

Examen ACCOV du 20 Juin 2002 : Correction

Question 1

La transition t_1 correspond à une demande de rendez-vous d'un site x ; la transition t_2 modélise la prise en compte de l'acceptation du rendez-vous par le serveur entre un site x et un site y ; la transition t_{serveur} abstrait le comportement du serveur qui réalise un rendez-vous dès qu'au moins deux sites x et y l'ont demandé; dans ce cas, le serveur signale cette acceptation aux sites x et y et précise avec qui le rendez-vous aura lieu en déposant deux marques dans la place Ack (la marque (x, y) signifie que x peut avoir un rendez-vous avec y ce qui permettra de débloquent x qui est en attente).

Par exemple, un rendez-vous réussi peut être obtenu par le franchissement de la transition t_1 pour $x = 1$ puis par le franchissement de t_1 pour $x = 2$ puis par le franchissement de t_{serveur} pour $x = 1$ et $y = 2$ et enfin par le franchissement de t_2 pour $(x, y) = (1, 2)$ puis pour $(x, y) = (2, 1)$.

Question 2

On a ici effectué une post-agglomération entre t_2 et t_3 .

Question 3

Pour démontrer l'absence d'interblocage il suffit de montrer que l'on a toujours au moins une transition franchissable. Pour raisonner simplement, il suffit de constater que les marques initialement toutes dans la place Repos vont se répartir entre les places Repos, Mess et Ack.

Si ni t_1 ni t_{serveur} ne sont franchissables, d'après leur pré-conditions et la remarque précédente, c'est que toutes les marques (sauf peut être une qui est dans Mess) sont dans la place Ack; si $N \geq 2$ il y a au moins une marque dans la place Ack ce qui rend la transition t_2 franchissable.

Dans le cas où ni t_2 ni t_{serveur} ne sont franchissables, par un raisonnement similaire, il y a au moins une marque dans Repos ce qui rend t_1 franchissable.

En fin, si ni t_1 ni t_2 ne sont franchissables, toutes les marques sont dans Mess qui contient alors au moins deux marques et rend la transition t_{serveur} franchissable.

Puisque le réseau réduit est équivalent au modèle initial pour les propriétés de bases on en conclut qu'il n'y a pas d'interblocage dans le premier modèle.

Question 4

Comme précédemment, les marques initialement dans Repos ne peuvent être par la suite que réparties entre les places Repos, Mess, Ack et Attente_Interne. On remarque également que le nombre de marques dans la place Attente_Interne est toujours égal au nombre de marques dans Id_En_Cours plus le nombre de marques dans Id_En_Attente et que le nombre de marque dans Id_En_Attente vaut 0 ou 1 (à cause de l'alternance de la marque dans Premier). De plus, il

est clair que la marque dans la place `Premier` vaut `True` si et seulement si le nombre de marque dans la place `Id_En_Attente` est pair ou nul; sinon cette marque vaut `False` (lorsqu'il y a un nombre impair de marque dans la place `Id_En_Attente`).

Si il y a au moins une marque dans la place `Repos` ou la place `Ack` alors soit `t1` soit `t2` est franchissable. Dans le cas contraire, toutes les marques sont répartis entre les places `Mess` et `Attente_Interne`.

Si le nombre de marque de `Attente_Interne` est nul, alors le nombre de marque dans `Id_En_Attente` est aussi nul et donc la marque dans la place `Premier` vaut `True`. Puisqu'il y a $N > 0$ marques dans la place `Mess`, la transition `te1` est franchissable.

Si maintenant le nombre de marque de `Attente_Interne` est non nul, soit le nombre de marques dans `Id_En_Cours` est non nul, et dans ce cas la transition `ts1` est franchissable; soit le le nombre de marques dans `Id_En_Cours` est nul et alors nécessairement le nombre de marques dans `Attente_Interne` vaut 1 et donc la marque dans `Premier` vaut `False` et donc `te2` est franchissable (car $N \geq 2$ et donc il y a au moins une marque dans `Mess`).

On conclut donc sur l'absence d'interblocage.

Question 5

Le problème qui se pose est le problème de cohérence des rendez-vous; en effet, prenons la séquence suivante: `te1(x=1)`, `te2(y=2)`, `te1(x=3)`, `te2(y=4)` `ts1(x=1, y=4)` on obtient les marques (2, 1) et (1,4) dans la place `Ack` ce qui signifie que 2 pense être en rendez-vous avec 1 alors que 1 pense être en rendez-vous avec 4.

Si l'on implante le serveur par un objet protégé, les transitions `te1` et `te2` par une entrée externe et la transition `ts1` par une entrée interne alors la sémantique de l'objet protégé fait que la transition `te1` ne peut être franchit dès lors que la transition `ts1` l'est. Les seules séquences possibles sont alors du genre `te1.te2.ts1` ce qui ne peut conduire à une incohérence sur les rendez-vous.

Question 6

```

separate (Prog1)
protected body Serveur is
  entry Demande(X: in Id; Y: out Id) when true is
  begin
    if Premier then
      Premier := False;
      Id_En_Attente := X;
      requeue Attente_Interne;
    else
      Couple_Possible := True;
      Id_En_Cours := X;
      Premier := True;
      Y := Id_En_Attente;
    end if;
  end Demande;
  entry Attente_Interne(X: in Id; Y: out Id) when Couple_Possible is
  begin
    Couple_Possible := False;
    Y := Id_En_Cours;
  end Attente_Interne;
end Serveur;

```

Question 7

Les deux types d'interblocage sont :

- un site x veut un rendez-vous avec un site y alors que y veut un rendez-vous avec z et z avec x ; on a ici un interblocage circulaire;
- tous les sites veulent se mettre en attente de rendez-vous avec n'importe quel autre site (franchissement de la transition τ_4). Dans ce cas, plus aucun site ne peut demander explicitement un rendez-vous et il y a un interblocage.

