Ndiaye et Gnebehi

Compte Rendu Paradigme de Programmation

I- Problème des deux cheminots

I.1- Cheminots première version

```
-- Cette solution ne peut malgré tout régler les problèmes de collisions
with TEXT_IO; use TEXT_IO;

procedure Cheminot_1 is
package int_io is new Integer_io(integer);  --=>principe de généricité
use int_io;
		--Variable Globale
		panier:integer:=0;

		-- Processus 1
		task processus1;  --=> la spécification
		task body processus1 is
			Begin
				while(panier=1) loop
				   Put_Line("Le Train_1 fait la sieste");
				end loop;
				   panier:=panier + 1;
				   Put_Line("Le Train_1 est en Section Critique");
				   -- protocole de sortie
				   panier:=panier - 1;

	end processus1;
	-- Processus 2
		task processus2;  --=> la spécification
		task body processus2 is
			Begin
				while(panier=1) loop
				   Put_Line("Le Train_2 fait la sieste");
				end loop;
				    panier:=panier + 1;
				   Put_Line("Le Train_2 est en Section Critique");
				   -- protocole de sortie
				   panier:=panier - 1;
	end processus2;

begin
Null;
end Cheminot_1;
```

Propriété1: **Non vérifiée**

Pour les traces d'exécutions suivantes :
-Si panier = 0 , P1 et P2 rentrent en SC
Ceci est la cause d'une collision entre les deux trains

Propriété2: **Non vérifiée**

-P1 demande la SC depuis l'état initial il peut l'avoir puisque panier=0
-P2 demande la SC depuis l'état initial il peut l avoir puisque panier=0

Propriété3: **Non vérifiée**

Supposons que
P1 en SC et
P2 en attente non réalisable car panier = 0
P2 entre son SC en même temps que P1

<u>Propriété Sûreté</u>: Interblocage Vérifiée

Supposons que P1 et P2 sont en interblocage c'est à dire en attente tous les deux
-P1 en attente => panier=1
-P2 en attente => panier=1
Les deux processus P1 et P2 rentrent en interblocage

<u>Propriété de vivacité:</u> Équité Non équitable

Supposons que P1 en SC , P2 en attente c'est à dire que P1 sort de la SC exécute son PS et redemande la SC
-Si P1 redemande la SC , il peut l'avoir => Non équitable

I.2- Cheminots deuxième version

```
-- Cette solution a pour but d'éviter les collisions mais plusieurs train ne peuvent passer
with TEXT_IO; use TEXT_IO;

procedure Cheminot_2 is
package int_io is new Integer_io(integer);  --=>principe de généricité
use int_io;
        --Variable Globale
        panier:integer:=0;

        -- Processus 1
        task processus1;  --=> la spécification
        task body processus1 is
            Begin
                while(panier=1) loop
                   Put_Line("Le cheminot bolivien s'arrete jusqu'à ce que le panier soit vide");
                end loop;
                   --Section critique
                   Put_Line("Le Train_1 est en Section Critique");
                   -- protocole de sortie
                   Put_Line("Le cheminot_1 sort de la zone critique et vide le panier");
                   panier:=panier + 1;

    end processus1;
    -- Processus 2
        task processus2;  --=> la spécification
        task body processus2 is
            Begin
                while(panier=0) loop
                   Put_Line("Le cheminot peruvien attend que le panier contienne une pierre pour passer");
                end loop;
                   --Section critique
                   Put_Line("Le Train_2 est en Section Critique");
                   -- protocole de sortie
                   Put_Line("Le cheminot_2 sort de la section critique et vide le panier");
                   panier:=panier - 1;
    end processus2;

begin
Null;
end Cheminot_2;
```

<u>Propriété1</u>: Vérifiée

Pour les traces d'exécutions suivantes :

-Si panier = 0 , P1 rentre en SC et P2 en attente
-Si panier = 1 , P2 rentre en SC P1 en attente

<u>Propriété2</u>: Non Vérifiée

-P1 demande la SC depuis l'état initial il peut l'avoir puisque panier=0

-P2 demande la SC depuis l'état initial il ne peut pas l'avoir puisque panier=0
Idem si panier=1

