<
Cours Modélisation et vérification des systèmes informatiques
Exercices (avec les corrections)
Modélisation d'algorithmes en PlusCal (I)
par Dominique Méry
4 décembre 2025

Exercice 1 ☑
Nous allons utiliser la fonctionnalité de traduction d'un algorithme PlusCal en un module TLA pour vérifier des algorithmes. Pour chaque question, on écrira un module TLA contenant une expressiond e l'algorithme puis on traduira par un module et ensuite on analysera le module obtenu par rapport à la correction partielle et l'absence d'erreurs à l'exécution.

Question 1.1

$\ell_1 : x = 10 \ \wedge \ y = z{+}x \ \wedge z = 2{\cdot}x$
$y := z{+}x$
$\ell_2 : x = 10 \ \wedge \ y = x{+}2{\cdot}10$

Question 1.2 *On suppose que p est un nombre premier :*

$\ell_1 : x = 2^p \ \wedge \ y = 2^{p+1} \ \wedge \ x{\cdot}y = 2^{2\cdot p+1}$
$x := y{+}x{+}2^x$
$\ell_2 : x = 5{\cdot}2^p \ \wedge \ y = 2^{p+1}$

Question 1.3

$\ell_1 : x = 1 \ \wedge \ y = 12$
$x := 2{\cdot}y$
$\ell_2 : x = 1 \ \wedge \ y = 24$

Question 1.4

$\ell_1 : x = 11 \ \wedge \ y = 13$
$z := x; x := y; y := z;$
$\ell_2 : x = 26/2 \ \wedge \ y = 33/3$

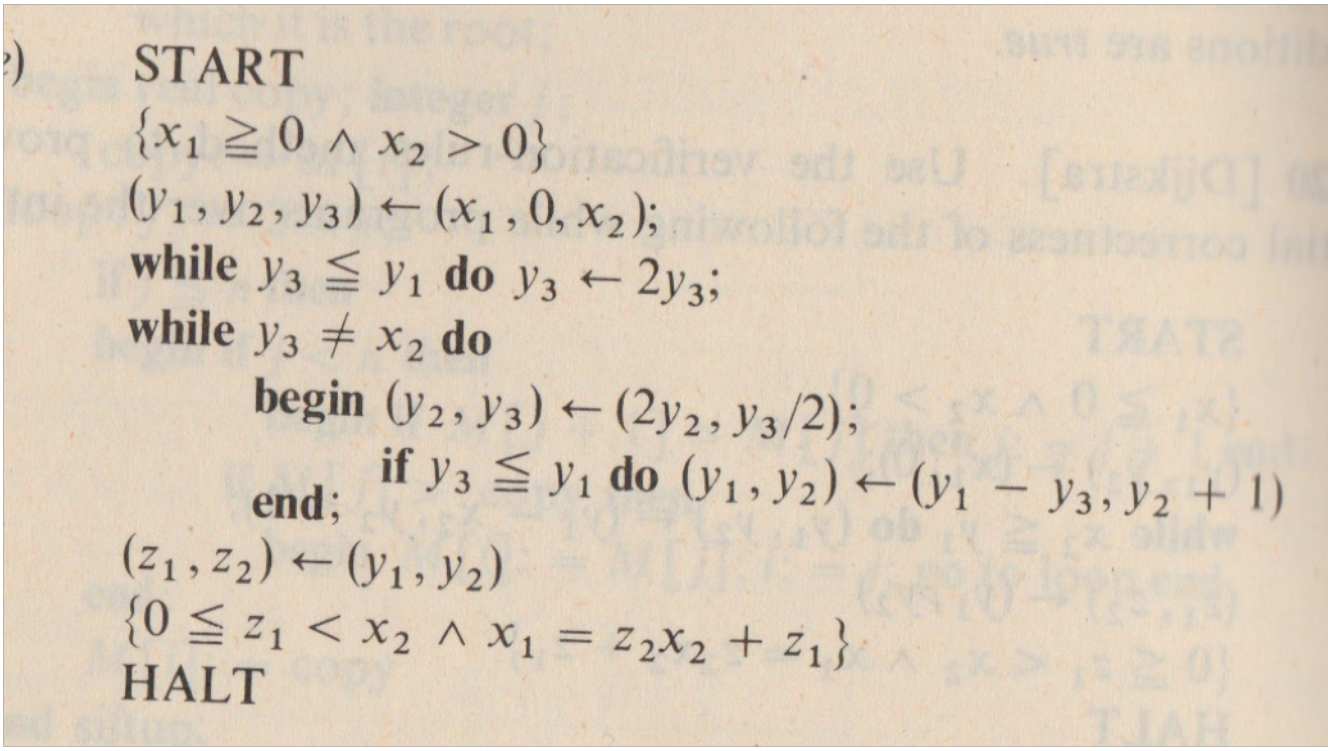
Question 1.5 *Dans cette question, on considère l'algorithme correspondant au calcul du maximum de deux nombres.*

Exercice 2 ☑
On considère l'algorithme suivant :

Requires : $x_0, y_0 \in \mathbb{N}$
Ensurees : $z_f = max(x_0, y_0)$

$\ell_0 : \{\ldots\}$
if $X < Y$ **then**
  $\ell_1 : \{\ldots\}$
  $Z := Y$;
  $\ell_2 : \{\ldots\}$
else
  $\ell_3 : \{\ldots\}$
  $Z := X$;
  $\ell_4 : \{\ldots\}$
;
$\ell_5 : \{\ldots\}$

