vs

Université de Lorraine
DIPLOME: Telecom Nancy 2A - Apprentissage
Épreuve: MVSI TP noté
Durée du sujet : 1 h 00
Date : Jeudi 27 novembre 2025 de 9 h 00 à 10 h 00
Lieu : Salle 1.8
Nom du rédacteur : Dominique Méry
Documents personnels autorisés

Les fichiers ont été préparés et sont sur l'espac
https://mery54.github.io/teaching/mvsi/

TP

Le dossier à remettre est une archive compressée (zip) contenant quatre dossiers ex1, ex2, ex3 et ex4. Chaque dossier répond aux questions en fournissant les éléments suivants:

- ex.txt: explications sur ce qui a été fait et sur les éventuels problèmes rencontrés pour mettre au point la solution. En particulier, vous pourrez indiqué les constantes utiulisées.
- ex.toolbox
- ex.tla
- ex.cfg

Le dossier portera le nom suivant tp-<nom>.zip et sera envoyé à dominique.mery@loria.fr avec le sujet "mvsi-tp". Il sera déposé sur arche aussi.

Exercice 1

Soit le contrat suivant:

variables $int\ x, int\ y, int\ z$
requires $P(x_0, y_0, z_0)$
ensures $Q(x_0, y_0, z_0, x_f, y_f, z_f)$
begin
$0 : z := z + x + y$
$1 : z = y \wedge z = x$
end

Le listing 1 contient le squelette du module TLA et sert de guide pour répondre aux questions.

Question 1.1 Compléter la précondition P, la postcondition Q et l'assume.

Question 1.2 Compléter l'action `step0_1`.

Question 1.3 Compléter la postcondition et vérifier la propriété de correc tion partielle.

Listing 1: exe253part

```
--------- MODULE exe25_3 --------
EXTENDS Integers,TLC
CONSTANTS x0,y0,z0  (* x0,y0,z0   are the initial values *), un
P(u,v,w) ==
Q(u0,v0,w0,u,v,w) ==
-------------------------------------------------------------
ASSUME
-------------------------------------------------------------
VARIABLES x,y,z,pc
-------------------------------------------------------------
step0_1 ==

skip == UNCHANGED <<x,y,z,pc>>

-------------------------------------------------------------
Init ==
Next ==
-------------------------------------------------------------

safetypc == pc="1" =>

=========
```

Exercice 2

Soit un nombre entier $a \in \mathbb{Z}$. On se donne l'expression algorithmique annotée comme suit:

$0 : x = z \ \wedge \ y = x * z \ \wedge \ x + y + z = 2a$
$z := x * y + 3 * y * y + 3 * x * y * y + y * x * z * z * x;$
$1 : \ z = (x + y)^3$

Le listing 2 contient le squelette du module TLA et sert de guide pour répondre aux questions.

Question 2.1 Compl'eter la précondition P, la postcondition Q et l'assume.

Question 2.2 Compléter l'action `step0_1`.

Question 2.3 Compl'eter la postcondition et v'erifier la propri'et'et de correction partielle.

Listing 2: exe253part

```
--------- MODULE exe25_4part --------
EXTENDS Integers,TLC
CONSTANTS x0,y0,z0  (* x0,y0,z0   are the initial values *), a, un
P(u,v,w) ==
Q(u0,v0,w0,u,v,w) ==
-----------------------------------------------------------------
ASSUME P(x0,y0,z0)
-----------------------------------------------------------------
VARIABLES x,y,z,pc
-----------------------------------------------------------------
step0_1 ==

skip == UNCHANGED <<x,y,z,pc>>

-----------------------------------------------------------------
Init ==
Next ==
-----------------------------------------------------------------

safetypc == pc="1" =>

=========
```

Exercice 3 Soit le programme C du listing 3. Son exécution est donnée dans le listing 4 et on constate des problèmes au niveau des résultats.Pour simplifier, cette fonction échange les valeurs des deux a et b en entrée. On s'intéresse à la fonction fS et on veut l'analyser.

