Université Bordeaux 1

Année : 2011/2012

Parcours : Master 1 Informatique
UE : J1IN8W14 : Conceptions Formelles
Date : Vendredi 20 avril 2012
Heure : 8h 30
Durée conseillée : 1 heure 30
Documents : autorisés
Epreuve de M. Alain Griffault

U. F. R.
Mathématique
Informatique

Code d'anonymat :

Avertissement

- La plupart des questions sont indépendantes.
- Le barème total est de 22 points car le sujet est assez long.
- L'espace laissé pour les réponses est suffisant (sauf si vous utilisez ces feuilles comme brouillon, ce qui est fortement déconseillé).

Le protocole du bit alterné (22 points)

L'objectif de l'étude est de vérifier quelques propriétés du protocole du bit alterné.
Le protocole du bit alterné (ABP) est un protocole bien connu pour transférer des données d'un producteur (*a Producer*) vers un consommateur (*a Consumer*). Le protocole utilise un émetteur (*a Sender*) et un récepteur (*a Receiver*) dialoguant à travers le réseau. Pour contrôler la validité du transfert, le protocole ABP utilise un *bit* de parité comme accusé de réception.

Algorithm 1 `Sender()`

```
1: parity ← true
2: while (true) do
3:     m ← input(Producer)
4:     repeat
5:         send(Receiver, m, parity)
6:         ack ← receive(Receiver)
7:     until parity = ack
8:     parity ← (¬parity)
9: end while
```

Algorithm 2 `Receiver()`

```
1: ack ← true
2: while (true) do
3:     repeat
4:         (m, parity) ← receive(Sender)
5:         send(Sender, parity)
6:     until parity = ack
7:     output(Consumer, m)
8:     ack ← (¬ack)
9: end while
```

En fait, ABP ne tient pas compte de la valeur du message. Seules les deux variables *parity* sont importantes. De ce fait, le message transmis n'est pas représenté dans la suite de cette étude.
L'approche suivie pour la modélisation et la vérification de ABP est la suivante :

1. Modélisation et validation de l'émetteur et du récepteur.
2. Modélisation et validation du protocole ABP avec des files d'attentes parfaites et bornées.
3. Vérification de quelques exigences.
4. Modélisation, validation et vérification du protocole ABP en représentant le réseau par des files d'attentes imparfaites et bornées.

Exercice 1 (Modélisation et validation de l'émetteur et du récepteur (4 points))
Une description AltaRica *de l'émetteur est :*

```
node Sender
  flow  f_ack : bool;            // the acknowledge sent by the receiver
  state parity : bool : public; // the expected parity
        ack : bool : public;     // to store the ack
        co : [0,3];              // ordinal counter of the program
  init  parity := true, ack := true, co := 0;
  event input, send, receive;
  trans
    co = 0                  |- input -> co := 1;
    co = 1                  |- send -> co := 2;
    co = 2                  |- receive -> ack := f_ack, co := 3;
    co = 3 & ack != parity  |- send -> co := 2;
    co = 3 & ack = parity   |- input -> parity:= ~parity, co := 1;
edon
```

