UPMC -- Master Informatique 15 novembre 2010
Parcours “Systèmes et Applications Répartis” (SAR)
Module “Langages Synchrones”
Année universitaire 2010-2011
E. Encrenaz

Examen, durée 2h
Notes de cours et de TP autorisées

L'examen comprend 3 exercices indépendents. Dans chaque exercice, les questions sont “relativement” indépendentes : vous n'êtes pas obligé d'avoir répondu à la question *i* pour répondre à la question *i+1*.

Le barême, sur 25 points, est donné à titre indicatif. La note d'examen correspondra au minimum entre le nombre de points que vous aurez obtenu par vos bonnes réponses et 20.

Exercice 1. Analyse statique de programmes Lustre (4 points)

On considère les noeuds Lustre suivants :

```
node EA1(a,b : bool) returns (s1,s2 : bool);
var x : bool;
let
        s1 = a and b;
        x = current (a when b);
        s2 = x and pre b;
tel
node EA2(a,b : bool) returns (s1,s2 : bool);
var x : bool;
let
        s1 = if b then a and x else b;
        x = a and s2;
        s2 = if not b then a and s1 else a;
tel
```

Question 1 (1 point) : Le noeud `EA1` est-il syntaxiquement correct ? Dans la négative, expliquez le problème syntaxique posé et proposez une correction. Dans l'affirmative, donnez le résultat de l'exécution du noeud sur laséquence d'entrée suivante :

(***utilisez la feuille réponse fournie en fin de sujet***)

tick :	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
a	F	F	T	T	F	T	T	T	F	T	T	T	F	F	T	T	F	T
b	F	F	F	T	T	F	T	F	F	T	T	F	F	T	F	T	T	T

Question 2 (3 points) : Le noeud EA2 est rejetté par le compilateur Lustre-v4. Expliquez le problème posé et proposez une correction. Le programme serait-il rejetté si le compilateur Lustre effectuait une analyse de causalité constructive plutôt que purement structurelle ? Justifiez précisément votre réponse.

Exercice 2. Lustre : Modélisation et vérification (14 points)

On souhaite modéliser la commande d'un clignotant d'automobile. Le clignotant comprend deux feux situés à gauche et deux feux situés à droite, commandés respectivement par les signaux `left` et `right`. Si `left = 1`, les deux feux de gauche clignotent et si `right = 1`, les deux feux de droite clignotent.

Le clignotant est activé à l'aide d'une manette à trois positions : basse, intermédiaire, haute. Lorsque la manette est en position haute, les feux gauches clignotent; lorsque la manette est en position basse, les feux droits clignotent. En position intermédiaire, aucun feu ne clignote.

Le changement de position de la manette est actionné par deux commandes exclusives `up` et `down` : chaque `up` impose (quand cela est possible) un mouvement de un cran vers le haut et chaque `down` un mouvement de un cran vers le bas. Le signal `reset` place la manette en position intermédiaire, quelquesoient les valeurs des signaux `up` et `down`.

La spécification du noeud commandant le clignotant est la suivante :

```
node cde_clignot(reset : bool; up, down : bool) returns(left, right : bool);
```

Question 1 (3 points) : Programmez en Lustre le noeud `cde_clignot`.

Question 2 (0.5 point) : Les commandes `up` et `down` ne peuvent pas être actives simultanément. Modifiez le code du noeud `cde_clignot` pour prendre en compte cette hypothèse.

Question 3 (0.5 point) : Donnez le chronogramme associé au noeud pour la séquence d'entrée suivante : (***utilisez la feuille réponse fournie en fin de sujet***)

tick :	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
`reset`	F	T	F	F	F	F	F	F	F	T	F	T	F	F	F	F	F	F
`up`	F	F	T	T	F	F	F	F	F	F	F	T	F	T	T	F	T	T
`down`	F	F	F	F	F	T	F	T	F	T	T	F	T	F	F	F	F	F

Question 4 (5 points) : Ecrivez un noeud observateur pour chacune de propriétés suivantes :

- Les feux gauches et droits ne clignotent pas simultanément
- Lors d'une réinitialisation, aucun feu ne clignote
- Une commande `up` désactive le clignotement des feux droits si il n'y a pas de

réinitialisation.