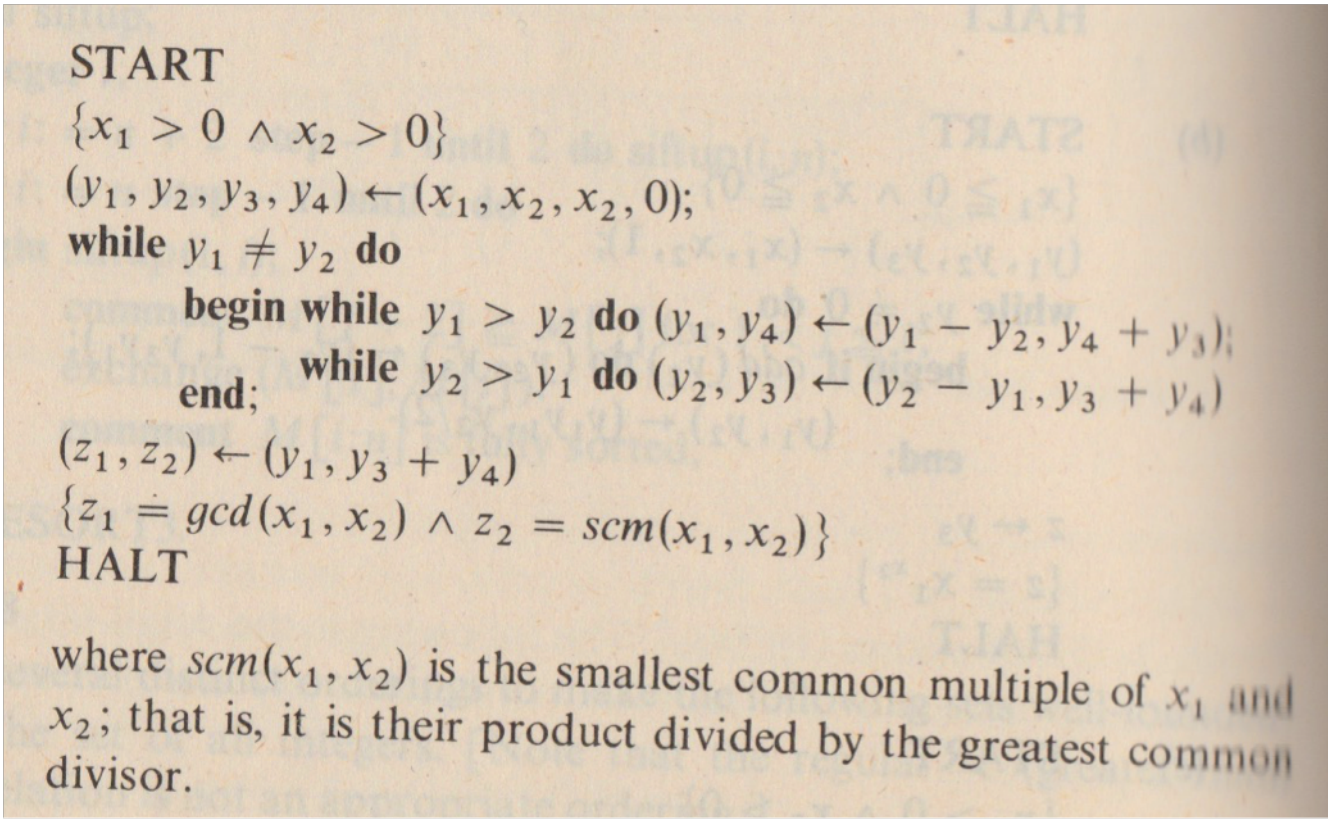
Algorithme 1: MAX annotée



Question 2.1 *Montrer que cet algorithme est aprtiellement correct par rapport à sa précondition et à sa postcondition qu'il faudra énoncer. Pour cela, on traduira cet algorithme sous forme d'un module à partir du langage PlusCal.*

Question 2.2 *Montrer qu'il est sans erreur à l'exécution.*

Exercice 3 ⍁
On considère l'algorithme suivant :



START
$\{x_1 > 0 \wedge x_2 > 0\}$
$(y_1, y_2, y_3, y_4) \leftarrow (x_1, x_2, x_2, 0);$
while $y_1 \neq y_2$ **do**
begin while $y_1 > y_2$ **do** $(y_1, y_4) \leftarrow (y_1 - y_2, y_4 + y_3);$
while $y_2 > y_1$ **do** $(y_2, y_3) \leftarrow (y_2 - y_1, y_3 + y_4)$
end;
$(z_1, z_2) \leftarrow (y_1, y_3 + y_4)$
$\{z_1 = gcd(x_1, x_2) \wedge z_2 = scm(x_1, x_2)\}$
HALT

where $scm(x_1, x_2)$ is the smallest common multiple of x_1 and x_2; that is, it is their product divided by the greatest common divisor.

Question 3.1 *Montrer que cet algorithme est aprtiellement correct par rapport à sa précondition et à sa postcondition qu'il faudra énoncer. Pour cela, on traduira cet algorithme sous forme d'un module à partir du lanage PlusCal.*

Question 3.2 *Montrer qu'il est sans erreur à l'exécution.*

◇– **Solution de l'exercice 3**

MODULE *pluscal4*

EXTENDS *TLC*, *Integers*, *Naturals*
CONSTANTS $x_1, x2, min, max$

(.
$-$*wfNext*
$--$*algorithm division* {
variables $y_1, y2, y_3, z1, z_2$;
{
$l1 : y_1 := x_1; y2 := 0; y_3 := x_2;$
$l2$: *while* $(y_3 \leq y_1)$ {
$y3 := 2 \cdot y_3;$
};
$l3$: *while* $(y_3 \neq x2)$ {
$y2 := 2 \cdot y_2;$
$y3 := y_3 \div 2;$
$l4$: *if* $(y_3 \leq y_1)$ {
$y1 := y_1 - y3;$
$y2 := y_2 + 1;$
};

```
    };
  l5 :  z_1 := y_1;
    z2 := y_2;
  l6 : print ⟨x_1, x2, z_1, z2⟩;
    }
}

.)
BEGIN TRANSLATION
CONSTANT defaultInitValue
VARIABLES y_1, y_2, y_3, z_1, z_2, pc

vars ≜ ⟨ y_1, y_2, y_3, z_1, z_2, pc ⟩

Init ≜  Global variables
        ∧ y_1 = defaultInitValue
        ∧ y_2 = defaultInitValue
        ∧ y_3 = defaultInitValue
        ∧ z_1 = defaultInitValue
        ∧ z_2 = defaultInitValue
        ∧ pc = "l1"

l1 ≜ ∧ pc = "l1"
     ∧ y_1' = x1
     ∧ y_2' = 0
     ∧ y_3' = x2
     ∧ pc' = "l2"
     ∧ UNCHANGED ⟨ z_1, z_2 ⟩

l2 ≜ ∧ pc = "l2"
     ∧ IF y_3 ≤ y1
         THEN ∧ y_3' = 2·y3
              ∧ pc' = "l2"
         ELSE ∧ pc' = "l3"
              ∧ y_3' = y3
     ∧ UNCHANGED ⟨ y_1, y_2, z_1, z_2 ⟩

l3 ≜ ∧ pc = "l3"
     ∧ IF y_3 ≠ x2
         THEN ∧ y_2' = 2·y2
              ∧ y_3' = (y_3 ÷ 2)
              ∧ pc' = "l4"
         ELSE ∧ pc' = "l5"
              ∧ UNCHANGED ⟨ y_2, y_3 ⟩
     ∧ UNCHANGED ⟨ y_1, z_1, z_2 ⟩

l4 ≜ ∧ pc = "l4"
     ∧ IF y_3 ≤ y1
         THEN ∧ y_1' = y_1−y3
              ∧ y_2' = y_2+1
         ELSE ∧ TRUE
              ∧ UNCHANGED ⟨ y_1, y_2 ⟩
     ∧ pc' = "l3"
     ∧ UNCHANGED ⟨ y_3, z_1, z_2 ⟩

l5 ≜ ∧ pc = "l5"
```