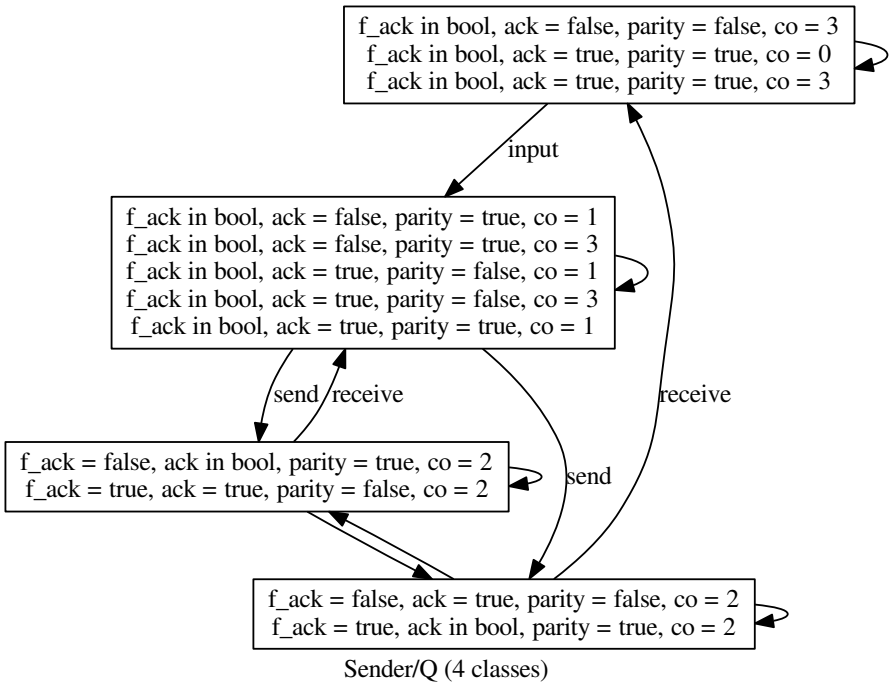

Figure 1 – Le graphe quotient de la sémantique de l'émetteur

Question 1.1 (2 points) *Donner les correspondances entre l'algorithme, le code* AltaRica*, et la sémantique.*

Une description AltaRica *du récepteur est :*

```
node Receiver
  flow f_parity : bool;          // the parity sent by the Sender
  state parity : bool : public; // to store the receive parity
        ack : bool : public;    // the expected parity value
        co : [0,2];             // ordinal counter of the program
  init  ack := true, co := 0;
  event output, send, receive;
  trans
    co = 0                  |- receive -> parity := f_parity, co := 1;
    co = 1 & ack = parity   |- send -> co := 2;
    co = 1 & ack != parity  |- send -> co := 0;
    co = 2                  |- output -> ack := ~ack, co := 0;
edon
```

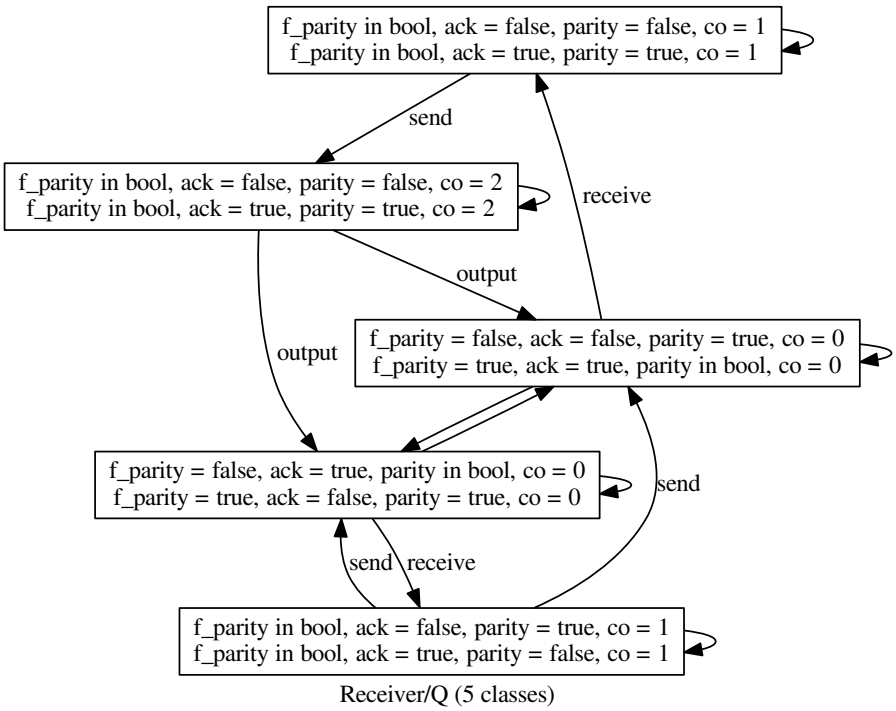

Figure 2 – Le graphe quotient de la sémantique du récepteur

Question 1.2 (2 points) *Donner les correspondances entre l'algorithme, le code* AltaRica*, et la sémantique.*