Propriété3: Vérifiée

Supposons que

P1 en SC et P2 en attente réalisable car panier = 0

C'est à dire P1 sort de la SC exécute son PS c'est à dire panier=panier+1
et libère P2 de l'attente active

Propriété Sûreté: Interblocage **Non Vérifiée**

Supposons que P1 et P2 sont en interblocage c'est à dire en attente tous les deux
-P1 en attente => panier=1
-P2 en attente => panier=0

Contradiction : La variable panier ne peut prendre 2 valeurs.
Conclusion : Les deux processus P1 et P2 ne rentrent pas en interblocage

Propriété de vivacité: Équité Équitable

Supposons que P1 en SC , P2 en attente c'est à dire panier=0.
P1 sort de la SC exécute son PS c'est à dire panier=panier+1
et redemande la SC
-Si P1 redemande la SC , il ne peut pas l'avoir => équitable

Si le cheminot Peruvien ne peut pas faire passer plusieurs trains par jour c'est su au fait que pour qu'un cheminot Peruvien passe il faut qu'un cheminot Bolivien passe avant comme l'indique la (propriété 2)

I.3- Cheminots troisième version

```
-- Cette solution marche à merveille jusqu'à ce qu'on constate que plus aucun train ne circule
with TEXT_IO; use TEXT_IO;

procedure Cheminot_3 is
package int_io is new Integer_io(integer);  --=>principe de généricité
use int_io;
            --Variable Globale
            panierBolivien:integer:=0;
            panierPeruvien:integer:=0;

            -- Processus 1
            task processus1;  --=> la spécification
            task body processus1 is
                  Begin
                  panierBolivien:=panierBolivien + 1;
                        while(panierPeruvien=1) loop
                           panierBolivien:=panierBolivien - 1;
                           Put_Line("Le cheminot Bolivien fait la sieste pour se réveiller plus tard");
                           panierBolivien:=panierBolivien +1;
                        end loop;
                           -- Section critique
                           Put_Line("Le Train_1 est en Section Critique");
                           -- protocole de sortie
                           Put_Line ("Le Train_1 entre en Section critique");
                           panierBolivien:=panierBolivien - 1;

      end processus1;
      -- Processus 2
            task processus2;  --=> la spécification
            task body processus2 is
                  Begin
                  panierPeruvien:=panierPeruvien+1;
                        while(panierBolivien=1) loop
                           panierPeruvien:=panierPeruvien - 1;
                           Put_Line("Le cheminot Peruvien fait la sieste pour se réveiller plus tard");
             panierPeruvien:=panierPeruvien + 1;
                        end loop;
```

```
                        -- Section Critique
                        Put_Line("Le Train_2 est en Section Critique");
                        -- protocole de sortie
                        Put_Line("Le cheminot sort de la section critique et vide le panier");
                        panierPeruvien:=panierPeruvien - 1;
    end processus2;

begin
Null;
end Cheminot_3;
```

Propriété 1: Vérifiée

Supposons P1 et P2 rentrent en SC
c'est à dire on a réussi à rentrer en SC par entrelacement

- P1 en SC => *panierPeruvien=0*
- P2 en SC => *panierBolivien=0*
  P1 doit exécuter l'instruction *panierBolivien=panierBolivien + 1* et
  P2 doit exécuter l'instruction *panierPeruvien=panierPeruvien+1*

Contradiction : Il y a au plus un processus en SC

Propriété 2: Vérifiée

P1 demande seul la SC , est ce qu'il peut l'avoir ?
- Depuis l'état initial , si P1 demande la SC , il peut l'avoir vu que *panierPeruvien=0*
- Si P2 demande seul la SC depuis l'état initial il peut aussi l'avoir car *panierBolivien=0*

Propriété 3: Vérifiée

-P1 en SC , P2 en attente c'est à dire *panierPeruvien=1 , panierBolivien=1*
P1 sort de la SC , exécute son PS *panierBolivien:=panierBolivien – 1* ceci débloqua P2 de l'attente
-Idem P2 en SC et P1 en attente

