Cours Modélisation et vérification des systèmes informatiques
Exercices (avec les corrections)
Modélisation d'algorithmes en PlusCal (II)
par Dominique Méry
20 novembre 2024

Exercice 1 *(Vérification de l'annotation de l'algorithme du calcul du maximum d'une liste) appex5_1.tla*

Question 1.1 *Ecrire un module TLA$^+$ contenant une définition PlusCal de cet algorithme.*

Question 1.2 *Ecrire la propriété à vérifier pour la correction partielle.*

Question 1.3 *Ecrire la propriété à vérifier pour l'absence d'erreurs à l'exécution.*

Vérification **precondition** : $\left(\begin{array}{l} n \in \mathbb{N} \wedge \\ n \neq 0 \wedge \\ f \in 0 \,..\, n-1 \rightarrow \mathbb{N} \end{array}\right)$

postcondition : $\left(\begin{array}{l} m \in \mathbb{N} \wedge \\ m \in ran(f) \wedge \\ (\forall j \cdot j \in 0 \,..\, n-1 \Rightarrow f(j) \leq m) \end{array}\right)$

local variables : $i \in \mathbb{Z}$

$m := f(0);$
$i := 1;$
while $i < n$ **do**
  if *f(i)>m* **then**
    $m := f(i);$
  ;
  $i{+}{+};$
;

Algorithme 1: Algorithme du maximum d'une liste non annotée

Listing 1 – appex5-1.tla

```
----------------------- MODULE appex5_1 ----------------------------------
EXTENDS Naturals, Integers, TLC

CONSTANT n0

f0 == [k \in 0..n0-1 |->
                     IF k=0 THEN 3
                     ELSE IF k=1 THEN 6
                     ELSE IF k=2 THEN 2*k
                     ELSE IF k=3 THEN 9
                     ELSE 5]


(*
-termination
-wfNext
--algorithm Maximum {
  variables i = 0;
```

/* algorithme de calcul du maximum avec une boucle while de l'exercice **??** */

precondition : $\left(\begin{array}{l} n \in \mathbb{N} \wedge \\ n \neq 0 \wedge \\ f \in 0 \mathinner{..} n{-}1 \rightarrow \mathbb{N} \end{array}\right)$

postcondition : $\left(\begin{array}{l} m \in \mathbb{N} \wedge \\ m \in ran(f) \wedge \\ (\forall j \cdot j \in 0 \mathinner{..} n{-}1 \Rightarrow f(j) \leq m) \end{array}\right)$

local variables : $i \in \mathbb{Z}$

$\ell_0 : \{ \left(\begin{array}{l} n \in \mathbb{N} \wedge \\ n \neq 0 \wedge \\ f \in 0 \mathinner{..} n{-}1 \rightarrow \mathbb{N} \end{array}\right) \wedge i \in \mathbb{Z} \wedge i \in \mathbb{Z} \wedge ...\}$

$m := f(0);$

$\ell_1 : \{ \left(\begin{array}{l} n \in \mathbb{N} \wedge \\ n \neq 0 \wedge \\ f \in 0 \mathinner{..} n{-}1 \rightarrow \mathbb{N} \end{array}\right) \wedge i \in \mathbb{Z} \wedge m = f(0)\}$

$i := 1;$

$\ell_2 : \{ \left(\begin{array}{l} n \in \mathbb{N} \wedge \\ n \neq 0 \wedge \\ f \in 0 \mathinner{..} n{-}1 \rightarrow \mathbb{N} \end{array}\right) \wedge i = 1 \wedge \left(\begin{array}{l} m \in \mathbb{N} \wedge \\ m \in ran(f[0..i{-}1]) \wedge \\ (\forall j \cdot j \in 0 \mathinner{..} i{-}1 \Rightarrow f(j) \leq m) \end{array}\right)\}$

while $i < n$ **do**

$\ell_3 : \{ \left(\begin{array}{l} n \in \mathbb{N} \wedge \\ n \neq 0 \wedge \\ f \in 0 \mathinner{..} n{-}1 \rightarrow \mathbb{N} \end{array}\right) \wedge i \in 1..n{-}1 \wedge \left(\begin{array}{l} m \in \mathbb{N} \wedge \\ m \in ran(f[0..i{-}1]) \wedge \\ (\forall j \cdot j \in 0 \mathinner{..} i{-}1 \Rightarrow f(j) \leq m) \end{array}\right)\}$

if *f(i)>m* **then**

$\ell_4 : \{ \left(\begin{array}{l} n \in \mathbb{N} \wedge \\ n \neq 0 \wedge \\ f \in 0 \mathinner{..} n{-}1 \rightarrow \mathbb{N} \end{array}\right) \wedge i \in 1..n{-}1 \wedge \left(\begin{array}{l} m \in \mathbb{N} \wedge \\ m \in ran(f[0..i{-}1]) \wedge \\ (\forall j \cdot j \in 0 \mathinner{..} i{-}1 \Rightarrow f(j) \leq m) \end{array}\right) \wedge$
$f(i) > m\}$

$m := f(i);$

$\ell_5 : \{ \left(\begin{array}{l} n \in \mathbb{N} \wedge \\ n \neq 0 \wedge \\ f \in 0 \mathinner{..} n{-}1 \rightarrow \mathbb{N} \end{array}\right) \wedge i \in 1..n{-}1 \wedge \left(\begin{array}{l} m \in \mathbb{N} \wedge \\ m \in ran(f[0..i]) \wedge \\ (\forall j \cdot j \in 0 \mathinner{..} i \Rightarrow f(j) \leq m) \end{array}\right)\}$