$\land\ z_1' = y1$
$\land\ z_2' = y2$
$\land\ pc' = \text{"l6"}$
$\land$ UNCHANGED $\langle\ y_1,\ y_2,\ y_3\ \rangle$

$l6 \triangleq \land\ pc = \text{"l6"}$
$\land\ PrintT(\langle x_1, x2, z_1, z2\rangle)$
$\land\ pc' = \text{"Done"}$
$\land$ UNCHANGED $\langle\ y_1,\ y_2,\ y_3,\ z_1,\ z_2\ \rangle$

$Next \triangleq l_1 \lor l_2 \lor l_3 \lor l_4 \lor l_5 \lor l6$
$\lor$ Disjunct to prevent deadlock on termination
$(pc = \text{"Done"} \land$ UNCHANGED $vars)$

$Spec \triangleq Init \land \Box[Next]_{vars}$

$Termination \triangleq \Diamond(pc = \text{"Done"})$

END TRANSLATION

$Iloop(u, v) \triangleq x_1 = v{\cdot}x_2{+}u$

$Qpc \triangleq pc = \text{"Done"} \Rightarrow x_1 = z2{\cdot}x_2{+}z1 \land 0 \leq z_1 \land z_1 < x2$

$COND(U) \triangleq min \leq U \land U \leq max$

$Qef \triangleq min \leq y_1 \land y_1 \leq max$

$i \triangleq Iloop(y_1, y2)$

Modification History
Last modified Tue Oct 04 08 :15 :08 CEST 2016 by mery
Created Wed Nov 18 16 :33 :27 CET 2015 by mery

Fin 3

Exercice 4 ☑
Nous allons utiliser la fonctionnalité de traduction d'un algorithme PlusCal en un module TLA pour vérifier des algorithmes. Pour chaque question, on écrira un module TLA contenant une expression de l'algorithme puis on traduira par un module et ensuite on analysera le module obtenu par rapport à la correction partielle et l'absence d'erreurs à l'exécution.

Question 4.1 *Soit l'annotation suivante :*

$\ell_1 : x = 10 \land y = z{+}x \land z = 2{\cdot}x$
$y := z{+}x$
$\ell_2 : x = 10 \land y = x{+}2{\cdot}10$

Traduire en PlusCal.

◇– **Solution de la question 4.1**

MODULE $tp4$

EXTENDS *TLC*, *Integers*, *Naturals*

(·

```
--algorithm ex1 {
variables x, y, z;
{
init : x := 10; z := 2·x; y := z+x;
l1 : y := z+x;
print ⟨x, y, z⟩;

}
}
·)

i ≜
    ∧ pc = "l1" ⇒ x = 10 ∧ y = z+x ∧ z = 2·x
    ∧ pc = "Done" ⇒ x = 10 ∧ y = x+2·10
```

Fin 4.1

Exercice 5 *On considère l'algorithme suivant :*

START
$\{x_1 > 0 \wedge x_2 > 0\}$
$(y_1, y_2, y_3, y_4) \leftarrow (x_1, x_2, x_2, 0)$;
while $y_1 \neq y_2$ **do**
    begin while $y_1 > y_2$ **do** $(y_1, y_4) \leftarrow (y_1 - y_2, y_4 + y_3)$;
    end; **while** $y_2 > y_1$ **do** $(y_2, y_3) \leftarrow (y_2 - y_1, y_3 + y_4)$
$(z_1, z_2) \leftarrow (y_1, y_3 + y_4)$
$\{z_1 = gcd(x_1, x_2) \wedge z_2 = scm(x_1, x_2)\}$
HALT

where $scm(x_1, x_2)$ is the smallest common multiple of x_1 and x_2; that is, it is their product divided by the greatest common divisor.

Question 5.1 *Montrer que cet algorithme est aprtiellement correct par rapport à sa précondition et à sa postcondition qu'il faudra énoncer. Pour cela, on traduira cet algorithme sous forme d'un module à partir du lanage PlusCal.*

Question 5.2 *Montrer qu'il est sans erreur à l'exécution.*

Exercice 6 *Exponentiation en PlusCal*
Ecrire l'algorithme de l'exponentiation avec PlusCal.
On suppose que x_1 et x_2 sont des constantes.

◇– **Solution de l'exercice 6**

MODULE $tp3$

```
EXTENDS Naturals, Integers, TLC

CONSTANT MAXINT, u, v

(·
–termination
–wfNext
--algorithm Power {
    variables x1 = u;    1st integer
```

precondition : $x_1 \in \mathbb{N} \wedge x_2 \in \mathbb{N} \wedge x_1 \neq 0$
postcondition : $z = {x_1}^{x_2}$
local variables : $y_1, y_2, y_3 \in \mathbb{Z}$