Interblocage:
La version 3 ne permet pas de vérifier cette propriété

Propriété de vivacité (Équité): Équitable

-P1 en SC et P2 en attente c'est à dire *panierPeruvien=1 , panierBolivien=1*
Quand P1 sort de la SC , il exécute son PS *panierBolivien:=panierBolivien – 1*
Quand P1 redemande , il ne peut pas l'avoir car *panierPeruvien=1*

II- Code ADA de l'Algorithme de Peterson

```
with TEXT_IO; use TEXT_IO;

procedure Peterson2process is
package int_io is new Integer_io(integer);
use int_io;

tour:integer:=0;
d1:boolean:=false;
d2:boolean:=false;

task processus1;
task body processus1 is
I:integer;

begin
        for I in 1..2 loop
                -- protocole d entree
```

```
                        tour:=2;
                        d1:=true;
                        while ((d2=true)and(tour=2))  loop
                                Put_Line ("Processus1 en attente");
                        end loop;

                        -- process1 entre en SC
                        Put_Line ("processus1 en SC");
                        d1:=false;
        end loop;
end processus1;

task processus2;
task body processus2 is
J:integer;
begin
        for J in 1.. 2 loop
                        -- protocole d entree
                        tour:=1;
                        d2:=true;
                        while ((d1=true)and(tour=1)) loop
                                Put_Line ("Processus2 en attente");
                        end loop;

                        -- process2 entre en SC
                        Put_Line ("processus2 en SC");
                        -- protocole de sortie
                        d2:=false;
        end loop;
end processus2;
begin
        Null;
end Peterson2process;
```

III- Code ADA de l'Algorithme de Peterson pour deux processus (la version symétrique)

```
with TEXT_IO; use TEXT_IO;
procedure PetersonSym is
package int_io is new Integer_io(integer);
use int_io;
demande:array(0..1)of boolean := (false,false);
tour:integer;
procedure entree(i:integer)is
        begin
                demande(i):=true;
                tour:=(i+1) mod 2;
                while((demande(tour) = true)and(tour /= i))loop
                Put_Line("Attente");
                end loop;
        end entree;
        procedure sortie(i:integer)is
        begin
           demande(i):=true;
               Put_Line("Sortie");
        end sortie;
        task process0;
        task body process0 is
        i:integer;
        begin
#####test avec 2 itérations#####
              for i in 1..4 loop
              entree(0);
#####processus entre en SC #####
          Put_Line("processus0 en SC");
#####protocole de sortie#####
            sortie(0);
        end loop;
        end process0;
        task process1;
        task body process1 is
        i:integer;
        begin
#####test avec 2 itérations#####
            for i in 1..4 loop
                        entree(0);
#####processus entre en SC#######
                        Put_Line("processus1 en SC");
#####protocole de sortie#####
                        sortie(0);
             end loop;
        end process1;
begin
        Null;
```

```
end PetersonSym
```

IV- Montrer que l'algorithme de Peterson pour 2 processus, généralisé (de manière brutale) pour 3 processus ne fonctionne pas

```
with TEXT_IO; use TEXT_IO;

procedure Peterson3processus is
package int_io is new Integer_io(integer);-->principe de généricité
use int_io;

	--Variable Globale
	demande1:boolean:=false;
	demande2:boolean:=false;
	demande3:boolean:=false;
	tour:integer;

	-- Processus 1
		task processus1;  --=> la spécification
		task body processus1 is
			Begin
			demande1:=true;
			tour:=2;
				while( ((demande1=true)or(demande3=true)) and (tour/=1)) loop
				--tour peut etre = 2,3,.....  sauf 1 et d1=vrai
				   Put_Line("Processus1 en attente");
				end loop;
				   -- Section critique
				 --     d1=faux
				   Put_Line("Processus1 en SC");
				   --PS
				   demande1:=false;

	end processus1;

	 -- Processus 2
		task processus2;  --=> la spécification
		task body processus2 is
			Begin
			demande2:=true;
			tour:=3;
				while( ((demande1=true)or(demande3=true)) and (tour/=2)) loop
				   Put_Line("Processus2 en attente");
				end loop;
				   -- Section critique
				   Put_Line("Processus2 en SC");
				   --PS
				   demande2:=false;

	end processus2;

	 -- Processus 3
		task processus3;  --=> la spécification
		task body processus3 is
			Begin
			demande3:=true;
			tour:=1;
				while( ((demande1=true)or(demande2=true)) and (tour/=3)) loop
				   Put_Line("Processus3 en attente");
				end loop;
				   -- Section critique
				   Put_Line("Processus3 en SC");
				   --PS
				   demande3:=false;

	end processus3;
begin
NULL;
end Peterson3processus;
```