;

$\ell_6 : \{ \left(\begin{array}{l} n \in \mathbb{N} \wedge \\ n \neq 0 \wedge \\ f \in 0 \mathinner{..} n{-}1 \rightarrow \mathbb{N} \end{array}\right) \wedge i \in \mathbb{Z} \wedge \wedge i \in 1..n{-}1 \wedge \left(\begin{array}{l} m \in \mathbb{N} \wedge \\ m \in ran(f[0..i]) \wedge \\ (\forall j \cdot j \in 0 \mathinner{..} i \Rightarrow f(j) \leq m) \end{array}\right)\}$

$i{+}{+};$

$\ell_7 : \{ \left(\begin{array}{l} n \in \mathbb{N} \wedge \\ n \neq 0 \wedge \\ f \in 0 \mathinner{..} n{-}1 \rightarrow \mathbb{N} \end{array}\right) \wedge i \in 1..n{-}1 \wedge \left(\begin{array}{l} m \in \mathbb{N} \wedge \\ m \in ran(f[0..i{-}1]) \wedge \\ (\forall j \cdot j \in 0 \mathinner{..} i{-}1 \Rightarrow f(j) \leq m) \end{array}\right)\}$

;

$\ell_8 : \{ \left(\begin{array}{l} n \in \mathbb{N} \wedge \\ n \neq 0 \wedge \\ f \in 0 \mathinner{..} n{-}1 \rightarrow \mathbb{N} \end{array}\right) \wedge i = n \wedge \left(\begin{array}{l} m \in \mathbb{N} \wedge \\ m \in ran(f) \wedge \\ (\forall j \cdot j \in 0 \mathinner{..} n{-}1 \Rightarrow f(j) \leq m) \end{array}\right)\}$

Algorithme 2: Algorithme du maximum d'une liste annoté

```
            m=0;
            f=f0;
            n=n0;
            r;
{
            l0:m:=f[0];
            l1:i:=1;
            l2: while (i<n) {
            l3: if (f[i]>m){
            l4: m:=f[i];
            } ;
            l5:i:=i+1;
            };
            r := m;
}
}
*)
```

===

Exercice 2 *Exponentiation appex5_2.tla*
Soit l'algorithme annoté calculant la puissance $z = {x_1}^{x_2}$.

- *Precondition :* $x_1 \in \mathbb{N} \wedge x_2 \wedge \mathbb{N}$
- *Postcondition :* $z = {x_1}^{x_2}$

On suppose que x_1 *et* x_2 *sont des constantes.*

Question 2.1 *Ecrire un module TLA/TLA*$^+$ *permettant de valider les conditions de vérification et, en particulier, de montrer la correction partielle.*

Question 2.2 *Modifier la machine pour prendre en compte l'absence d'erreurs à l'exécution.*

Listing 2 – appex5-2.tla

```
---------------------- MODULE appex5_2 --------------------------------
EXTENDS Naturals, Integers, TLC
---------------------------------------------
CONSTANT MAXINT, x10, x20, MININT
----------------------
typeInt(u) == u \in Int
pre == x10 \in Nat /\ x20 \in Nat /\ x10 # 0
----------------------------------------------------------------
(* precondition *)
ASSUME pre
-----------------------------------------------------
(*
--algorithm Exponentiation {
  variables
            x1=x10;
            x2=x20;
            y1;
            y2;
            y3;
            z;
{
     l0:
     y1:=x1; y2:=x2; y3:=1;
```

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

$\ell_0 : \{y_1, y_2, y_3, z \in \mathbb{Z}\}$
$y_1 := x_1; y_2 := x_2 : y_3 := 1;$
$\ell_1 : \{y_1 = x_1 \land y_2 = x_2 \land y_3 = 1 \land y_1, y_2, y_3 \in \mathbb{N} \land z \in \mathbb{Z}\}$
$\ell_{11} : \{y_3 \cdot {y_1}^{y_2} = {x_1}^{x_2} \land y_1, y_2, y_3 \in \mathbb{N} \land z \in \mathbb{Z}\}$
while $y_2 \neq 0$ **do**
    $\ell_2 : \{y_2 \neq 0 \land y_3 \cdot {y_1}^{y_2} = {x_1}^{x_2} \land y_1, y_2, y_3 \in \mathbb{N} \land z \in \mathbb{Z}\}$
    if $impair(y_2)$ **then**
        $\ell_3 : \{impair(y_2) \land y_2 \neq 0 \land y_3 \cdot {y_1}^{y_2} = {x_1}^{x_2} \land y_1, y_2, y_3 \in \mathbb{N} \land z \in \mathbb{Z}\}$
        $y_2 := y_2 - 1;$
        $\ell_4 : \{y_2 \geq 0 \land pair(y_2) \land y_3 \cdot y_1 \cdot {y_1}^{y_2} = {x_1}^{x_2} \land y_1, y_2, y_3 \in \mathbb{N} \land z \in \mathbb{Z}\}$
        $y_3 := y_3 \cdot y_1;$
        $\ell_5 : \{y_2 \geq 0 \land pair(y_2) \land y_3 \cdot {y_1}^{y_2} = {x_1}^{x_2} \land y_1, y_2, y_3 \in \mathbb{N} \land z \in \mathbb{Z}\}$
    ;
    $\ell_6 : \{y_2 \geq 0 \land pair(y_2) \land y_3 \cdot {y_1}^{y_2} = {x_1}^{x_2} \land y_1, y_2, y_3 \in \mathbb{N} \land z \in \mathbb{Z}\}$
    $y_1 := y_1 \cdot y_1;$
    $\ell_7 : \{y_2 \geq 0 \land pair(y_2) \land y_3 \cdot y_1 .^{y_2 \; div 2} = {x_1}^{x_2} \land y_1, y_2, y_3 \in \mathbb{N} \land z \in \mathbb{Z}\}$
    $y_2 := y_2 \; div \; 2;$
    $\ell_8 : \{y_2 \geq 0 \land y_3 \cdot {y_1}^{y_2} = {x_1}^{x_2} \land y_1, y_2, y_3 \in \mathbb{N} \land z \in \mathbb{Z}\}$
;
$\ell_9 : \{y_2 = 0 \land y_3 \cdot {y_1}^{y_2} = {x_1}^{x_2} \land y_1, y_2, y_3 \in \mathbb{N} \land z \in \mathbb{Z}\}$
$z := y_3;$
$\ell_{10} : \{y_2 = 0 \land y_3 \cdot {y_1}^{y_2} = {x_1}^{x_2} \land y_1, y_2, y_3 \in \mathbb{N} \land z \in \mathbb{Z} \land z = {x_1}^{x_2}\}$