Listing 3: swap.c

```
#include <stdio.h>
#include <limits.h>
// ***************************************************************
int ffS(int x,int y)
{int a,b,c,oa,ob,oc,r;
  a=x;b=y;oa=a;ob=b;oc=c;
  a=ob;
  b=oa;
  r=(a == y && b == x);
  return(r);
}
// ***************************************************************
int fS(int x,int y)
{
  int a,b,c,r;
  a=x;b=y;
  a=b;
  b=a;
  r=(a == y && b == x);
```

```
  return(r);
}
// ***************************************************************
int main()
{int x,y;



  for (int i = 0; i < 11; ++i){
    x=i;
    y=i+1;
    int a,b,c,r,oa,ob;
    a=x;b=y;oa=a;ob=b;
  a=ob;
  b=oa;
  r=(a == y && b == x);
  printf("La valeur de la fonction  pour  x=%d  et y=%d  est  a=%d  et b=%d \n",x,y,a,b);}

  printf("\n");  printf("\n");  printf("\n");
// ***************************************************************
  for (int i = 0; i < 11; ++i){
    x=i;
    y=i+1;
    int a,b,c,r;
  a=x;b=y;
  a=b;
  b=a;
  r=(a == y && b == x);
  printf("La valeur de la fonction  pour  x=%d  et y=%d  est  a=%d  et b=%d \n",x,y,a,b);}
     return 0;
}
```

Listing 4: swap.txt

```
La valeur de la fonction  pour  x=0  et y=1  est  a=1  et b=0
La valeur de la fonction  pour  x=1  et y=2  est  a=2  et b=1
La valeur de la fonction  pour  x=2  et y=3  est  a=3  et b=2
La valeur de la fonction  pour  x=3  et y=4  est  a=4  et b=3
La valeur de la fonction  pour  x=4  et y=5  est  a=5  et b=4
La valeur de la fonction  pour  x=5  et y=6  est  a=6  et b=5
La valeur de la fonction  pour  x=6  et y=7  est  a=7  et b=6
La valeur de la fonction  pour  x=7  et y=8  est  a=8  et b=7
La valeur de la fonction  pour  x=8  et y=9  est  a=9  et b=8
La valeur de la fonction  pour  x=9  et y=10  est  a=10  et b=9
La valeur de la fonction  pour  x=10  et y=11  est  a=11  et b=10



La valeur de la fonction  pour  x=0  et y=1  est  a=1  et b=1
La valeur de la fonction  pour  x=1  et y=2  est  a=2  et b=2
La valeur de la fonction  pour  x=2  et y=3  est  a=3  et b=3
La valeur de la fonction  pour  x=3  et y=4  est  a=4  et b=4
La valeur de la fonction  pour  x=4  et y=5  est  a=5  et b=5
La valeur de la fonction  pour  x=5  et y=6  est  a=6  et b=6
La valeur de la fonction  pour  x=6  et y=7  est  a=7  et b=7
La valeur de la fonction  pour  x=7  et y=8  est  a=8  et b=8
La valeur de la fonction  pour  x=8  et y=9  est  a=9  et b=9
La valeur de la fonction  pour  x=9  et y=10  est  a=10  et b=10
La valeur de la fonction  pour  x=10  et y=11  est  a=11  et b=11
```

Nous allons compléter le listing 6 et ce listing est partiellement complété.

Question 3.1 On suppose que les valeurs de a et de b sont données et on suppose donc que a=a0 et b=b0 où a0 et b0 désigne les valeurs initiales de a et de b. On se reporte au code du listing 5.

Donner la postcondition Q qui est la relation entre les valeurs initiales et les valeurs finales afin que le code échange les valeurs des variables c'est-à-dire que la valeur de r doit être vraie toujours.

Listing 5: labelswap.c

```
int a=x,b=y,c,r;
l0:
  c=a;
l1:
  a=b;
l2:
  b=c;
l3:
  r=(a == y && b == x);
```

Question 3.2 Traduire ce code sous la forme d'actions suivant les étiquettes `l0`, `l1`, `l2`,`l3` et `done`

Question 3.3 Vérifier la propriété de correction partielle et et celle d'absence des erreurs à l'exécution.

Listing 6: exe252part.tla

```
--------- MODULE exe25\_2part --------
EXTENDS Integers,TLC
CONSTANTS a0,b0,c0,r0, un
comp(u,v) == u=b0 /\ v=a0
P(u,v,w,x) == TRUE
Q(u0,v0,w0,x0,u,v,w,x) ==
-------------------------------------------------------------------
VARIABLES a,b,c,r,pc
-------------------------------------------------------------------
l0 ==
l1 ==
l2 ==
l3 ==
skip == UNCHANGED <<c,a,b,r,pc>>
-------------------------------------------------------------------
Init == a=a0 /\ b=b0 /\ c=c0 /\ r=r0 /\ pc="l0"
Next ==
-------------------------------------------------------------------
safetypc == pc="done" =>
===================================================================
```

Exercice 4 Soit le progtramme C décrit dans le listing 7.