Exercice 2 (Modélisation et validation du protocole ABP avec un réseau parfait (4 points))

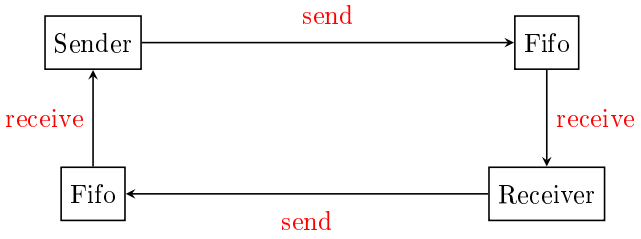

FIGURE 3 – ABP avec un réseau de deux fifos

Nous avons besoin d'une file parfaite et bornée pouvant mémoriser un booléen. La sémantique espérée d'un telle file est donnée par le graphe suivant.

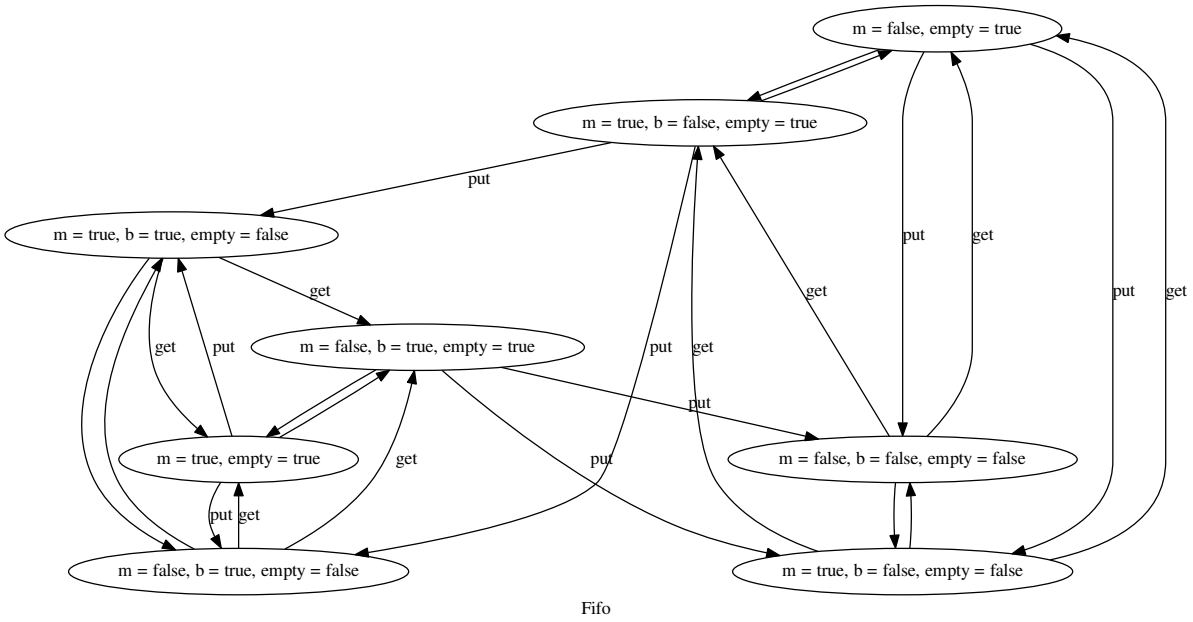

FIGURE 4 – Le graphe de la sémantique d'une file

Question 2.1 (2 points) *Compléter le code* ALTARICA *de* `Fifo` *afin d'obtenir la sémantique souhaitée.*

```
node Fifo
  flow  m : bool;          // La valeur a enregistrer
  state b : bool : public;
        empty : bool;
  init  empty := true;
  event put, get;
  trans

                    |- put ->                              ;

                    |- get ->                              ;

edon
```

La figure 3 décrit le système global. Le graphe suivant donne la sémantique souhaitée.