Question 8

Le serveur doit cette fois gérer deux types de sites : ceux voulant servir un rendez-vous quelconque (sites "serveurs") et ceux voulant un rendez-vous spécifique (sites "clients").

Pour éviter le seconde type d'interblocage, il suffit que le serveur n'accepte qu'au plus $N - 1$ demande sur l'entrée (ou la procédure) implémentant la transition 4.

Pour éviter le premier type d'interblocage le serveur peut :

- soit détecter des cycles et refuser une demande qui crée un cycle;
- soit simplement refuser une demande d'un site x en tant que client si un autre site est déjà en attente de rendez-vous avec ce site; c'est la solution que l'on implémente dans le programme suivant.

Le code de cette solution est donné dans les pages qui suivent.

Le programme principal

```
with Text_Io , Ada.Numerics.Discrete_Random ; use Text_Io ;

procedure Prog2 is

  N: constant Natural := 5;
  type Id is mod N;
  type Tab_Bool is array(Id) of Boolean;
  type Tab_Id   is array(Id) of Id;

  =====
  package Choix is new Ada.Numerics.Discrete_Random (Id);
  =====

  protected Numero is
    -- meme code que version precedente
  end Numero;

  =====
  protected Serveur is
    entry Demande_T1(X: in Id; Y: in Id; Acceptation: out boolean);
    entry Demande_T4(X: in Id; Y: out Id; Acceptation: out boolean);
  private
    entry Attente_Interne_T1(X: in Id; Y: in Id; Acceptation: out boolean);
    entry Attente_Interne_T4(X: in Id; Y: out Id; Acceptation: out boolean);

    Est_En_Attente_T1: Tab_Bool := (others => False);
    Est_En_Attente_T4: Tab_Bool := (others => False);

    Attent_Qui : Tab_Id;
    Sert_Qui   : Tab_Id;

    Garde_Int_T1: Boolean := False;
    Garde_Int_T4: Boolean := False;
  end Serveur;

  protected body Serveur is separate;

  =====
  procedure Work(X,Y: in Id) is
    -- meme code que version precedente
  end Work;

  =====
  task type Site;

  task body Site is
    Ego, Y, Mon_Choix : Id;
    Gen                : Choix.Generator;
    Ok                 : Boolean;
  begin
    Numero.Get(Ego);
    Choix.Reset(Gen);
    loop
      Mon_Choix := Choix.Random(Gen);
      if ( Mon_Choix = Ego) then
        Serveur.Demande_T4(Ego, Y, Ok);
      else
        Y := Mon_Choix;
        Serveur.Demande_T1(Ego, Y, Ok);
      end if;
      if ( Ok ) then
        Work(Ego, Y);
      end if;
    end loop;
  end Site;

  =====
  Les_Sites : array(Id) of Site;
begin
  null;
end Prog2;
```

Le corps de l'objet protégé Serveur

```
separate (Prog2)
protected body Serveur is

-----
entry Demande_T1(X: in Id; Y: in Id; Acceptation: out Boolean)
when True is
begin
  if Est_En_Attente_T4(Y) then
    Acceptation := True;
    Sert_Qui(Y) := X;
    Garde_Int_T4 := True;
  else
    -- on verifie que l'on ne cree pas d'interblocage
    if Est_En_Attente_T1(Y) then
      Acceptation := False;
    else
      Est_En_Attente_T1(X) := True;
      Attent_Qui(X) := Y;
      requeue Attente_Interne_T1;
    end if;
  end if;
end;

-----
entry Demande_T4(X: in Id; Y: out Id; Acceptation: out Boolean)
when True is
begin
  --
  -- on regarde si un site est en attente pour un rdv avec X
  -- dans ce cas on édbloque l'appelant et la garde associee
  -- a Attente_Interne_T1
  --
  for I in Id loop
    if (Est_En_Attente_T1(I)) and then (Attent_Qui(I) = X) then
      Y := I;
      Acceptation := True;
      Sert_Qui(X) := Y;
      Garde_Int_T1 := True;
      return;
    end if;
  end loop;
  --
  -- ici personne n'attendait X; on évrifie que l'on ne cree pas
  -- d'interblocage avant de mettre en attente X
  --
  if (Attente_Interne_T4.Count = N-1) then
    Acceptation := False;
  else
    Est_En_Attente_T4(X) := True;
    Sert_Qui(X) := X;
    requeue Attente_Interne_T4;
  end if;
end;

-----
entry Attente_Interne_T1(X: in Id; Y: in Id; Acceptation: out Boolean)
when Garde_Int_T1 is
begin
  if ( Sert_Qui(Y) = X ) then
    Acceptation := True;
    Garde_Int_T1 := False;
    Est_En_Attente_T1(X) := False;
  else
    requeue Attente_Interne_T1;
  end if;
end;
```

```

entry Attente_Interne_T4(X: in Id; Y: out Id; Acceptation: out Boolean)
when Garde_Int_T4 is
begin
  -- si la garde est ouverte c'est qu'un site Y a "reserve" le serveur
  -- X en positionnant Sert_Qui(X) a Y avec X /= Y (le valeur par
  -- defaut quand un site se met en attente de demande de rdv)
  -- ne pas oublier de remettre a faux la garde de l'entree
  if ( Sert_Qui(X) /= X ) then
    Y := Sert_Qui(X);
    Acceptation := True;
    Est_En_Attente_T4(X) := False;
    Garde_Int_T4 := False;
  else
    requeue Attente_Interne_T4;
  end if;
end;

end Serveur;

```
