.6 (2 points) *Donnez les spécifications vous permettant de répondre aux questions suivantes :*

1. *Est-il possible que la redondance produise une sortie erronée.*
2. *Est-il possible que des pannes ne soient pas détectées.*
3. *Est-il possible qu'une détection se produise sans aucune panne.*