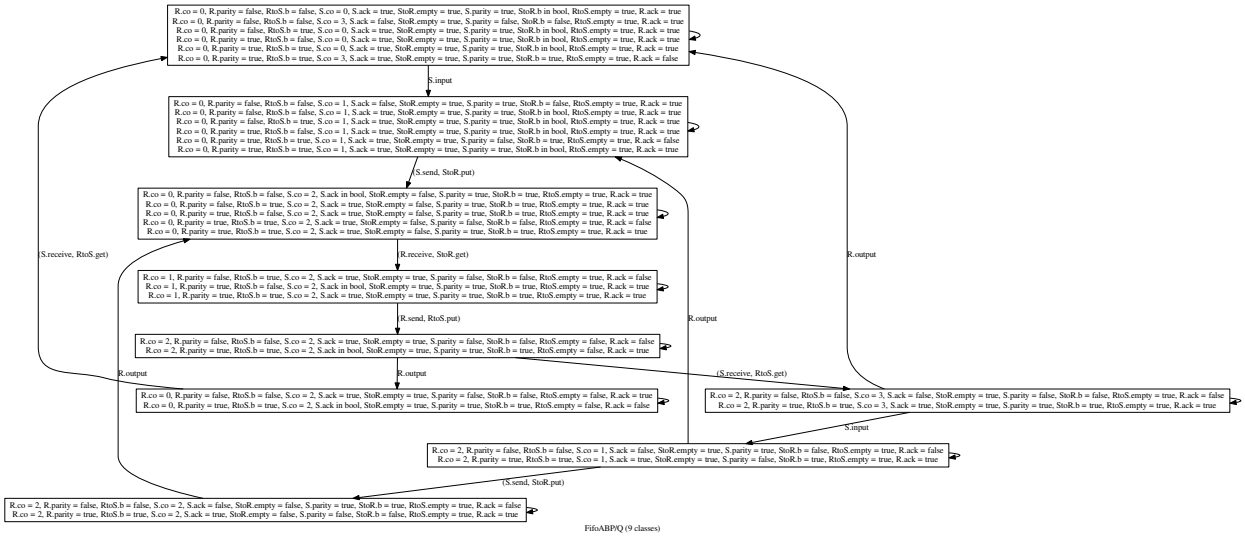

FIGURE 5 – Le graphe quotient de la sémantique du système FifoABP

Question 2.2 (2 points) *Compléter le code* ALTARICA *de* ***FifoABP*** *afin d'obtenir la sémantique souhaitée.*

```
node FifoABP
  sub S : Sender;
      R : Receiver;
      StoR, RtoS : Fifo;
  assert
    // Liens entre les variables d'etats et de flux pour transmettre une valeur




  sync
    // Quand un composant envoie un message, un autre le recoit




edon
```

Exercice 3 (Vérification de quelques exigences. (6 points))
Nous souhaitons vérifier que ABP transmet correctement les messages du producteur au consommateur. Pour cela, nous devons vérifier que :
– *le protocole ne meurt jamais,*
– *il n'y a pas de perte de message,*
– *il n'y a pas de corruption de message,*
– *il n'y a pas de duplication de message,*
– *les messages sont reçus dans l'ordre d'émission.*

Question 3.1 (2 points) *Expliquer les calculs suivants :*

```
with FifoABP, FifoErrorABP, Fifo2ErrorsABP do
  // Usefull computations
  Input := label S.input;
  Ssend := label S.send & rsrc([S.co=1]);
  Sresend := label S.send & rsrc([S.co=3]);
  RreceiveOK := label R.receive & rtgt([R.ack = R.parity]);
  RreceiveKO := label R.receive & rtgt([R.ack != R.parity]);
  Output := label R.output;
done
```