$\ell_0 : \{y_1, y_2, y_3, z \in \mathbb{Z}\}$
$(y_1, y_2, y_3) := (x_1, x_2, 1)$;
$\ell_1 : \{y_3 \cdot {y_1}^{y_2} = {x_1}^{x_2} \wedge y_1, y_2, y_3 \in \mathbb{N} \wedge z \in \mathbb{Z}\}$
while $y_2 \neq 0$ **do**
  $\ell_2 : \{y_2 \neq 0 \wedge y_3 \cdot {y_1}^{y_2} = {x_1}^{x_2} \wedge y_1, y_2, y_3 \in \mathbb{N} \wedge z \in \mathbb{Z}\}$
  if $impair(y_2)$ **then**
    $\ell_3 : \{impair(y_2) \wedge y_2 \neq 0 \wedge \boxed{\bullet\bullet\bullet} \wedge y_1, y_2, y_3 \in \mathbb{N} \wedge z \in \mathbb{Z}\}$
    $y_2 := y_2 - 1$;
    $\ell_4 : \{y_2 \geq 0 \wedge pair(y_2) \wedge y_3 \cdot y_1 \cdot {y_1}^{y_2} = {x_1}^{x_2} \wedge y_1, y_2, y_3 \in \mathbb{N} \wedge z \in \mathbb{Z}\}$
    $y_3 := y_3 \cdot y_1$;
    $\ell_5 : \{y_2 \geq 0 \wedge pair(y_2) \wedge y_3 \cdot {y_1}^{y_2} = {x_1}^{x_2} \wedge y_1, y_2, y_3 \in \mathbb{N} \wedge z \in \mathbb{Z}\}$
  ;
  $\ell_6 : \{y_2 \geq 0 \wedge pair(y_2) \wedge \boxed{\bullet\bullet\bullet} \wedge y_1, y_2, y_3 \in \mathbb{N} \wedge z \in \mathbb{Z}\}$
  $y_1 := y_1 \cdot y_1$;
  $\ell_7 : \{y_2 \geq 0 \wedge pair(y_2) \wedge y_3 \cdot {y_1}^{y_2\ div 2} = {x_1}^{x_2} \wedge y_1, y_2, y_3 \in \mathbb{N} \wedge z \in \mathbb{Z}\}$
  $y_2 := y_2\ div\ 2$;
  $\ell_8 : \{y_2 \geq 0 \wedge \boxed{\bullet\bullet\bullet} \wedge y_1, y_2, y_3 \in \mathbb{N} \wedge z \in \mathbb{Z}\}$
;
$\ell_9 : \{y_2 = 0 \wedge \boxed{\bullet\bullet\bullet} \wedge y_1, y_2, y_3 \in \mathbb{N} \wedge z \in \mathbb{Z}\}$
$z := y_3$;
$\ell_{10} : \{y_2 = 0 \wedge y_3 \cdot {y_1}^{y_2} = {x_1}^{x_2} \wedge y_1, y_2, y_3 \in \mathbb{N} \wedge z \in \mathbb{Z} \wedge z = {x_1}^{x_2}\}$

Algorithme 2: Algorithme de l'exponentitaion indienne annoté

```
                x2  =  v;   2nd integer
                y1  =  0;
                y2 = 0;
                y3 = 0;
                z  = 0;

{
   l1 :  print ⟨x1,  x2⟩;

   y1 := x1;
   y2 := x2;
   y3 := 1;
  l2 : while (y2 / =  0) {

      L3 : if ( y2 % 2  ≠  0) {
         l4 : y2 := y2−1;
         l5 : y3 := y3·y1;
      };
      l6 : y1  :=  y1·y1;
      l7 : y2 :=  y2  ÷  2;
   };
   l8 :  z  :=  y3;
   l9 :  print ⟨x1,  x2, z⟩;
}

}
·)
```

Modification History

Last modified Fri Feb 26 06 :26 :27 CET 2016 by mery

Created Wed Sep 09 17 :02 :47 CEST 2015 by mery

La traduction par l'outil donne alors le module suivant :

MODULE $tp3$

```
EXTENDS Naturals, Integers, TLC

CONSTANT MAXINT, u, v

(·
−termination
−wfNext
−−algorithm Power {
    variables x1  = u;   1st integer
                x2  =  v;   2nd integer
                y1  =  0;
                y2 = 0;
                y3 = 0;
                z  = 0;

{
   l1 :  print ⟨x1,  x2⟩;
```

```
    y1 := x1;
    y2 := x2;
    y3 := 1;
   l2 : while (y2 / = 0) {

        L3 : if ( y2 % 2 ≠ 0) {
            l4 : y2 := y2−1;
            l5 : y3 := y3·y1;
        };
        l6 : y1 := y1·y1;
        l7 : y2 := y2 ÷ 2;
    };
    l8 : z := y3;
    l9 : print ⟨x1, x2, z⟩;
}

}
·)
```

BEGIN TRANSLATION

VARIABLES $x_1,\ x_2,\ y_1,\ y_2,\ y_3,\ z,\ pc$

$vars \triangleq \langle\ x_1,\ x_2,\ y_1,\ y_2,\ y_3,\ z,\ pc\ \rangle$

$Init \triangleq$ Global variables

$\land\ x_1 = u$
$\land\ x_2 = v$
$\land\ y_1 = 0$
$\land\ y_2 = 0$
$\land\ y_3 = 0$
$\land\ z = 0$
$\land\ pc = \text{"l1"}$

$l1 \triangleq \land\ pc = \text{"l1"}$
$\land\ PrintT(\langle x_1,\ x_2\rangle)$
$\land\ y_1' = x1$
$\land\ y_2' = x2$
$\land\ y_3' = 1$
$\land\ pc' = \text{"l2"}$
$\land\ \text{UNCHANGED}\ \langle\ x_1,\ x_2,\ z\ \rangle$

$l2 \triangleq \land\ pc = \text{"l2"}$
$\land\ \text{IF}\ y_2 / = 0$
$\text{THEN}\ \land\ pc' = \text{"L3"}$
$\text{ELSE}\ \land\ pc' = \text{"l8"}$
$\land\ \text{UNCHANGED}\ \langle\ x_1,\ x_2,\ y_1,\ y_2,\ y_3,\ z\ \rangle$

$L3 \triangleq \land\ pc = \text{"L3"}$
$\land\ \text{IF}\ y_2\ \%\ 2 \neq 0$
$\text{THEN}\ \land\ pc' = \text{"l4"}$
$\text{ELSE}\ \land\ pc' = \text{"l6"}$
$\land\ \text{UNCHANGED}\ \langle\ x_1,\ x_2,\ y_1,\ y_2,\ y_3,\ z\ \rangle$

$l4 \triangleq \land\ pc = \text{"l4"}$
$\land\ y_2' = y_2−1$
$\land\ pc' = \text{"l5"}$
$\land\ \text{UNCHANGED}\ \langle\ x_1,\ x_2,\ y_1,\ y_3,\ z\ \rangle$

$$
\begin{aligned}
l5 \triangleq\ & \land pc = \text{"l5"} \\
& \land y_3' = y_3 \cdot y1 \\
& \land pc' = \text{"l6"} \\
& \land \text{UNCHANGED} \langle x_1, x_2, y_1, y_2, z \rangle
\end{aligned}
$$

$$
\begin{aligned}
l6 \triangleq\ & \land pc = \text{"l6"} \\
& \land y_1' = y_1 \cdot y1 \\
& \land pc' = \text{"l7"} \\
& \land \text{UNCHANGED} \langle x_1, x_2, y_2, y_3, z \rangle
\end{aligned}
$$