<u>Propriété 1</u>: Non Vérifiée

Trace 1: Processus1 et Processus2

Pour les traces d'exécutions suivantes :

Supposons que P1 et P2 sont en SC
-P1 en SC => d1=vrai et tour!=1
-P2 en SC => d2=vrai et tour!=2

Pas de contradiction P1 et P2 sont en SC

Trace2: Processus1 et Processus3
Pour les traces d'exécutions suivantes :

Supposons que P1 et P3 sont en SC
-P1 en SC => d1=vrai et tour!=1
-P3 en SC => d3=vrai et tour!=3
Pas de contradiction P1 et P3 sont en SC

Trace 3: Processus1 ,Processus2 et Processus3
Pour les traces d'exécutions suivantes :

Supposons que P1 et P2 et P3 sont en SC
-P1 en SC => d1=vrai et tour!=1
-P2 en SC => d2=vrai et tour!=2
-P3 en SC => d3=vrai et tour!=3
Pas de contradiction P1 et P2 sont en SC

Propriété 2: **Non Vérifiée**

-P1 demande la SC depuis l'état initial il ne peut pas l'avoir puisque d1=faux
-P2 demande la SC depuis l'état initial il ne peut pas l'avoir puisque d1=faux
-P3 demande la SC depuis l'état initial il ne peut pas l'avoir puisque d1=faux

Propriété3: Non Vérifiée

Trace 1: Processus1 ,Processus2

-P1 en SC et P2 en attente car d1=vrai,tour !=2,c'est à dire P1 sort de la SC ,
exécute son PS => d1=faux , Ceci ne permettra pas de débloquer P2 de l'attente active
-Idem pour P2 en SC et P1 en attente

Propriété de Sûreté: Interblocage Vérifiée

Trace 1: Processus1 et Processus2
Supposons que P1 et P2 sont en interblocage c'est à dire en attente tous les deux
- P1 en attente => d2=vrai , tour!=1
- P2 en attente => d1=vrai , tour !=2

Trace2: Processus1 et Processus3
Supposons que P1 et P3 sont en interblocage c'est à dire en attente tous les deux

-Si d2=vrai et tour!=1 , P1 en attente
-Si d1=vrai et tour !=3, P3 en attente

Trace 3: Processus1 ,Processus2 et Processus3
Pour les traces d'exécutions suivantes :

-Si d2=vrai et tour!=1 , P1 en attente
-Si d1=vrai et tour!=2 , P2 en attente
-Si d1=vrai et tour!=3, P3 en attente
Ainsi tout les processus sont bloqués

<u>Propriété de Vivacité</u>: Équité Vérifiée

Trace 1: Processus1 et Processus2
Supposons que P1 en SC et P2 en attente c'est à dire d1=vrai et tour =2
P1 sort de la SC exécute son protocole de sortie c'est à dire d1=faux
P1 redemande la SC il peut pas l'avoir car d1=vrai (Équitable)

Trace 2: Processus1 et Processus3
Supposons que P1 en SC et P3 en attente c'est à dire d2=vrai et tour =3
P1 sort de la SC exécute son protocole de sortie c'est à dire d1=faux
P1 redemande la SC il peut pas l'avoir car d1=vrai (Équitable)

