ENSERB
Année 1999-2000

Filière Informatique 3^{eme} année
Option Génie Logiciel

Vendredi 17 mars, 14 heures-16 heures
Tous documents autorisés.

Ingénierie Systèmes Critiques
Alain Griffault

SUJET + CORRIGE

Avertissement

Les deux exercices sont totalement indépendant. L'ensemble est long, mais le total des points possibles est de 24.

1 Vérifications formelles

Un système de transition est le support d'un jeu à deux joueurs A et B avec la règle: *le joueur qui ne peut pas jouer a perdu.*

Nous disposons de la fonction au point fixe suivante:

```
\* celui qui ne peux pas jouer a perdu - le dernier qui joue gagne *\
function last_win(termine:state; coupX:trans; coupY:trans)
         return Xgagnant:state;
var Yperdant:state
begin
Xgagnant = src(rtgt(Yperdant) /\ coupX);
Yperdant = termine \/ (src(rtgt(Xgagnant) /\ coupY) - src(rtgt(*-Xgagnant) /\ coupY))
end.
```

que nous utilisons de la façon suivante:

```
coupB := ; \\ un calcul pour préciser les transitions que le joueur B peut effectuer
coupA := ; \\ un calcul pour préciser les transitions que le joueur A peut effectuer
Agagnant := last_win(dead,coupA,coupB);
Bgagnant := last_win(dead,coupB,coupA);
AwinPremierJoueur := Agagnant /\ initial;
AwinSecondJoueur := tgt(rsrc(initial) /\ coupB) - Agagnant;
BwinPremierJoueur := Bgagnant /\ initial;
BwinSecondJoueur := tgt(rsrc(initial) /\ coupA) - Bgagnant;
```

Question 1.1 (2 points) *Rappelez brièvement ce que calculent les différentes expressions précédentes, et comment ces résultats doivent être interprétés.*

Réponse à la question 1.1:

`Agagnant` (resp. `Agagnant`) calcule les états du jeu à partir desquels, si A (resp. B) joue bien, il est certain de gagner pour la règle *le joueur qui ne peut pas jouer a perdu.*

`AwinPremierJoueur` (resp. `BwinPremierJoueur`) calcule les états initiaux du jeu à partir desquels, si A (resp. B) joue bien, il est certain de gagner. Ce calcul permet donc de conclure à la question suivante: *Existe t-il une stratégie gagnante pour A (resp. B) si A (resp. B) est le joueur qui joue le premier?*

`AwinSecondJoueur` (resp. `BwinSecondJoueur`) calcule les états du jeu à partir desquels A (resp. B) joue son premier coup, et même si A (resp. B) joue bien, il n'est pas certain de gagner. Ce calcul permet donc de conclure à la question suivante: *Existe t-il une stratégie gagnante pour A (resp. B) si A (resp. B) n'est pas le joueur qui joue le premier?*

Pour établir une stratégie gagnante, il serait préférable de disposer non pas des situations gagnantes et/ou perdantes, mais des coups qu'il faut jouer pour gagner.

Question 1.2 (3 points) *Donnez les équations d'une nouvelle fonction au point fixe, dont le résultat est l'ensemble des transitions correspondant aux coups gagnant.*

Réponse à la question 1.2:

```
function T_last_win(termine:state; coupX:trans; coupY:trans)
  return Xgagnant:trans;
var Yperdant:trans
begin
Xgagnant = (rtgt(termine) /\ coupX) \/
           ((rtgt(src(Yperdant)) /\ coupX) -
            (rtgt(src(coupY-Yperdant)) /\ coupX));
Yperdant = (rtgt(src(Xgagnant)) /\ coupY)
end.
```

Maintenant qu'il est possible de calculer les coups gagnant pour un joueur (A par exemple), il est possible de calculer les seules *vraies parties* gagnées par A, parties dans lesquelles le joueur A qui possède une stratégie gagnante ne joue effectivement que des coups gagnant; et dans lesquelles l'autre joueur B joue un coup quelconque parmi ceux autorisés.

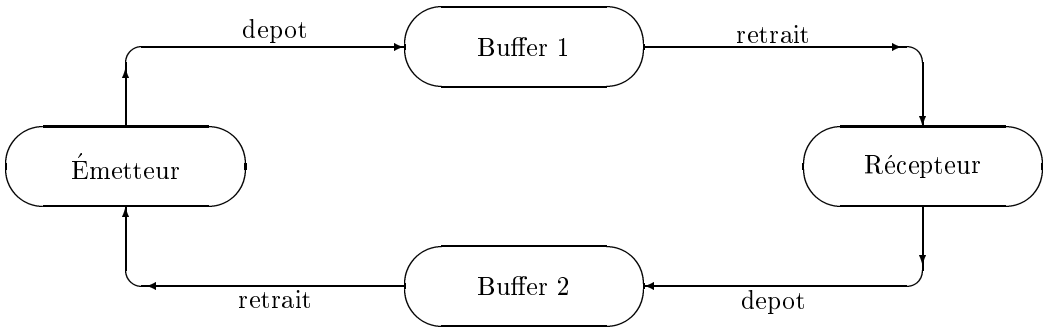
Question 1.3 (3 points) *Donnez les équations d'une nouvelle fonction au point fixe, dont le résultat est l'ensemble des transitions correspondant aux seules* vraies parties *possibles à partir d'une situation initiale gagnante pour le joueur A.*