$$
\begin{aligned}
l7 \triangleq\ & \land pc = \text{"l7"} \\
& \land y_2' = (y_2 \div 2) \\
& \land pc' = \text{"l2"} \\
& \land \text{UNCHANGED} \langle x_1, x_2, y_1, y_3, z \rangle
\end{aligned}
$$

$$
\begin{aligned}
l8 \triangleq\ & \land pc = \text{"l8"} \\
& \land z' = y3 \\
& \land pc' = \text{"l9"} \\
& \land \text{UNCHANGED} \langle x_1, x_2, y_1, y_2, y_3 \rangle
\end{aligned}
$$

$$
\begin{aligned}
l9 \triangleq\ & \land pc = \text{"l9"} \\
& \land PrintT(\langle x_1, x_2, z \rangle) \\
& \land pc' = \text{"Done"} \\
& \land \text{UNCHANGED} \langle x_1, x_2, y_1, y_2, y_3, z \rangle
\end{aligned}
$$

$$
\begin{aligned}
Next \triangleq\ & l_1 \lor l_2 \lor L_3 \lor l_4 \lor l_5 \lor l_6 \lor l_7 \lor l_8 \lor l9 \\
& \lor \text{Disjunct to prevent deadlock on termination} \\
& \quad (pc = \text{"Done"} \land \text{UNCHANGED } vars)
\end{aligned}
$$

$$Spec \triangleq Init \land \Box[Next]_{vars}$$

$$Termination \triangleq \Diamond(pc = \text{"Done"})$$

END TRANSLATION

Modification History

Last modified Fri Feb 26 06 :26 :27 CET 2016 by mery

Created Wed Sep 09 17 :02 :47 CEST 2015 by mery

Fin 6

Exercice 7 *Vérifier le programme C suivant en la traduisant dans un module PlusCal. Pour cela, on conservera les éléments algorithmiques suffisants pour analyser le programme C. Par exemple, on ne traduira pas les instructions de lecture ou les inclusions de bibliothèques.*

```
/* C Program to find roots of a quadratic equation when coefficients are entered by
/* Library function sqrt() computes the square root. */

#include <stdio.h>
#include <math.h> /* This is needed to use sqrt() function.*/
int main()
```

```
{
  float a, b, c, determinant, r1,r2, real, imag;
  printf("Enter coefficients a, b and c: ");
  scanf("%f%f%f",&a,&b,&c);
  determinant=b*b-4*a*c;
  if (determinant>0)
  {
      r1= (-b+sqrt(determinant))/(2*a);
      r2= (-b-sqrt(determinant))/(2*a);
      printf("Roots are: %.2f and %.2f",r1 , r2);
  }
  else if (determinant==0)
  {
    r1 = r2 = -b/(2*a);
    printf("Roots are: %.2f and %.2f", r1, r2);
  }
  else
  {
    real= -b/(2*a);
    imag = sqrt(-determinant)/(2*a);
    printf("Roots are: %.2f+%.2fi and %.2f-%.2fi", real, imag, real, imag);
  }
  return 0;
}
```

Output 1

```
Enter coefficients a, b and c: 2.3
4
5.6
Roots are: -0.87+1.30i and -0.87-1.30i
```

Output 2

```
Enter coefficients a, b and c: 4
1
0
Roots are: 0.00 and -0.25
```

To solve this program, library function sqrt() is used. This function calculates th
sqrt() function.
Similar C Programming Examples
C Program to Add Two Complex Numbers by Passing Structure to a Function
C Program to Find Quotient and Remainder of Two Integers Entered by User
C Program to Check Prime or Armstrong Number Using User-defined Function
C Program to Display Prime Numbers Between Intervals Using User-defined Function
C Program to Display its own Source Code as Output

◇– **Solution de l'exercice 7**

MODULE $tp7$

EXTENDS $TLC, Integers, Naturals, Reals$

CONSTANTS $a0, b0, c0$

```
(·

--algorithm resolution {
variables a = a_0, b = b_0, c = c_0, determinant;
{

  l0 : determinant := b·b−4·a·c;
  if (determinant > 0)
  {

     l1 : print ⟨"Two Roots are : "⟩;
  }
  else if (determinant = 0)
  {

    l2 : print ⟨"One Double roots are : "⟩;
  }
  else
  {

    l3 : print ⟨"Two complex Roots are : "⟩;
  }
  }
}
·)
```

BEGIN TRANSLATION

```
CONSTANT defaultInitValue
VARIABLES a, b, c, determinant, pc

vars ≜ ⟨ a, b, c, determinant, pc ⟩

Init ≜   Global variables
        ∧ a = a0
        ∧ b = b0
        ∧ c = c0
        ∧ determinant = defaultInitValue
        ∧ pc = "l0"

l0 ≜ ∧ pc = "l0"
     ∧ determinant' = b·b−4·a·c
     ∧ IF determinant' > 0
          THEN ∧ pc' = "l1"
          ELSE ∧ IF determinant' = 0
                    THEN ∧ pc' = "l2"
                    ELSE ∧ pc' = "l3"
     ∧ UNCHANGED ⟨ a, b, c ⟩

l1 ≜ ∧ pc = "l1"
     ∧ PrintT(⟨"Two Roots are : "⟩)
     ∧ pc' = "Done"
     ∧ UNCHANGED ⟨ a, b, c, determinant ⟩

l2 ≜ ∧ pc = "l2"
     ∧ PrintT(⟨"One Double roots are : "⟩)
     ∧ pc' = "Done"
     ∧ UNCHANGED ⟨ a, b, c, determinant ⟩
```

$l3 \triangleq \land pc = \text{"l3"}$
$\land PrintT(\langle \text{"}Two\ complex\ Roots\ are : \text{"} \rangle)$
$\land pc' = \text{"Done"}$
$\land \text{UNCHANGED } \langle a, b, c, determinant \rangle$

$Next \triangleq l_0 \lor l_1 \lor l_2 \lor l3$
$\lor$ Disjunct to prevent deadlock on termination
$(pc = \text{"Done"} \land \text{UNCHANGED } vars)$

$Spec \triangleq Init \land \Box[Next]_{vars}$

$Termination \triangleq \Diamond(pc = \text{"Done"})$

END TRANSLATION

Modification History
Last modified Mon Nov 23 11 :14 :11 CET 2015 by mery
Created Wed Nov 18 16 :33 :27 CET 2015 by mery