- S'il n'y a pas de réinitialisation et que les feux gauches ne clignotent pas, une commande `down` active le clignotement des feux droits.
- S'il n'y a pas de réinitialisation, après deux commandes `down` successives, les feux droits clignotent.

En pratique, le clignotant est désactivé par une action sur la manette (cf question précédente) ou alors par une combinaison de mouvements du volant :

a) Lorsque le clignotant gauche est activé, une rotation du volant d'un angle au moins égal à α_{min} vers la gauche suivie d'une rotation vers la droite (même minime) arrête le clignotement.

b) Lorsque le clignotant droit est activé, une rotation du volant d'un angle au moins égal à α_{min} vers la droite suivie d'une rotation vers la gauche (même minime) arrête le clignotement.

c) Lorsqu'il n'y a pas de clignotement, le mouvement du volant n'est pas mesuré.

La mesure du mouvement du volant n'a lieu que lorsque le clignotant est activé. Cette activation est matérialisée par deux signaux `debut` et `fin`. L'occurrence d'un front montant d'un signal `debut` déclenche la prise de mesure du mouvement du volant et un front montant sur le signal `fin` achève cette mesure. Pendant la mesure, l'indicateur `g_d` indique si ce sont les feux gauches ou droits qui clignotent. Arbitrairement, `g_d = 0` désigne un clignotement gauche et `g_d = 1` un clignotement droit.

On considérera que la rotation instantanée du volant est matérialisée par une vitesse angulaire θ, exprimée en degré par top d'horloge. Par convention, cette vitesse est négative pour la rotation à gauche et positive pour la rotation à droite. La mesure de l'angle de rotation courante α consiste à sommer les vitesses angulaires instantanées à chaque top.

Le noeud Lustre `volant` prend en entrée la vitesse angulaire θ , les signaux `debut` et `fin` et l' indicateur `g_d` déterminant le sens du clignotement. Ce noeud calcule à chaque top l'angle de rotation courante du volant α, et positionne le signal `desactive` à 1 si une des deux conditions précédentes (a) ou (b) est vérifiée. Une fois positionné à 1, le signal `desactive` reste à 1 pour 1 top uniquement.

L'interface du noeud volant est donnée ci-après :

```
node volant(theta : int; debut,fin,g_d : bool) returns (desactive : bool);
```

Question 5 (3 points): Programmez le corps du noeud `volant`.

Question 6 (2 points) : Associez les noeuds `cde_clignotant` et `volant` pour activer et désactiver les

clignotants. (indication : vous construirez un nouveau noeud instanciant `cde_clignot` et `volant`; la désactivation produite par le noeud `volant` réinitialise le noeud `cde_clignotant` et un clignotant non actif clôt la mesure de rotation du volant). Vous pouvez introduire des entrées et variables internes supplémentaires en précisant leur rôle.

Exercice 3. Esterel : Compilation en automate et vérification (7 points)

On considère le programme Esterel suivant, composé d'un unique module M1 :

```
module M1 :
      input E1, E2;
      output S1, S2, S3, S4;

      loop
            emit S1;
            present E2 then emit S3 else emit S4 end present;
            abort
                  emit S2;
                  await E1;
                  emit S3;
            when E2 do emit S4 end abort;
            await E1;
      end loop
end module
```

Question 1 (1 point) : Représentez l'exécution du programme lorsque celui-ciest soumis à la séquence d'entrée suivante ***sur la feuille réponse fournie en fin de sujet*** :

Tick :	1	2	3	4	5	6	7	8
Entrées	E2	E1	E1	E1 E2		E2	E1	
Sorties								

Question 2 (3 points) : Identifiez les points d'arrêt du programme et construisez la machine de Mealy du programme agglomérant en une seule transition toutes les actions exécutées entre deux points d'arrêt. *Vous vous assurerez que l'automate construit est complet et déterministe.*

Question 3 (3 points) : le module M1 vérifie-t-il les propriétés suivantes (vous justifierez précisement la validité de chaque propriété, à partir du code Esterel ou de la structure de l'automate).