Trace 3: Processus1 , Processus2 et Processus3
Supposons que P1 en SC et
- P2 en attente c'est à dire d1=vrai tour=2
- P3 en attente c'est à dire d2=vrai et tour=3
P1 exécute son PS c'est à dire d1=faux
et redemande la SC , il peut pas l'avoir car d1=vrai

V- Solution

V-1 Explication

La solution proposée est une solution basée sur l'algorithme de Peterson .
On dispose d'une variable ***entrer*** décrivant qu'un processus entre en section critique ou pas .
Un processus désirant rentrer en section critique y entre que si les demandes des autres processus sont fausses.
On dispose également de trois variables ***demande1, demande2 et demande3*** par défaut égales à Vrai. Ces variables indiquent que tout les processus demandent à entrer en section critique.

V-2 Code ADA

```
with TEXT_IO; use TEXT_IO;

procedure Solution is
package int_io is new Integer_io(integer);
use int_io;

demande1: boolean := TRUE;
demande2: boolean := TRUE;
demande3: boolean := TRUE;

task processus1;
task body processus1 is

I:integer;
entrer:boolean;

begin
                        -- protocole d entree
                        entrer := FALSE;
                        while entrer = FALSE loop
                                if demande2 = FALSE and demande3 = FALSE then
                                        entrer := TRUE;
                                        demande1 := TRUE;
                                else
                                        demande1 := FALSE;
                                        Put_Line("Processus 1 en attente");
                                end if;
                        end loop;
                        -- process1 entre en SC
                        Put_Line ("Processus 1 en SC");
                        -- protocole de sortie
                        demande1 := FALSE;
                        Put_Line ("Processus 1 sort de la SC");
end processus1;
```

```
task processus2;
task body processus2 is

J:integer;
entrer:boolean;

Begin
                        -- protocole d entree
                        entrer := FALSE;
                        while entrer = FALSE loop
                                if demande1 = FALSE and demande3 = FALSE then
                                        entrer := TRUE;
                                        demande2 := TRUE;
                                else
                                        demande2 := FALSE;
                                        Put_Line("Processus 2 en attente");
                                end if;
                        end loop;

                        -- process2 entre en SC
                        Put_Line ("Processus 2 en SC");

                        -- protocole de sortie
                        demande2 := FALSE;
                        Put_Line ("Processus 2 sort de la SC");
end processus2;

task processus3;
task body processus3 is

K:integer;
entrer:boolean;

Begin
                        -- protocole d entree
                        entrer := FALSE;
                        while entrer = FALSE loop
                                if demande1 = FALSE and demande2 = FALSE then
                                        entrer := TRUE;
                                        demande3 := TRUE;
                                else
                                        demande3 := FALSE;
                                        Put_Line("Processus 3 en attente");
                                end if;
                        end loop;
                        -- process3 entre en SC
                        Put_Line ("Processus 3 en SC");

                        -- protocole de sortie
                        demande3 := FALSE;
                        Put_Line ("Processus 3 sort de la SC");
end processus3;

begin
Null;
end Solution;
```

V-3 Vérification des trois propriétés

Propriété 1 : Vérifiée

-Si demande1=vrai c'est P1 en SC et P2,P3 en attente
-Si demande2=vrai c'est P2 en SC et P1,P3 en attente
-Si demande3=vrai c'est P3 en SC et P2,P1 en attente

Propriété 2 : Vérifiée

Depuis l'état initial si
-P1 demande la SC il peut l'avoir vu que demande1=vrai
-P2 demande la SC il peut l'avoir vu que demande2=vrai
-P3 demande la SC il peut l'avoir vu que demande3=vrai

Propriété 3 : Vérifiée

Trace 1 : Processus 1 et Processus 2
P1 en SC et P2 en attente => demande1=vrai, demande2=faux
Quand P1 sort de la SC il exécute son PS c'est à dire demande1=faux ceci permettra de débloquer P2 de l'attente active

Trace 2 : Processus 1 et Processus 3
P1 en SC et P3 en attente => demande1=vrai, demande3=faux
Quand P1 sort de la SC il exécute son PS c'est à dire demande1=faux ceci permettra de débloquer P3 de l'attente active