Fin 7

Exercice 8 ☑
Vérifier le programme C suivant en la traduisant dans un module PlusCal.

```c
/* C program to check whether a number is palindrome or not */

#include <stdio.h>
int main()
{
  int n, reverse=0, rem,temp;
  printf("Enter an integer: ");
  scanf("%d", &n);
  temp=n;
  while(temp!=0)
  {
     rem=temp%10;
     reverse=reverse*10+rem;
     temp/=10;
  }
/* Checking if number entered by user and it's reverse number is equal. */
  if(reverse==n)
       printf("%d is a palindrome.",n);
  else
       printf("%d is not a palindrome.",n);
  return 0;
}
```

◇– **Solution de l'exercice 8**

MODULE *pluscal5*

EXTENDS *TLC*, *Integers*, *Naturals*
CONSTANTS $n0$

(·

```
--algorithm compte{
variables n = n_0, reverse = 0, rem, temp;
{
  temp := n;
  while (temp ≠ 0)
  {
    rem  := temp % 10;
    reverse := reverse·10+rem;
    temp  := temp ÷ 10;
  };
  if (reverse = n) {
    print ⟨"palin", n⟩;
    }
    else
    {
    print ⟨"not palin", n⟩;
    }

}
}

.)
```

BEGIN TRANSLATION

CONSTANT *defaultInitValue*
VARIABLES n, *reverse*, *rem*, *temp*, *pc*

$vars \triangleq \langle n, reverse, rem, temp, pc \rangle$

$Init \triangleq$ Global variables
$\land n = n0$
$\land reverse = 0$
$\land rem = defaultInitValue$
$\land temp = defaultInitValue$
$\land pc = "Lbl\_1"$

$Lbl_1 \triangleq \land pc = "Lbl\_1"$
$\land temp' = n$
$\land pc' = "Lbl\_2"$
$\land$ UNCHANGED $\langle n, reverse, rem \rangle$

$Lbl_2 \triangleq \land pc = "Lbl\_2"$
$\land$ IF $temp \neq 0$
THEN $\land rem' = temp \% 10$
$\land reverse' = reverse·10+rem'$
$\land temp' = (temp \div 10)$
$\land pc' = "Lbl\_2"$
ELSE $\land$ IF $reverse = n$
THEN $\land PrintT(\langle$"palin", $n\rangle)$
ELSE $\land PrintT(\langle "not\ palin", n\rangle)$
$\land pc' =$ "Done"
$\land$ UNCHANGED $\langle reverse, rem, temp \rangle$
$\land n' = n$

$Next \triangleq Lbl_1 \lor Lbl\_2$
$\lor$ Disjunct to prevent deadlock on termination
(*pc* = "Done" $\land$ UNCHANGED *vars*)

$Spec \triangleq Init \wedge \Box[Next]_{vars}$

$Termination \triangleq \Diamond(pc = \text{"Done"})$

END TRANSLATION

Fin 8

Exercice 9 *Vérifier le programme C suivant en la traduisant dans un module PlusCal.*

```
#include <stdio.h>
int main()
{
   int n,count=0;
   printf("Enter an integer: ");
   scanf("%d", &n);
   while(n!=0)
   {
        n/=10;                  /* n=n/10 */
        ++count;
   }
   printf("Number of digits: %d",count);
}
```

◇– **Solution de l'exercice 9**

MODULE $tp9$

EXTENDS $TLC, Integers, Naturals$
CONSTANTS $n0$

(·
––*algorithm compte*{
variables n, *count* $= 0$;
{
$l0 : n := n_0$;
$l1$: *while* $(n \neq 0)$
{
$l2 : n := n \div 10$;
$l3$: *count* := *count* $+1$;
};
$l4$: *print* $\langle$"$Number$ *of digits* : $\%d$", *count*$\rangle$;
}
}

·)

BEGIN TRANSLATION
CONSTANT *defaultInitValue*
VARIABLES n, *count*, *pc*

$vars \triangleq \langle n, count, pc \rangle$

$Init \triangleq$ Global variables
$\wedge n = defaultInitValue$

$\land$ *count* $=$ 0
$\land$ *pc* $=$ "l0"

$l0 \triangleq$ $\land$ *pc* $=$ "l0"
$\land$ n' $=$ $n0$
$\land$ *pc*$'$ $=$ "l1"
$\land$ *count*$'$ $=$ *count*

$l1 \triangleq$ $\land$ *pc* $=$ "l1"
$\land$ IF $n \neq 0$
THEN $\land$ *pc*$'$ $=$ "l2"
ELSE $\land$ *pc*$'$ $=$ "l4"
$\land$ UNCHANGED $\langle$ n, *count* $\rangle$

$l2 \triangleq$ $\land$ *pc* $=$ "l2"
$\land$ n' $=$ $(n \div 10)$
$\land$ *pc*$'$ $=$ "l3"
$\land$ *count*$'$ $=$ *count*

$l3 \triangleq$ $\land$ *pc* $=$ "l3"
$\land$ *count*$'$ $=$ *count* $+1$
$\land$ *pc*$'$ $=$ "l1"
$\land$ n' $=$ n

$l4 \triangleq$ $\land$ *pc* $=$ "l4"
$\land$ *PrintT*($\langle$"$Number$ *of digits* : $\%d$", *count*$\rangle$)
$\land$ *pc*$'$ $=$ "Done"
$\land$ UNCHANGED $\langle$ n, *count* $\rangle$

Next $\triangleq$ $l_0 \lor l_1 \lor l_2 \lor l_3 \lor l4$
$\lor$ Disjunct to prevent deadlock on termination
(*pc* $=$ "Done" $\land$ UNCHANGED *vars*)

Spec $\triangleq$ *Init* $\land$ $\Box[$*Next*$]_{vars}$

Termination $\triangleq$ $\Diamond$(*pc* $=$ "Done")

END TRANSLATION

test$_1$ $\triangleq$ *pc* $=$ "Done" $\land$ $n_0 = 345$ $\Rightarrow$ *count* $=$ 3
test$_2$ $\triangleq$ *pc* $=$ "Done" $\land$ $n_0 = 0$ $\Rightarrow$ *count* $=$ 0
safe $\triangleq$ $n \neq$ *defaultInitValue* $\Rightarrow$ $0 \leq n \land n \leq n0$