- E1 et S3 sont toujours actifs simultanément
- S3 et S4 ne sont jamais actifs simultanément

- S4 finira toujours par être émis
- S3 pourra être émis plus tard
- S2 est forcément suivi, au cycle d'après, par S3
- S2 sera forcément suivi, plus tard (mais pas forcément au cycle d'après), par S3

UPMC -- Master Informatique 15 novembre 2010
Parcours “Systèmes et Applications Répartis” (SAR)
Module “Langages Synchrones”
Année universitaire 2010-2011
E. Encrenaz

Examen – feuille réponse

NOM : ..... PRENOM : .....

Exercice 1, Question 1 : Chronogramme.

Chaque case indique la valeur de vérité du signal d'entrée ou de sortie ou d'un signal interne.

tick :	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
a	F	F	T	T	F	T	T	T	F	T	T	T	F	F	T	T	F	T
b	F	F	F	T	T	F	T	F	F	T	T	F	F	T	F	T	T	T
x																		
s1																		
s2																		

Exercice 2, Question 3 : Chronogramme.

Chaque case indique la valeur de vérité du signal d'entrée ou de sortie.

tick :	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
reset	F	T	F	F	F	F	F	F	F	T	F	T	F	F	F	F	F	F
up	F	F	T	T	F	F	F	F	F	F	F	T	F	T	T	F	T	T
down	F	F	F	F	F	T	F	T	F	T	T	F	T	F	F	F	F	
left																		
right																		

Exercice 3, Question 1 : Exécution.

Chaque case indique un événement : les signaux présents (en entrée), ou émis (en sortie).

Tick :	1	2	3	4	5	6	7	8
Entrées	E2	E1	E1	E1 E2		E2	E1	
Sorties								

Réponses

EX1, Q1

le noeud EA1 est syntaxiquement correct.

tick :	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
a	F	F	T	T	F	T	T	T	F	T	T	T	F	F	T	T	F	T
b	F	F	F	T	T	F	T	F	F	T	T	F	F	T	F	T	T	T
x	X	X	X	T	F	F	T	T	T	T	T	T	T	F	F	T	F	T
s1	F	F	F	T	F	F	T	F	F	T	T	F	F	F	F	T	F	T
s2	X	F	F	F	F	F	F	T	F	F	T	T	F	F	F	F	F	T

EX2,Q2

Le noeud EA2 n'est pas syntaxiquement correct : présence d'une boucle combinatoire entre les signaux `s1, x et s2`.

Le système d'équations associés au noeud est le suivant :

```
s1  = b.a.x
x = a.s2
s2 = !b.a.s1 + b.a
```

Ce système respecte la causalité booléenne et même la causalité constructive.

EX2,Q1

```
node cde_clignot(reset : bool; up, down : bool) returns(left, right : bool);
var  state : int;
let
     state = 0 -> if reset then 0
                    else if (up and (pre(state) < 1)) then (pre(state) +1)
                  else if (down and (pre(state) > -1)) then (pre(state) -1)
                  else pre(state);
     left = (state = 1);
     right = (state = -1);
     --assert(not (up and down));
tel;
```

EX2,Q3

Chaque case indique la valeur de vérité du signal d'entrée ou de sortie.

tick :	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
reset	F	T	F	F	F	F	F	F	F	T	F	T	F	F	F	F	F	F
up	F	F	T	T	F	F	F	F	F	F	F	T	F	T	T	F	T	T
down	F	F	F	F	F	T	F	T	F	T	T	F	T	F	F	F	F	F
left	F	F	T	T	T	F	F	F	F	F	F	F	F	F	T	T	T	T
right	F	F	F	F	F	F	F	T	T	F	T	F	T	F	F	F	F	F