Réponse à la question 1.3:

```
function VraiesPartiesAGagne(debut:state; AGagnant:trans; BPerdant:trans)
  return Z:trans;
var T:trans;
    U:trans
begin
T = (rsrc(debut)/\AGagnant) \/
    (rsrc(tgt(U))/\AGagnant);
U = (rsrc(tgt(T))/\BPerdant);
Z = T \/ U
end.
```

2 Modélisations formelles

Le but de l'exercice est l'étude des propriétés du protocole du bit alterné en utilisant le système global suivant :



L'émetteur dans ce protocole exécute le code suivant :

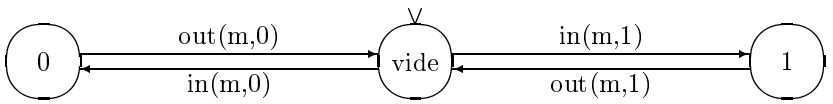
```
parite := true;
while true do begin
              recevoir(message); /* d'un producteur quelconque */
              emettre(message,parite);
              recevoir(ack,ackParite);
              while parite <> ackParite do begin
                                          emettre(message,parite);
                                          recevoir(ack,ackParite);
                                          end;
              parite := not parite;
              end;
```

et le récepteur le code suivant :

```
ackParite := true;
while true do begin
              recevoir(message,parite);
              emettre(ack,parite);
              while parite <> ackParite do begin
                                          recevoir(message,parite);
                                          emettre(ack,parite);
                                          end;
              emettre(message); /* vers le consommateur */
              ackParite := not ackParite;
              end;
```

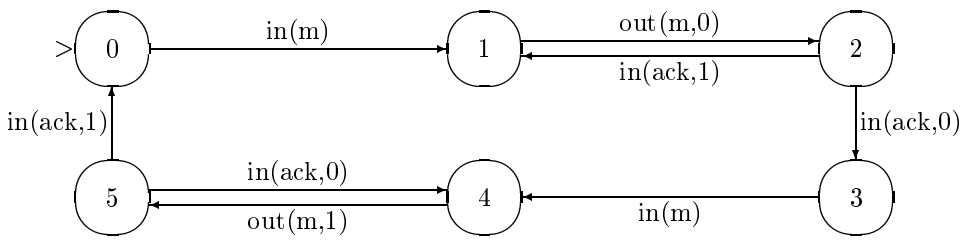
Question 2.1 (1 point) *Dessiner le système de transition des buffers en supposant qu'ils sont de taille 1.*

Réponse à la question 2.1: Dans tous les dessins, les boucles e sur les états ne sont pas mises.



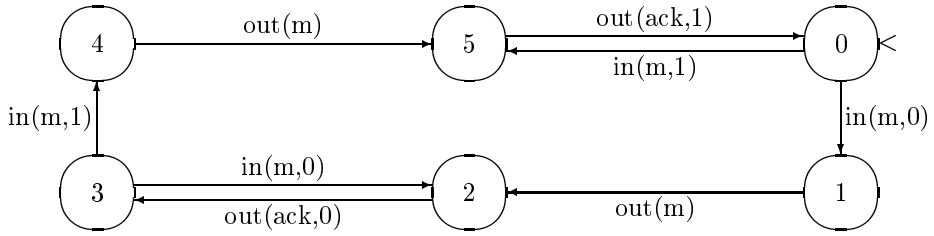
Question 2.2 (2 points) *Dessiner le système de transition de l'émetteur.*

Réponse à la question 2.2:



Question 2.3 (2 points) *Dessiner le système de transition du récepteur.*

Réponse à la question 2.3:



Question 2.4 (2 points) *Donner le système de synchronisation du système global.*

Réponse à la question 2.4:

Émetteur	Buffer 1	Buffer 2	Récepteur
in(m)	e	e	e
out(m,0)	in(m,0)	e	e
out(m,1)	in(m,1)	e	e
in(ack,0)	e	out(m,0)	e
in(ack,1)	e	out(m,1)	e
e	out(m,0)	e	in(m,0)
e	out(m,1)	e	in(m,1)
e	e	in(m,0)	out(ack,0)
e	e	in(m,1)	out(ack,1)
e	e	e	out(m)

Question 2.5 (2 points) *Donner quelques propriétés qui vous semblent pertinentes pour cette étude et leurs interprétations.*

Réponse à la question 2.5:

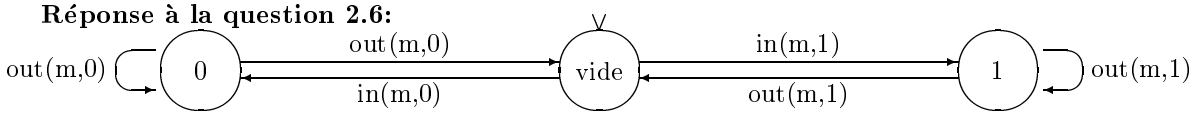
`dead := * - src(*);` calcule les états bloquant du système. Les états bloquant correspondent à une des deux situations suivantes: soit l'émetteur et le récepteur désirent envoyer un message et les buffers sont pleins, soit ils désirent recevoir un message et les buffers sont vides. Pour un protocole comme celui-ci, ou chaque entité émet et reçoit en alternance, cela ne peut provenir que de pertes et/ou de duplications de messages par le réseau.