Algorithme 3: Version solution annotée

```
    w:while (y2 /= 0) {

      l2:

if ( y2 % 2 # 0) {
        l3:y2:=y2-1;
        l4:y3:=y3*y1;
        l5:skip;
      };
      l6:y1 := y1*y1;      l7:y2:= y2 \div   2;
      l8:skip;
    };
    l9: z := y3;
    l10: print <<x1, x2,z>>;
}

}
*)
\* BEGIN TRANSLATION (chksum(pcal) = "14eb71f" /\ chksum(tla) = "f9286308")
CONSTANT defaultInitValue
VARIABLES x1, x2, y1, y2, y3, z, pc

vars == << x1, x2, y1, y2, y3, z, pc >>

Init == (* Global variables *)
        /\ x1 = x10
        /\ x2 = x20
        /\ y1 = defaultInitValue
        /\ y2 = defaultInitValue
        /\ y3 = defaultInitValue
        /\ z = defaultInitValue
        /\ pc = "l0"

l0 == /\ pc = "l0"
      /\ y1'␣=␣x1
␣␣␣␣␣␣/\␣y2' = x2
      /\ y3'␣=␣1
␣␣␣␣␣␣/\␣pc' = "w"
      /\ UNCHANGED << x1, x2, z >>

w == /\ pc = "w"
     /\ IF y2 /= 0
           THEN /\ pc'␣=␣"l2"
␣␣␣␣␣␣␣␣␣␣␣ELSE␣/\␣pc' = "l9"
     /\ UNCHANGED << x1, x2, y1, y2, y3, z >>

l2 == /\ pc = "l2"
      /\ IF y2 % 2 # 0
            THEN /\ pc'␣=␣"l3"
␣␣␣␣␣␣␣␣␣␣␣␣ELSE␣/\␣pc' = "l6"
      /\ UNCHANGED << x1, x2, y1, y2, y3, z >>

l3 == /\ pc = "l3"
      /\ y2'␣=␣y2-1
␣␣␣␣␣␣/\␣pc' = "l4"
      /\ UNCHANGED << x1, x2, y1, y3, z >>
```

```
l4 == /\ pc = "l4"
      /\ y3'␣=␣y3*y1
␣␣␣␣␣␣/\␣pc' = "l5"
      /\ UNCHANGED << x1, x2, y1, y2, z >>

l5 == /\ pc = "l5"
      /\ TRUE
      /\ pc'␣=␣"l6"
␣␣␣␣␣␣/\␣UNCHANGED␣<<␣x1,␣x2,␣y1,␣y2,␣y3,␣z␣>>

l6␣==␣/\␣pc␣=␣"l6"
␣␣␣␣␣␣/\␣y1' = y1*y1
      /\ pc'␣=␣"l7"
␣␣␣␣␣␣/\␣UNCHANGED␣<<␣x1,␣x2,␣y2,␣y3,␣z␣>>

l7␣==␣/\␣pc␣=␣"l7"
␣␣␣␣␣␣/\␣y2' = (y2 \div   2)
      /\ pc'␣=␣"l8"
␣␣␣␣␣␣/\␣UNCHANGED␣<<␣x1,␣x2,␣y1,␣y3,␣z␣>>

l8␣==␣/\␣pc␣=␣"l8"
␣␣␣␣␣␣/\␣TRUE
␣␣␣␣␣␣/\␣pc' = "w"
      /\ UNCHANGED << x1, x2, y1, y2, y3, z >>

l9 == /\ pc = "l9"
      /\ z'␣=␣y3
␣␣␣␣␣␣/\␣pc' = "l10"
      /\ UNCHANGED << x1, x2, y1, y2, y3 >>

l10 == /\ pc = "l10"
       /\ PrintT(<<x1, x2,z>>)
       /\ pc'␣=␣"Done"
␣␣␣␣␣␣␣/\␣UNCHANGED␣<<␣x1,␣x2,␣y1,␣y2,␣y3,␣z␣>>

(*␣Allow␣infinite␣stuttering␣to␣prevent␣deadlock␣on␣termination.␣*)
Terminating␣==␣pc␣=␣"Done"␣/\␣UNCHANGED␣vars

Next␣==␣l0␣\/␣w␣\/␣l2␣\/␣l3␣\/␣l4␣\/␣l5␣\/␣l6␣\/␣l7␣\/␣l8␣\/␣l9␣\/␣l10
␣␣␣␣␣␣␣␣␣␣␣\/␣Terminating

Spec␣==␣Init␣/\␣[][Next]_vars

Termination␣==␣<>(pc␣=␣"Done")

\*␣END␣TRANSLATION



L␣==␣{"l0","l1"}
D␣==␣MININT..MAXINT

DD(X)␣==␣X=defaultInitValue␣=>␣X␣\in␣D

i␣==
```

```
␣␣␣␣/\␣pc␣\in␣L
␣␣␣␣/\␣DD(y1)␣/\␣DD(y2)␣/\␣DD(y3)␣/\␣DD(z)
␣␣␣␣/\␣typeInt(x1)/\␣typeInt(x2)␣/\␣typeInt(y1)␣/\␣typeInt(y2)␣/\␣typeInt(y3)␣/\␣ty
␣␣␣␣/\␣pc="y0"␣=>␣x1=x10␣/\␣x2=x20
␣␣␣␣/\␣pc="l1"␣=>␣x1=x10␣/\␣x2␣=␣x20␣/\␣␣y2␣\geq␣0␣␣/\␣y3*␣y1^y2␣=␣x1^x2
␣␣␣␣/\␣␣pc␣=␣"w"␣=>␣x1=x10␣/\␣x2␣=␣x20␣/\␣␣y2␣\geq␣0␣␣/\␣y3*␣y1^y2␣=␣x1^x2
␣␣␣␣/\␣pc="l2"␣=>␣␣y2␣#␣0␣/\␣y3*␣y1^y2␣=␣x1^x2




Q1␣==␣pc␣#␣"Done"
Qpc␣==␣pc␣="Done"␣=>␣z=x1^x2

=====================================================================
\*␣Modification␣History
\*␣Last␣modified␣Sun␣Nov␣17␣20:05:05␣CET␣2024␣by␣mery
\*␣Created␣Wed␣Sep␣09␣17:02:47␣CEST␣2015␣by␣mery
```