Fin 9

Exercice 10 *Nous avons récupéré un programme C qui réalise le tri à bulles. Traduire ce programme en PlusCal et analyser sa correction partielle et l'ansence d'erreurs à l'exécution.*

```
void tri_a_bulle(int* t, int const size)
{
        int en_desordre = 1;
        int i,j;
```

```
        for (i = 0; (i < size) && en_desordre; ++i)
        {
                en_desordre = 0;
                for (j = 1; j < (size - i); ++j)
                {
                        if (t[j-1] > t[j])
                        {
                                int temp = t[j-1];
                                t[j-1] = t[j];
                                t[j] = temp;
                                en_desordre = 1;
                        }
                }
        }
}
```

```
    #include<algorithm> // pour la fonction swap

void tri_a_bulle(int *t, int const size)
{
        bool en_desordre = true;
        for (int i = 0; i < size && en_desordre; ++i)
        {
                en_desordre = false;
                for (int j = 1; j < size - i; ++j)
                        if (t[j-1] > t[j])
                        {
                                std::swap(t[j], t[j-1]);
                                en_desordre = true;
                        }
        }
}
```

◇– Solution de l'exercice 10

MODULE $tp10$

EXTENDS *TLC*, *Integers*, *Naturals*

$n \triangleq 5$
$t0 \triangleq [k \in 0..n-1 \mapsto$
IF $k = 0$ THEN 3
ELSE IF $k = 1$ THEN 6
ELSE IF $k = 2$ THEN $2 \cdot k$
ELSE IF $k = 3$ THEN 9
ELSE 5]

```
(*
--algorithm tribulle {
variables t, i, j, d = TRUE, temp;
{
   t := t0;
   i := 0;
   while(i < n ∧ d)
       {
            d := FALSE;
            j := 1;
```

```
            while ( j < (n − i))
            {
                if (t[j−1] > t[j])
                {
                    temp := t[j−1];
                    t[j−1] := t[j];
                    t[j] := temp;
                    d := TRUE;
                };
                j := j+1;
            };
            i := i+1;
        }
}

}

.)
```

BEGIN TRANSLATION

```
CONSTANT defaultInitValue
VARIABLES t, i, j, d, temp, pc

vars ≜ ⟨ t, i, j, d, temp, pc ⟩

Init ≜   Global variables
        ∧ t = defaultInitValue
        ∧ i = defaultInitValue
        ∧ j = defaultInitValue
        ∧ d = TRUE
        ∧ temp = defaultInitValue
        ∧ pc = "Lbl_1"

Lbl_1 ≜ ∧ pc = "Lbl_1"
        ∧ t' = t0
        ∧ i' = 0
        ∧ pc' = "Lbl_2"
        ∧ UNCHANGED ⟨ j, d, temp ⟩

Lbl_2 ≜ ∧ pc = "Lbl_2"
        ∧ IF i < n ∧ d
            THEN ∧ d' = FALSE
                 ∧ j' = 1
                 ∧ pc' = "Lbl_3"
            ELSE ∧ pc' = "Done"
                 ∧ UNCHANGED ⟨ j, d ⟩
        ∧ UNCHANGED ⟨ t, i, temp ⟩

Lbl_3 ≜ ∧ pc = "Lbl_3"
        ∧ IF j < (n − i)
            THEN ∧ IF t[j−1] > t[j]
                    THEN ∧ temp' = t[j−1]
                         ∧ t' = [t EXCEPT ![j−1] = t[j]]
                         ∧ pc' = "Lbl_4"
                    ELSE ∧ pc' = "Lbl_5"
                         ∧ UNCHANGED ⟨ t, temp ⟩
                 ∧ i' = i
```

$$
\begin{array}{l}
\qquad\qquad \text{ELSE} \ \land\ i' \ = \ i{+}1 \\
\qquad\qquad\qquad\ \land\ pc' \ = \ \text{"}Lbl\_2\text{"} \\
\qquad\qquad\qquad\ \land\ \text{UNCHANGED} \ \langle\ t,\ temp\ \rangle \\
\qquad\quad \land\ \text{UNCHANGED} \ \langle\ j,\ d\ \rangle
\end{array}
$$

$$
\begin{array}{l}
Lbl\_5 \ \triangleq\ \land\ pc \ = \ \text{"}Lbl\_5\text{"} \\
\qquad\qquad\ \land\ j' \ = \ j{+}1 \\
\qquad\qquad\ \land\ pc' \ = \ \text{"}Lbl\_3\text{"} \\
\qquad\qquad\ \land\ \text{UNCHANGED} \ \langle\ t,\ i,\ d,\ temp\ \rangle
\end{array}
$$

$$
\begin{array}{l}
Lbl\_4 \ \triangleq\ \land\ pc \ = \ \text{"}Lbl\_4\text{"} \\
\qquad\qquad\ \land\ t' \ = \ [t \ \text{EXCEPT} \ ![j] \ = \ temp] \\
\qquad\qquad\ \land\ d' \ = \ \mathit{TRUE} \\
\qquad\qquad\ \land\ pc' \ = \ \text{"}Lbl\_5\text{"} \\
\qquad\qquad\ \land\ \text{UNCHANGED} \ \langle\ i,\ j,\ temp\ \rangle
\end{array}
$$

$$
\begin{array}{l}
Next \ \triangleq\ Lbl\_1 \ \lor\ Lbl\_2 \ \lor\ Lbl\_3 \ \lor\ Lbl\_5 \ \lor\ Lbl\_4 \\
\qquad\quad \lor \quad \text{Disjunct to prevent deadlock on termination} \\
\qquad\qquad (pc \ = \ \text{"Done"} \ \land\ \text{UNCHANGED} \ vars)
\end{array}
$$

$$Spec \ \triangleq\ Init \ \land\ \Box[Next]_{vars}$$

$$Termination \ \triangleq\ \Diamond(pc \ = \ \text{"Done"})$$

END TRANSLATION

$$
\begin{array}{l}
Safe_1 \ \triangleq\ pc \ \neq\ \text{"Done"} \\
Safe_2 \ \triangleq\ pc = \text{"Done"} \ \Rightarrow\ \forall\, e \ \in\ 0..n{-}1 : \ t[e] \ \leq\ t[e{+}1] \\
Safe_3 \ \triangleq\ pc = \text{"Done"} \ \Rightarrow\ \{\, e \ \in\ 0..n{-}1 : \ t[e] \ \leq\ t[e{+}1]\} \ = \ 0..n{-}1 \\
Safe_4 \ \triangleq\ pc = \text{"Done"} \ \Rightarrow\ (t[0] \ \leq\ t[1]) \ \land\ (t[1] \ \leq\ t[2])
\end{array}
$$

Fin 10

Exercice 11 *Analyser le programme C suivant qui réalise le tri par sélection.*

```
      void selection(int *t, int taille)
{
      int i, mini, j, x;

      for (i = 0; i < taille - 1; i++)
      {
           mini = i;
           for (j = i + 1; j < taille; j++)
                if (t[j] < t[mini])
                     mini = j;
            x = t[i];
            t[i] = t[mini];
            t[mini] = x;
      }
}
```