EX2, Q4

```
-- prop exclusive : not (left and right)
-- prop reset : reset -> not (left or right)
-- prop desactive_R : pre(right) and up and not reset -> not right
-- prop active_R : not pre(left) and down and not reset -> right
-- prop active_R_2 : pre(pre down) and not pre(pre reset) and pre(down) and not
pre(reset) -> right
```

EX2,Q5

```
node volant(theta : int; debut,fin,g_d : bool) returns (desactive : bool);
var alpha : int;
    acquisition : bool;
let
    acquisition = false ->   if (debut and not pre debut) then 1
                             else if pre(acquisition) and not fin then 1
                             else if (fin and not pre fin) then 0
                             else pre(acquisition);

    alpha = theta -> if not acquisition then 0 else pre(alpha) + theta;
    desactive = false -> (acquisition) and
                         ((g_d and alpha > 20 and theta < 0) or
                          (not g_d and alpha < 0 and alpha < -20 and theta > 0));
tel;
3 points : 1 pour alpha et 2 pour desactive
```

EX2,Q6

```
J'ai introduit deux entrées vol_g et vol_d pour représenter le sens mouvement du
volant (vers la gauche ou la droite) et la variable interne theta fixée à +/- 8
selon le sens de rotation du volant. Pleft et pright représentent left et right à
l'instant précédent.
```

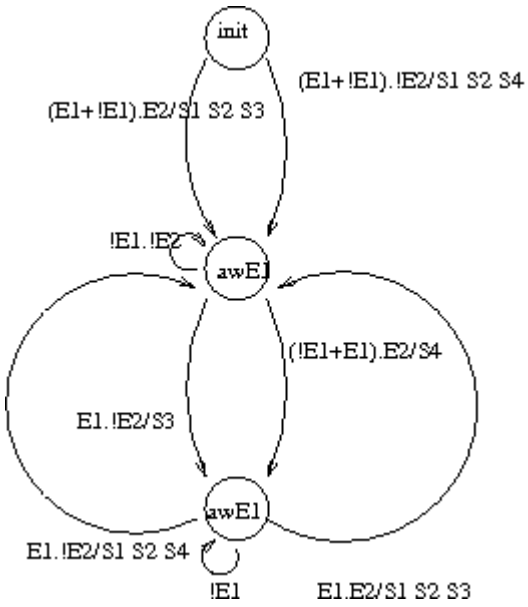
```
node sys(reset, up, down, vol_g, vol_d: bool) returns (left,right,desac : bool);
var theta : int;
    pleft, pright : bool;

let
        pleft = 0 -> pre left;
        pright = 0 ->  pre right;
        theta = if vol_d then 8 else if vol_g then  -8 else 0;
        desac = 0 -> volant( theta,
                             ((pleft and not pre(pleft)) or (pright and not
pre(pright)),
                             pright, not (pleft or pright));
        (left, right) = cde_clignot(desac,up,down);
tel;
```

EX3, Q1

Tick :	1	2	3	4	5	6	7	8
Entrées	E2	E1	E1	E1 E2		E2	E1	
Sorties	S1 S3 S2	S3	S1 S4 S2	S4			S1 S4 S2	

EX3, Q2

l'automate complet et déterministe du programme :



EX3, Q3

- E1 et S3 sont toujours actifs simultanément
  - non (cf chronogramme précédent)
- S3 et S4 ne sont jamais actifs simultanément
  - ils ne sont jamais actifs simultanément : sur chaque transition, on a soit S3 soit S4 soit aucune des deux mais jamais les deux.
- S4 finira toujours par être émis
  - non : boucle (!E2/S3 -> E1 E2/S1 S2 S3)* ou (!E1!E2)* de awE1_1, ou même (!E1)* de awE1_2
- S3 pourra être émis plus tard
  - oui des deux états, on il y a une transition sortante emettant S3
- S2 est forcément suivi, au cycle d'après, par S3
  - non : cf. chronogramme
- S2 sera forcément suivi, plus tard (mais pas forcément au cycle d'après), par S3
  - non : boucle (E2/S4 -> E1.!E2/S1 S4 S2)*