Exercice 3 *(appex5_3.tla)*
On considère l'algorithme suivant :

START
$\{x_1 \geqq 0 \wedge x_2 > 0\}$
$(y_1, y_2, y_3) \leftarrow (x_1, 0, x_2)$;
while $y_3 \leqq y_1$ **do** $y_3 \leftarrow 2y_3$;
while $y_3 \neq x_2$ **do**
  begin $(y_2, y_3) \leftarrow (2y_2, y_3/2)$;
    if $y_3 \leqq y_1$ **do** $(y_1, y_2) \leftarrow (y_1 - y_3, y_2 + 1)$
  end;
$(z_1, z_2) \leftarrow (y_1, y_2)$
$\{0 \leqq z_1 < x_2 \wedge x_1 = z_2 x_2 + z_1\}$
HALT

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 langage PlusCal.*

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

Listing 3 – appex5-3.tla

```
----------------------------------- MODULE appex5_3 -----------------------------------
EXTENDS TLC, Integers , Naturals
CONSTANTS x1 , x2 , min , max

(*
-wfNext
--algorithm division {
variables y1 , y2 , y3 , z1 , z2 ;
{
l1 : y1:=x1 ; y2:=0; y3:=x2 ;
l2 : while (y3 \leq y1){
     y3:=2*y3 ;
```

```
      };
  l3:while (y3#x2){
      assert x1=y2*x2+y1;
      y2:=2*y2;
      y3:=y3 \div 2;
      l4:if (y3\leq y1) {
      y1:=y1-y3;
      y2:=y2+1;
      };
      assert x1=y2*x2+y1;
      };
    l5: z1:=y1;
      z2:=y2;
      assert x1=y2*x2+y1;
      print <<x1,x2,z1,z2>>;
      }
  }

*)
\* BEGIN TRANSLATION
CONSTANT defaultInitValue
VARIABLES y1, y2, y3, z1, z2, pc

vars == << y1, y2, y3, z1, z2, pc >>

Init == (* Global variables *)
        /\ y1 = defaultInitValue
        /\ y2 = defaultInitValue
        /\ y3 = defaultInitValue
        /\ z1 = defaultInitValue
        /\ z2 = defaultInitValue
        /\ pc = "l1"

l1 == /\ pc = "l1"
      /\ y1'␣=␣x1
␣␣␣␣␣␣/\␣y2' = 0
      /\ y3'␣=␣x2
␣␣␣␣␣␣/\␣pc' = "l2"
      /\ UNCHANGED << z1, z2 >>

l2 == /\ pc = "l2"
      /\ IF y3 \leq y1
            THEN /\ y3'␣=␣2*y3
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣/\␣pc' = "l2"
            ELSE /\ pc'␣=␣"l3"
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣/\␣y3' = y3
      /\ UNCHANGED << y1, y2, z1, z2 >>

l3 == /\ pc = "l3"
      /\ IF y3#x2
            THEN /\ Assert(x1=y2*x2+y1,
                           "Failure␣of␣assertion␣at␣line␣15,␣column␣5.")
                 /\ y2'␣=␣2*y2
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣/\␣y3' = (y3 \div 2)
                 /\ pc'␣=␣"l4"
␣␣␣␣␣␣␣␣␣␣␣ELSE␣/\␣pc' = "l5"
```

```
                 /\ UNCHANGED << y2, y3 >>
      /\ UNCHANGED << y1, z1, z2 >>

l4 == /\ pc = "l4"
      /\ IF y3\leq y1
            THEN /\ y1'␣=␣y1-y3
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣/\␣y2' = y2+1
            ELSE /\ TRUE
                 /\ UNCHANGED << y1, y2 >>
      /\ Assert(x1=y2'*x2+y1', "Failure␣of␣assertion␣at␣line␣22,␣column␣5.")
      /\ pc'␣=␣"l3"
␣␣␣␣␣␣/\␣UNCHANGED␣<<␣y3,␣z1,␣z2␣>>

l5␣==␣/\␣pc␣=␣"l5"
␣␣␣␣␣␣/\␣z1' = y1
      /\ z2'␣=␣y2