Question 3.2 (2 points) *Expliquer les calculs suivants :*

```
// Asynchronous models of ABP
with FifoABP, FifoErrorABP, Fifo2ErrorsABP do
  // dead states.
  dead := any_s - src(any_t -self_epsilon);
  // P1 : between two Input, there is at least one Ssend
  //  and between two Ssend, there is at least one Input.
  notP1 := (reach(tgt(Input), any_t - Ssend) & src(Input))|
           (reach(tgt(Ssend), any_t - Input) & src(Ssend));
  // P2 : between two Ssend, there is at least one RreceiveOK
  //  and between two RreceiveOK, there is at least one Ssend.
  notP2 := (reach(tgt(Ssend), any_t - RreceiveOK) & src(Ssend))|
           (reach(tgt(RreceiveOK), any_t - Ssend) & src(RreceiveOK));
  // P3 : between two RreceiveOK, there is at least one Output
  //  and between two Output, there is at least one RreceiveOK.
  notP3 := (reach(tgt(RreceiveOK), any_t - Output) & src(RreceiveOK))|
           (reach(tgt(Output), any_t - RreceiveOK) & src(Output));
done
```

Question 3.3 (2 points) *Modifier les calculs suivants afin d'obtenir les résultats suivants.*

```
/*
 * Properties for node : FifoABP
 * # state properties : 1
 *
 * any_s = 42
 *
 * # trans properties : 1
 *
 * any_t = 91
 */
TEST(dead,0)     [PASSED]
TEST(notP1,0)    [PASSED]
TEST(notP2,0)    [PASSED]
TEST(notP3,0)    [PASSED]
TEST(notQ1,0)    [PASSED]
TEST(notQ2,0)    [PASSED]
TEST(notQ3,0)    [PASSED]


// Asynchronous models of ABP
with FifoABP, FifoErrorABP, Fifo2ErrorsABP do
  // Q1 : the first Input is before the first Ssend
  notQ1 :=









  // Q2 : the first Ssend is before the first RreceiveOK
  notQ2 :=









  // Q3 : the first RreceiveOK is before the first Output
  notQ3 :=









done
```

Exercice 4 (Modélisation et vérification du protocole ABP avec un réseau imparfait. (8 points))

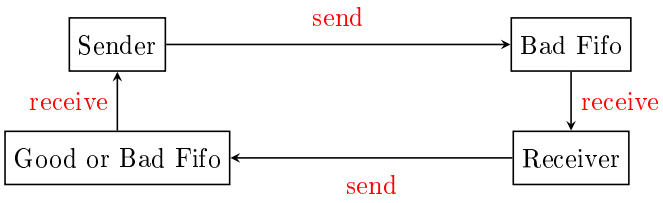

FIGURE 6 – ABP avec une ou deux fifos imparfaites

Pour cet exercice, nous utilisons une file imparfaite qui peut corrompre la valeur de ses données. Le graphe suivant donne la sémantique attendue d'une telle file.