Listing 7: power2.c

```
#include <stdio.h>
#include <limits.h>


int ffS(int x)
{int  r,k,cv,cw,or,ok,ocv,ocw;
  r=0;k=0;cv=0;cw=0;or=0;ok=k;ocv=cv;ocw=cw;
  while (k<x)
    {
    ok=k;ocv=cv;ocw=cw;
      k=ok+1;
      cw=ocw+2;
      cv=ocv+ocw+1;
     }
  r=cv;
  return(r);
}

int fS(int x)
```

```
  {int r=0;k=0;cv=0;cw=0;
  while (k<x)
    {
       k=k+1;
      cw=cw+2;
      cv=cv+cw+1;
    }
  r=cv;
  return(r);
}
// ************************************************************
int main()
{
  for (int i = 0; i < 11; ++i){
    printf("La␣valeur␣de␣la␣fonction␣␣pour␣␣z=%d␣␣est␣␣%d␣␣␣␣et␣on␣devrait␣obtenir␣%d\n",i
     return 0;
}
```

Si on l'exécute, on obtient les résultats du listing 8 et on constate qu'il y a un problème de calcul de la bonne fonction. Nous allons analyser cette situation avec le langage TLA$^+$.

Listing 8: running

```
La valeur de la fonction  pour  z=0  est  0    et on devrait obtenir 0
La valeur de la fonction  pour  z=1  est  3    et on devrait obtenir 1
La valeur de la fonction  pour  z=2  est  8    et on devrait obtenir 4
La valeur de la fonction  pour  z=3  est  15    et on devrait obtenir 9
La valeur de la fonction  pour  z=4  est  24    et on devrait obtenir 16
La valeur de la fonction  pour  z=5  est  35    et on devrait obtenir 25
La valeur de la fonction  pour  z=6  est  48    et on devrait obtenir 36
La valeur de la fonction  pour  z=7  est  63    et on devrait obtenir 49
La valeur de la fonction  pour  z=8  est  80    et on devrait obtenir 64
La valeur de la fonction  pour  z=9  est  99    et on devrait obtenir 81
La valeur de la fonction  pour  z=10  est  120    et on devrait obtenir 100
```

On extrait le fragment de code décrit par le listing 9 et on pose quelques questions pour compléter le fichier du listing 10.

Listing 9: code

```
int r=0;k=0;cv=0;cw=0;
  inloop: while (k<x)
    {
      k=k+1;
      cw=cw+2;
      cv=cv+cw+1;
    }
  final: r=cv;
```

Question 4.1 L'initialisation des variables est définie par l'expression suivante:

```
int r=0;k=0;cv=0;cw=0;
```

On suppose aussi que la valeur initiale de x doit toujours être un entier naturel.
Traduire ces informations par une assertion P(u,v,w,x,y,z) définissant les conditions initiales.
L'assertion P(u,v,w,x,y,z) est instanciée sous la forme P(x0,r0,k0,cv0,cw0,pc0).

Question 4.2 Traduire les conditions initiales en définissant l'assertion Init.

Question 4.3 Traduire ce code sous la forme d'actions suivant les étiquettes `inloop` et`final`. On pourra introduire une étiquette de fin de code et notée `done`. La valeur initiale de pc est inloop. Il faut définir les actions suivantes:

- (test vrai)) `inloop` vers `inloop`
- (test faux) `inloop` vers `final`
- `final` vers `done`

Question 4.4 Définir la relation Next.

Question 4.5 Définir l'assertion Q(u0,v0,w0,x0,y0,z0,u,v,w,x,y,z) qui définit la postcondition.
Définir l'assertion safetypc qui permet de tester la correc tion partielle de ce code.

Question 4.6 Vérifier les propriétés d e correction partielle et d'absence d'erreurs à l'exécution.

Listing 10: labelpower2.c

```
--------- MODULE exe25_1part --------
EXTENDS Integers,TLC
CONSTANTS x0 (* x0 is the input *),
          r0,k0,cv0,cw0,pc0,
          un (* un est la valeur reprÃ©senta,t l'indÃ©fini *)
-----------------------------------------------------------
power(u) == u*u
P(u,v,w,x,y,z) ==
Q(u0,v0,w0,x0,y0,z0,u,v,w,x,y,z) ==
-----------------------------------------------------------
ASSUME
-----------------------------------------------------------
VARIABLES x,r,k,cv,cw,pc
-----------------------------------------------------------
inloop ==
outloop ==
final ==
-----------------------------------------------------------
Init ==
Next ==
-----------------------------------------------------------
safetypc == pc="done" =>

===========================================================================
```

Fin de l'énoncé