␣␣␣␣␣␣/\␣Assert(x1=y2*x2+y1,␣"Failure␣of␣assertion␣at␣line␣26,␣column␣5.")
␣␣␣␣␣␣/\␣PrintT(<<x1,x2,z1',z2'>>)
␣␣␣␣␣␣/\␣pc' = "Done"
      /\ UNCHANGED << y1, y2, y3 >>

(* Allow infinite stuttering to prevent deadlock on termination. *)
Terminating == pc = "Done" /\ UNCHANGED vars

Next == l1 \/ l2 \/ l3 \/ l4 \/ l5
           \/ Terminating

Spec == Init /\ [][Next]_vars

Termination == <>(pc = "Done")

\* END TRANSLATION


Iloop(u,v)  == x1=v*x2+u
Qpc == pc="Done" => x1=z2*x2+z1 /\ 0 \leq z1 /\ z1 \leq x2
COND(U) ==    min \leq U /\ U \leq max

Qof ==  COND(y1) /\ COND(y2) /\ COND (y3) /\ COND(z1) /\ COND (z2)

i == Iloop(y1,y2)
=============================================================================
\* Modification History
\* Last modified Tue Nov 24 21:30:57 CET 2020 by mery
\* Created Wed Nov 18 16:33:27 CET 2015 by mery
```

Exercice 4 *annotation*
Montrer que chaque annotation est correcte ou incorrecte selon les conditions de vérifications énoncées comme suit
$\forall x, y, , x', y'.P_\ell(x, y) \ \wedge \ cond_{\ell,\ell'}(x, y) \ \wedge \ (x', y') = f_{\ell,\ell'}(x, y) \Rightarrow P_{\ell'}(x', y')$
Pour cela, on utlisera une machine et un ciontexte Event-B.

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

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

— *On suppose que p est un nombre premier :*
$\ell_1 : x = 2^p \ \wedge\ y = 2^{p+1} \ \wedge\ x{\cdot}y = 2^{2\cdot p+1}$
$x := y{+}x{+}2^x$
$\ell_2 : x = 5{\cdot}2^p \ \wedge\ y = 2^{p+1}$

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

Exercice 5 *(Vérification de l'annotation de l'algorithme du calcul du maximum d'une liste) Vérifier l'annotation de l'algorithme de calcul du maximum d'une liste 5. On se donne l'annotation et on demande de construire une machine permettant de vérifier cette annotation.*

Vérification **precondition** : $\begin{pmatrix} n \in \mathbb{N} \wedge \\ n \neq 0 \wedge \\ f \in 0\,..\,n{-}1 \rightarrow \mathbb{N} \end{pmatrix}$

postcondition : $\begin{pmatrix} m \in \mathbb{N} \wedge \\ m \in ran(f) \wedge \\ (\forall j \cdot j \in 0\,..\,n{-}1 \Rightarrow f(j) \leq m) \end{pmatrix}$

local variables : $i \in \mathbb{Z}$

$m := f(0);$
$i := 1;$
while $i < n$ **do**
  if *f(i)>m* **then**
    $m := f(i);$
  ;
  $i{+}{+};$
;

Algorithme 4: Algorithme du maximum d'une liste non annotée

CONTEXT CONTEXT 0

sets C

CONSTANTS $f\ n\ l0\ l1\ l2\ l3\ l4\ l5\ l6\ l7\ l8\ l9$

AXIOMS
$@axm1\ n \ \in\ \mathbb{N}1$
$@axm2\ f \ \in\ 0..n{-}1 \ \rightarrow\ \mathbb{N}$
$@axm3\ partition(C,\ \{l0\},\ \{l1\},\ \{l2\},\ \{l3\},\ \{l4\},\ \{l5\},\ \{l6\},\ \{l7\},\ \{l8\},\ \{l9\})$
$@axm4\ \forall P.\ P \ \subseteq\ \mathbb{N} \ \wedge\ finite(P) \ \Rightarrow\ (\exists\, am.\ am \ \in\ P \ \wedge\ (\forall\, k.\ k \ \in\ P \ \Rightarrow\ k \ \leq\ am))$
end