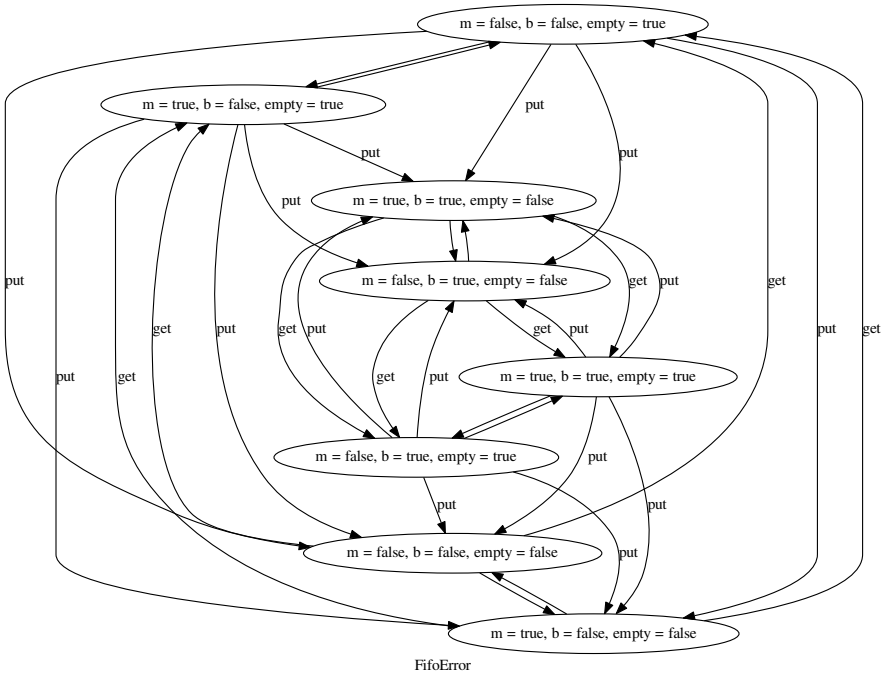

FIGURE 7 – Le graphe de la sémantique d'une file imparfaite

Question 4.1 (2 points) *Compléter le code* ALTARICA *de* `FifoError` *afin d'obtenir la sémantique souhaitée.*

```
node FifoError
  // La file peut transmettre un message corrompu
  flow  m : bool;
  state b : bool : public;
        empty : bool;
  init  empty := true;
  event put, get;
  trans




edon
```

Question 4.2 (2 points) *Adapter le code* AltaRica *des systèmes* `FifoErrorABP` *et* `Fifo2ErrorsABP`.

```
node FifoErrorABP
  sub S : Sender;
      R : Receiver;
      StoR : FifoError;
      RtoS : Fifo;
  assert




  sync







edon

node Fifo2ErrorsABP
  sub S : Sender;
      R : Receiver;
      StoR : FifoError;
      RtoS : FifoError;
  assert




  sync







edon
```

Si vous avez correctement adapté les codes AltaRica *et les spécifications, vous devez obtenir :*

```
/*
 * Properties for node : FifoErrorABP
 * # state properties : 1
 *
 * any_s = 59
 *
 * # trans properties : 1
 *
 * any_t = 139
 */
TEST(dead,0)     [PASSED]
TEST(notP1,0)    [PASSED]
TEST(notP2,0)    [PASSED]
TEST(notP3,0)    [PASSED]
TEST(notQ1,0)    [PASSED]
TEST(notQ2,0)    [PASSED]
TEST(notQ3,0)    [PASSED]
```

```
/*
 * Properties for node : Fifo2ErrorsABP
 * # state properties : 1
 *
 * any_s = 98
 *
 * # trans properties : 1
 *
 * any_t = 254
 */
TEST(dead,0)     [PASSED]
TEST(notP1,0)    [PASSED]
TEST(notP2,0)    [FAILED] actual size = 8
TEST(notP3,0)    [PASSED]
TEST(notQ1,0)    [PASSED]
TEST(notQ2,0)    [PASSED]
TEST(notQ3,0)    [PASSED]
```

Question 4.3 (2 points) *Donner les calculs permettant d'obtenir le graphe suivant.*

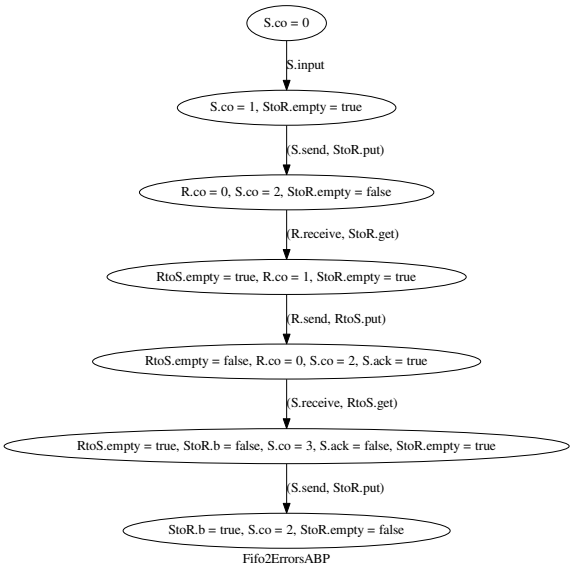

FIGURE 8 – Fifo2ErrorsABP ne satisfait pas P2

```
// Asynchronous models of ABP
with Fifo2ErrorsABP do




done
```

Question 4.4 (2 points) *Expliquer pourquoi* `Fifo2ErrorsABP` *ne satisfait pas* `P2`