`cfc := loop(rtgt(initial),*);` calcule les transitions appartenant à la composante fortement connexe passant par la situation initiale. Il est souhaitable pour un protocole que cela corresponde à l'ensemble des transitions.

On désire maintenant vérifier si ce protocole est résistant aux duplications (et même multiplications) de messages par le réseau de communications.

Question 2.6 (2 points) *Dessiner le système de transition d'un buffer de taille 1 qui multiplie des messages.*

Réponse à la question 2.6:



On désire maintenant vérifier si ce protocole est résistant aux pertes de messages par le réseau de communications. Il est possible de modéliser cette perte de message soit dans le comportement du buffer, soit dans les contraintes de synchronisation.

Question 2.7 (2 points) *Donner le système de synchronisation du système global avec possibilités de pertes de messages.*

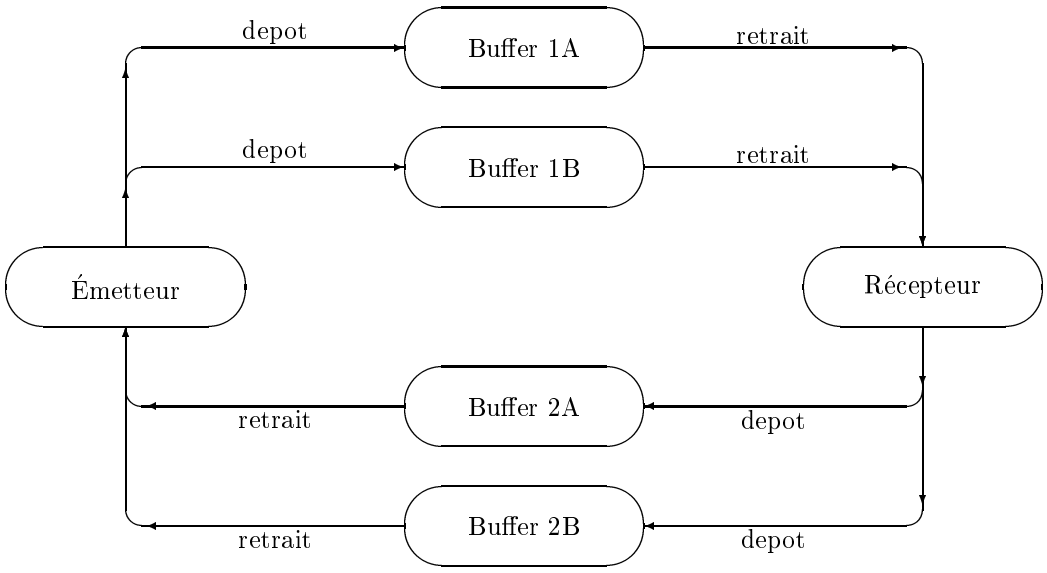
Réponse à la question 2.7: Plusieurs solutions : soit un message envoyé au buffer n'est pas reçu par celui-ci (Buffer 1), soit un message envoyé par le buffer n'est pas reçu par le destinataire (Buffer 2).

Émetteur	Buffer 1	Buffer 2	Récepteur
in(m)	e	e	e
out(m,0)	in(m,0)	e	e
out(m,1)	in(m,1)	e	e
in(ack,0)	e	out(m,0)	e
in(ack,1)	e	out(m,1)	e
e	out(m,0)	e	in(m,0)
e	out(m,1)	e	in(m,1)
e	e	in(m,0)	out(ack,0)
e	e	in(m,1)	out(ack,1)
e	e	e	out(m)
out(m,0)	e	e	e
out(m,1)	e	e	e
e	e	out(m,0)	e
e	e	out(m,1)	e

On désire maintenant vérifier si ce protocole est résistant aux déséquencements (l'ordre des réceptions n'est pas forcément identique à l'ordre des émissions) de messages par le réseau de communications.

Question 2.8 (1 point) *Donnez une nouvelle architecture globale afin de pouvoir traiter les déséquencements.*

Réponse à la question 2.8:



Question 2.9 (2 points) *Si l'on combine dans un même modèle, la perte, la duplication et le déséquencement de messages, pensez-vous vos calculs et vos interprétations de la question 2.5 soient toujours valables. Justifiez.*

Réponse à la question 2.9: Les interprétations des calculs ne sont plus valables. Par exemple, si le buffer 1A perd son message, mais que le buffer 1B duplique à ce même instant un message ayant la même parité, il n'y aura ni blocage, ni perturbation du service. Les deux erreurs "s'annulant".