/* algorithme de calcul du maximum avec une boucle while de l'exercice **??** */

precondition : $\left(\begin{array}{l} n \in \mathbb{N} \wedge \\ n \neq 0 \wedge \\ f \in 0 \mathinner{..} n{-}1 \rightarrow \mathbb{N} \end{array}\right)$

postcondition : $\left(\begin{array}{l} m \in \mathbb{N} \wedge \\ m \in ran(f) \wedge \\ (\forall j \cdot j \in 0 \mathinner{..} n{-}1 \Rightarrow f(j) \leq m) \end{array}\right)$

local variables : $i \in \mathbb{Z}$

$\ell_0 : \{ \left(\begin{array}{l} n \in \mathbb{N} \wedge \\ n \neq 0 \wedge \\ f \in 0 \mathinner{..} n{-}1 \rightarrow \mathbb{N} \end{array}\right) \wedge i \in \mathbb{Z} \wedge i \in \mathbb{Z} \wedge ...\}$

$m := f(0);$

$\ell_1 : \{ \left(\begin{array}{l} n \in \mathbb{N} \wedge \\ n \neq 0 \wedge \\ f \in 0 \mathinner{..} n{-}1 \rightarrow \mathbb{N} \end{array}\right) \wedge i \in \mathbb{Z} \wedge m = f(0)\}$

$i := 1;$

$\ell_2 : \{ \left(\begin{array}{l} n \in \mathbb{N} \wedge \\ n \neq 0 \wedge \\ f \in 0 \mathinner{..} n{-}1 \rightarrow \mathbb{N} \end{array}\right) \wedge i = 1 \wedge \left(\begin{array}{l} m \in \mathbb{N} \wedge \\ m \in ran(f[0..i{-}1]) \wedge \\ (\forall j \cdot j \in 0 \mathinner{..} i{-}1 \Rightarrow f(j) \leq m) \end{array}\right)\}$

while $i < n$ **do**

$\ell_3 : \{ \left(\begin{array}{l} n \in \mathbb{N} \wedge \\ n \neq 0 \wedge \\ f \in 0 \mathinner{..} n{-}1 \rightarrow \mathbb{N} \end{array}\right) \wedge i \in 1..n{-}1 \wedge \left(\begin{array}{l} m \in \mathbb{N} \wedge \\ m \in ran(f[0..i{-}1]) \wedge \\ (\forall j \cdot j \in 0 \mathinner{..} i{-}1 \Rightarrow f(j) \leq m) \end{array}\right)\}$

if *f(i)>m* **then**

$\ell_4 : \{ \left(\begin{array}{l} n \in \mathbb{N} \wedge \\ n \neq 0 \wedge \\ f \in 0 \mathinner{..} n{-}1 \rightarrow \mathbb{N} \end{array}\right) \wedge i \in 1..n{-}1 \wedge \left(\begin{array}{l} m \in \mathbb{N} \wedge \\ m \in ran(f[0..i{-}1]) \wedge \\ (\forall j \cdot j \in 0 \mathinner{..} i{-}1 \Rightarrow f(j) \leq m) \end{array}\right) \wedge$
$f(i) > m\}$

$m := f(i);$

$\ell_5 : \{ \left(\begin{array}{l} n \in \mathbb{N} \wedge \\ n \neq 0 \wedge \\ f \in 0 \mathinner{..} n{-}1 \rightarrow \mathbb{N} \end{array}\right) \wedge i \in 1..n{-}1 \wedge \left(\begin{array}{l} m \in \mathbb{N} \wedge \\ m \in ran(f[0..i]) \wedge \\ (\forall j \cdot j \in 0 \mathinner{..} i \Rightarrow f(j) \leq m) \end{array}\right)\}$

;

$\ell_6 : \{ \left(\begin{array}{l} n \in \mathbb{N} \wedge \\ n \neq 0 \wedge \\ f \in 0 \mathinner{..} n{-}1 \rightarrow \mathbb{N} \end{array}\right) \wedge i \in \mathbb{Z} \wedge \wedge i \in 1..n{-}1 \wedge \left(\begin{array}{l} m \in \mathbb{N} \wedge \\ m \in ran(f[0..i]) \wedge \\ (\forall j \cdot j \in 0 \mathinner{..} i \Rightarrow f(j) \leq m) \end{array}\right)\}$

$i{+}{+};$

$\ell_7 : \{ \left(\begin{array}{l} n \in \mathbb{N} \wedge \\ n \neq 0 \wedge \\ f \in 0 \mathinner{..} n{-}1 \rightarrow \mathbb{N} \end{array}\right) \wedge i \in 1..n{-}1 \wedge \left(\begin{array}{l} m \in \mathbb{N} \wedge \\ m \in ran(f[0..i{-}1]) \wedge \\ (\forall j \cdot j \in 0 \mathinner{..} i{-}1 \Rightarrow f(j) \leq m) \end{array}\right)\}$

;

$\ell_8 : \{ \left(\begin{array}{l} n \in \mathbb{N} \wedge \\ n \neq 0 \wedge \\ f \in 0 \mathinner{..} n{-}1 \rightarrow \mathbb{N} \end{array}\right) \wedge i = n \wedge \left(\begin{array}{l} m \in \mathbb{N} \wedge \\ m \in ran(f) \wedge \\ (\forall j \cdot j \in 0 \mathinner{..} n{-}1 \Rightarrow f(j) \leq m) \end{array}\right)\}$