◇– Solution de l'exercice 11

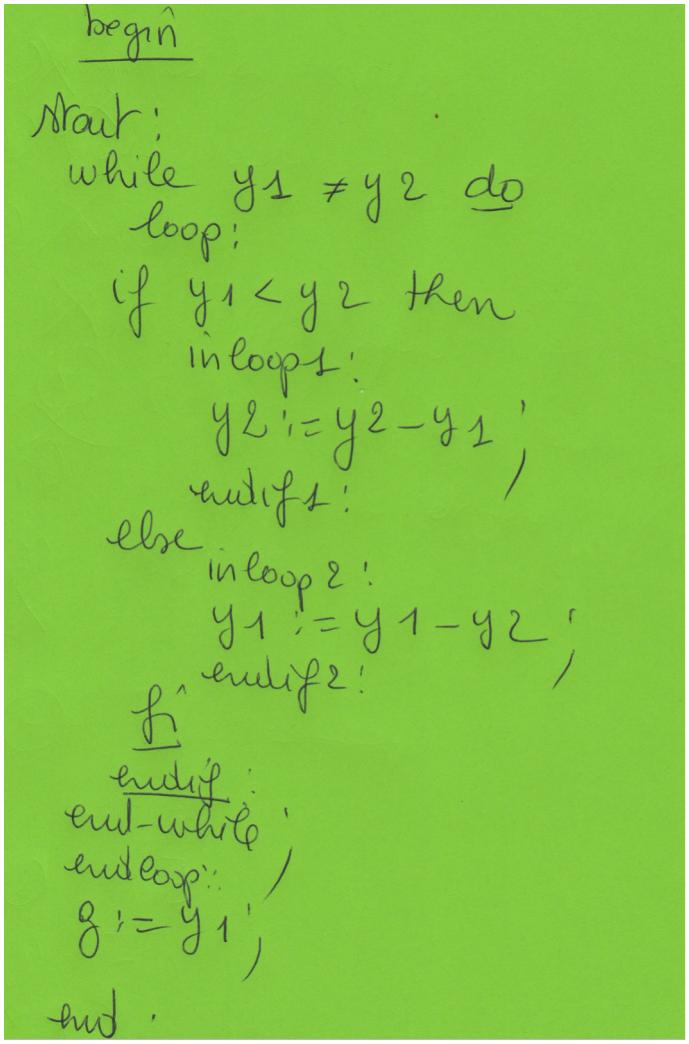
MODULE $tp11$

EXTENDS $TLC, Integers, Naturals$

```
CONSTANTS n0
(·
−−algorithm trisel {
variables
{

   void selection(int ·t, int taille)
{
    int i, mini, j, x;

    for (i = 0; i < taille − 1; i++)
    {
       mini = i;
       for (j = i + 1; j < taille; j++)
            if (t[j] < t[mini])
                mini = j;
        x = t[i];
        t[i] = t[mini];
        t[mini] = x;
    }

}

·)
```

Fin 11

Exercice 12 *GCD en Event-B*
Soit l'algorithme GCD calculant le pgcd de deux nombres entiers positifs quelconques a *et* b.



Question 12.1 *Traduire l'algorithme en un module TLA$^+$ qui précise les annotations et qui précise les invariants pour la correction partielle et l'absence d'erreurs à ml'exécution.*

◇– **Solution de la question 12.1**

MODULE *gcd*

EXTENDS *Integers*, *TLC*
CONSTANTS a, b
VARIABLES y_1 , y_2 , z, l

$Init \triangleq y_1 = a \land y_2 = b \land z = 0 \land l = \text{"start"}$
$L1 \triangleq l = \text{"start"} \land y_1 \neq y_2 \land l' = \text{"loop"} \land \text{UNCHANGED} \langle y_1 , y_2 , z \rangle$
$L2 \triangleq l = \text{"start"} \land y_1 = y_2 \land l' = \text{"outloop"} \land \text{UNCHANGED} \langle y_1 , y_2 , z \rangle$
$L3 \triangleq l = \text{"loop"} \land y_1 < y_2 \land l' = \text{"inloop1"} \land \text{UNCHANGED} \langle y_1 , y_2 , z \rangle$
$L4 \triangleq l = \text{"inloop1"} \land y_2' = y_2 - y1 \land l' = \text{"endif1"} \land \text{UNCHANGED} \langle y_1 , z \rangle$
$L5 \triangleq l = \text{"endif1"} \land l' = \text{"endif"} \land \text{UNCHANGED} \langle y_1 , y_2, z \rangle$
$L6 \triangleq l = \text{"loop"} \land y_1 > y_2 \land l' = \text{"inloop2"} \land \text{UNCHANGED} \langle y_1 , y_2 , z \rangle$
$L7 \triangleq l = \text{"inloop2"} \land y_1' = y_1 - y2 \land l' = \text{"endif2"} \land \text{UNCHANGED} \langle y_2 , z \rangle$
$L8 \triangleq l = \text{"endif2"} \land l' = \text{"endif"} \land \text{UNCHANGED} \langle y_1 , y_2, z \rangle$
$L9 \triangleq l = \text{"endif"} \land y_1 \neq y_2 \land l' = \text{"loop"} \land \text{UNCHANGED} \langle y_1 , y_2, z \rangle$
$L10 \triangleq l = \text{"endif"} \land y_1 = y_2 \land l' = \text{"endloop"} \land \text{UNCHANGED} \langle y_1 , y_2, z \rangle$

$L11 \triangleq l = \text{"endloop"} \wedge z' = y_1 \wedge l' = \text{"end"} \wedge \text{UNCHANGED} \langle y_1 , y_2 \rangle$

$\mathit{Next} \triangleq \vee L_1 \vee L_2 \vee L_3 \vee L_4 \vee L5$
$\quad \vee L_6 \vee L_7 \vee L_8 \vee L_9 \vee L_{10} \vee L11$
$\mathit{safe}_1 \triangleq 0 \leq y_1 \wedge y_1 \leq a \wedge 0 \leq y_2 \wedge y_2 \leq b$
$\mathit{safe}_2 \triangleq y_1 \neq y2$
$\mathit{safe}_3 \triangleq l \neq \text{"end"}$
$\mathit{safe}_4 \triangleq l \neq \text{"endif2"}$

Fin 12.1

Question 12.2 *Construire une machine Event-B qui liste les actions de l'algorithme et qui donne l'invariant pour la correction partielle et l'absence d'erreurs à l'exécution.*