Algorithme 5: Algorithme du maximum d'une liste annoté

MACHINE $algorithm$ SEES CONTEXT 0

VARIABLES $l\ m\ i$

INVARIANTS
$@inv1\ l \in C$
$@inv2\ m \in \mathbb{N}$
$@inv3\ i \in \mathbb{N}$
$@inv4\ i \in 0..n$
$@inv5\ l = l0 \Rightarrow m \in \mathbb{N} \land i \in \mathbb{N}$
$@inv6\ l = l1 \Rightarrow m = f(0)$
$@inv7\ l = l2 \Rightarrow i = 1 \land m = f(0) \land i \leq n \land 0..i{-}1 \subseteq dom(f) \land (\forall\ j\ .\ j \in 0..i{-}1 \Rightarrow f(j)$
$@inv8\ l = l3 \Rightarrow i < n \land 0..i \subseteq dom(f) \land (\forall\ j\ .\ j \in 0..i{-}1 \Rightarrow f(j) \leq m) \land m \in ran(f)$
$@inv9\ l = l4 \Rightarrow i < n \land 0..i \subseteq dom(f) \land (\forall\ j\ .\ j \in 0..i{-}1 \Rightarrow f(j) \leq m) \land f(i) > m \land$
$@inv10\ l = l5 \Rightarrow i < n \land 0..i \subseteq dom(f) \land (\forall\ j\ .\ j \in 0..i{-}1 \Rightarrow f(j) \leq m) \land (\forall\ j\ .\ j \in$
$@inv11\ l = l6 \Rightarrow i < n \land 0..i \subseteq dom(f) \land (\forall\ j\ .\ j \in 0..i \Rightarrow f(j) \leq m) \land m \in ran(f)$
$@inv12\ l = l7 \Rightarrow i \leq n \land 0..i{-}1 \subseteq dom(f) \land (\forall\ j\ .\ j \in 0..i{-}1 \Rightarrow f(j) \leq m) \land m \in ran($
$@inv13\ l = l8 \Rightarrow i = n \land dom(f) \subseteq 0..i{-}1 \land (\forall\ j\ .\ j \in 0..i{-}1 \Rightarrow f(j) \leq m) \land m \in ran($
$theorem\ @post\ l = l8 \Rightarrow (\forall\ j.\ j \in 0..n{-}1 \Rightarrow f(j) \leq m) \land m \in ran(f)$
$theorem\ @pre\ f \in 0..n{-}1 \rightarrow \mathbb{N} \land i \in 0..n \land m \in \mathbb{N} \Rightarrow m \in \mathbb{N} \land i \in \mathbb{N}$

EVENTS
EVENT $INITIALISATION$
then
$@act5\ l := l0$
$@act6\ m :\in \mathbb{N}$
$@act7\ i :\in 0..n$
end

EVENT $al0l1$
where
$@grd1\ l = l0$
then
$@act4\ l := l1$
$@act5\ m := f(0)$
end

EVENT $al1l2$
where
$@grd1\ l = l1$
then
$@act1\ l := l2$
$@act2\ i := 1$
end

EVENT $al2l3$
where
$@grd1\ l = l2$
$@grd2\ i < n$
then
$@act1\ l := l3$
end

EVENT $al2l8$
where
$@grd1\ l = l3$
$@grd2\ i \geq n$
then
$@act1\ l := l8$
end

EVENT $am3l4$

Exercice 6 *(Annotation du calcul de la racine carrée entière appex5_6.tla)*
L'algorithme annoté **??** *calcule la racine carrée entière d'un nombre entrier. Vérifier les annotations par un mdodèle Event-B.*

variables X, Y_1, Y_2, Y_3, Z
requires
- $x0 \in \mathbb{N}$
- $y10 \in Int$
- $y20 \in Int$
- $y30 \in Int$
- $z0 \in Int$

ensures
- $zf \cdot zf \leq x < (zf+1) \cdot (zf+1)$
- $xf = x0$
- $zf = y1f$
- $y2f = y1f+1$
- $y3f = 2 \cdot y1f+1$

begin
$\ell_0 : \{x \in \mathbb{N} \wedge z \in \mathbb{Z} \wedge y1 \in \mathbb{Z} \wedge y2 \in \mathbb{Z} \wedge y3 \in \mathbb{Z}\}$
$(Y_1, Y_2, Y_3) := (0, 1, 1);$
$\ell_1 : \{y2 = (y1+1) \cdot (y1+1) \wedge y3 = 2 \cdot y1+1 \wedge y1 \cdot y1 \leq x\}$
While $(Y_2 \leq X)$
$\ell_2 : \{y2 = (y1+1) \cdot (y1+1) \wedge y3 = 2 \cdot y1+1 \wedge y2 \leq x\}$
$(Y_1, Y_2, Y_3) := (Y_1+1, Y_2+Y_3+2, Y_3+2);$
$\ell_3 : \{y2 = (y1+1) \cdot (y1+1) \wedge y3 = 2 \cdot y1+1 \wedge y1 \cdot y1 \leq x\}$
od;
$\ell_4 : \{y2 = (y1+1) \cdot (y1+1) \wedge y3 = 2 \cdot y1+1 \wedge y1 \cdot y1 \leq x \wedge x < y2\}$
$Z := Y_1;$
$\ell_5 : \{y2 = (y1+1) \cdot (y1+1) \wedge y3 = 2 \cdot y1+1 \wedge y1 \cdot y1 \leq x \wedge x < y2 \wedge z = y1 \wedge z \cdot z \leq x \wedge x < (z+1) \cdot (z+1)\}$
end

Exercice 7 *Soient les contrats suivants. Pour chaque contrat, Ã©valuer sa validitÃ© avec le calcul des wps.*

Question 7.1

requires ...
ensures $z_f = 100 \wedge x_f+y_f = 12 \wedge x_f+x_0 = 4;$
variables x, y, z

begin
$/ \cdot @assert; \cdot /$
$x = x+1;$
$/ \cdot @assert; \cdot /$
$y = x+y+2;$
$/ \cdot @assert; \cdot /$
$z = x+y;$
$/ \cdot @assert; \cdot /$
end

La version ACSL est la suivante

Listing 4 – td51.c

```
struct data {
    unsigned x;
    unsigned y;
    unsigned z;
};

/*@
```

```
  @ ensures \result.z == 100 && \result.x+\result.y == 12 && \result.x + x0==4;
  */


struct data  exemple(int x0,int y0,int z0)
{
  int x=x0;
  int y=y0;
  int z=z0;
//@  assert  x == x0;
  x = x + 1;
  y=x+y+2;                                          \
  z = x +y;
  struct data r;
  r.x = x;r.y=y;r.z=z;
  return r;
}
```

Question 7.2

requires $x_0, y_0 \in \mathbb{N}$
ensures $z_f = max(x_0, y_0)$
variables x, y, z

begin
$/ \cdot @assert; \cdot /$
IF $x < y$ THEN
$/ \cdot @assert; \cdot /$
$z := y;$
$/ \cdot @assert; \cdot /$
ELSE
$/ \cdot @assert; \cdot /$
$z := x;$
$/ \cdot @assert; \cdot /$
FI;
$/ \cdot @assert; \cdot /$
end

La version ACSL est la suivante

Listing 5 – td52.c

```
/*@ requires x0 >= 0 && y0 >= 0;
  @ ensures (\result == x0 || \result == y0)  && \result >= x0 && \result >= y0;
  */

 exemple(int x0,int y0)
{
  int x=x0;
  int y=y0;
  int z;
//@  assert  x == x0  && y == y0;
  if ( x < y )
    {
    z = y;
    }
  else
    {
    z=x;
    };
```

```
    return z;
}
```

Exercice 8 *td58.c*
On suppose que val est une valeur entière. Vérifier l'annotation suivante :

Listing 6 – td51.c

```
#define v 3
/*@ requires val == v;
 */

int exemple(int val)
{
  int c = val ;
  //@ assert c == 2;
  int x;
  //@ assert c == 2;
  x = 3 * c ;
  //@ assert x == 6;
  return(0);
}
```

◇– **Solution de l'exercice 8**

Listing 7 – td51.c

```
#define v 3
/*@ requires val == v;
 */

int exemple(int val)
{
  int c = val ;
  //@ assert c == 2;
  int x;
  //@ assert c == 2;
  x = 3 * c ;
  //@ assert x == 6;
  return(0);
}
```

Fin 8

Exercice 9 *td59.c*
Vérifier l'annotation suivante :

Listing 8 – td59.c

```
int exemple()
{
  int a = 42; int b = 37;
  int c = a+b;
l1: b == 37 ;
  a -= c;
  b += a;
l2: b == 0 && c == 79;
  return(0);
}
```

Exercice 10 *Vérifier l'annotation suivante :*

Listing 9 – td510.c

```
int main()
{
  int z;
  int a = 4;
  //@ assert a == 4 ;
  int b = 3;
  //@ assert  b == 3 && a == 4;
  int c = a+b;
  //@ assert b == 3 && c == 7 && a == 4 ;
  a += c;
  b += a;
  //@ assert a == 11 && b == 14 && c == 7 ;
  //@ assert a +b  == 25 ;
  z = a*b;
  //@ assert  a == 11 && b == 14 && c == 7 && z == 154;
  return(0);
}
```

◇– **Solution de l'exercice 10**

Listing 10 – td510.c

```
int main()
{
  int z;
  int a = 4;
  //@ assert a == 4 ;
  int b = 3;
  //@ assert  b == 3 && a == 4;
  int c = a+b;
  //@ assert b == 3 && c == 7 && a == 4 ;
  a += c;
  b += a;
  //@ assert a == 11 && b == 14 && c == 7 ;
  //@ assert a +b  == 25 ;
  z = a*b;
  //@ assert  a == 11 && b == 14 && c == 7 && z == 154;
  return(0);
}
```

Fin 10

Exercice 11 *Vérifier l'annotation suivante :*

```
int main()
{
  int a = 4;
  int b = 3;
  int c = a+b;
  a += c;
  b += a;
l:  a == 11 && b == 14 && c == 7 ;
  return(0);
}
```

◇— **Solution de l'exercice 11**

Listing 11 – annotation 5 (wp6.c)

```
int main()
{
  int a = 4;
  int b = 3;
  int c = a+b; // i:1
  a += c; // i:2
  b += a; // i:3
//@assert a == 11 && b == 14 && c == 7 ;
  return(0);
}
```

Fin 11