MVSI – Modelling and Verification of Computer Systems

Modelling algorithms in PlusCal (I)

Tutorial – with solutions

Dominique Méry – 20th November 2026

✎ Exercise 1 appex4...

We use the translation of a PlusCal algorithm into a TLA module to check algorithms. For each question, write a TLA module containing an expression of the algorithm, translate it into a module, and then analyse the resulting module with respect to partial correctness and the absence of run-time errors.

Question 1.1 (appex4_1_1.tla)

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

✔ Solution of exercise 1

Listing 1: TLA$^+$ module appex4_1_1.tla

```
--------- MODULE appex4_1_1 --------
EXTENDS Integers,TLC
-----------------------------------------------------------------
CONSTANTS x0,y0,z0
-----------------------------------------------------------------
pre(a,b,c)  == a=10 /\ c=2*a /\ b=c+a
-----------------------------------------------------------------
(* Verifying the existence of a triple that satisfies pre(x0;y0,z0) *)
ASSUME pre(x0,y0,z0)
-----------------------------------------------------------------
(*
--algorithm  test  {
variables x=x0,z=z0,y=y0,t;
{
l1: assert x=10 /\ z=2*x /\ y = z + x;
y:=z+x;
l2: assert x = 10 /\ y = x + 2*10;
}
}
*)
\* BEGIN TRANSLATION (chksum(pcal) = "6b4ae46f" /\ chksum(tla) = "d33b1a21")
CONSTANT defaultInitValue
VARIABLES x, z, y, t, pc

vars == << x, z, y, t, pc >>

Init == (* Global variables *)
        /\ x = x0
        /\ z = z0
        /\ y = y0
        /\ t = defaultInitValue
        /\ pc = "l1"

l1 == /\ pc = "l1"
      /\ Assert(x=10 /\ z=2*x /\ y = z + x,
                "Failure␣of␣assertion␣at␣line␣15,␣column␣5.")
      /\ y' = z+x
      /\ pc' = "l2"
      /\ UNCHANGED << x, z, t >>

l2 == /\ pc = "l2"
      /\ Assert(x = 10 /\ y = x + 2*10,
                "Failure␣of␣assertion␣at␣line␣17,␣column␣5.")
      /\ pc' = "Done"
      /\ UNCHANGED << x, z, y, t >>

```

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

Next == l1 \/ l2
           \/ Terminating

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

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

\* END TRANSLATION

\* ASSUME pre
-----------------------------------------------------------------
(* preminaries for checking  on  pssible values *)
MAX == 32767  (* 16 bits *)
D == -32768..32767
DD(X) ==  (X # defaultInitValue) => (X \in D)
-----------------------------------------------------------------
Inv ==
    /\ pc \in {"l1","l2","Done"}
    /\ x \in Int /\ y \in Int /\ z \in Int
    /\ pc="l1" =>  x=10 /\  z=2*x /\ y=z+x
    /\ pc="l2" =>   x=10 /\ y=x+2*10
    /\ pc="Done" =>   x=10 /\ y=x+2*10
Safety_Partial_Correctness == pc="Done" =>   x=10 /\ y=x+2*10
Safety_rte ==  DD(x)  /\ DD(y) /\  DD(z) /\ DD(t)
check == Inv /\ Safety_Partial_Correctness /\ Safety_rte
prop == [] (x=x0)
=================================================
```

Listing 2: TLC configuration `appex4_1_1.cfg`

```
SPECIFICATION Spec
CONSTANT defaultInitValue = defaultInitValue
\* Add statements after this line.
\* SPECIFICATION
\* Uncomment the previous line and provide the specification name if it's declared
\* in the specification file. Comment INIT / NEXT parameters if you use SPECIFICATION.

CONSTANTS
    greeting = "Hello"
    x0=10
    y0 =30
     z0 = 20
     defaultInitValue = 50000

\* PROPERTY
\* Uncomment the previous line and add property names
\* prop
INVARIANT
\* Uncomment the previous line and add invariant names
check
Inv
```

Question 1.2 We assume that p is a prime number:

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

✔ Solution of exercise 1

Listing 3: TLA$^+$ module `appex4_1_2.tla`

```
--------- MODULE appex4_1_2 --------
EXTENDS Integers, Naturals,TLC
CONSTANTS x0,y0,p,maxi,defaultInitValue
```

```
--------------------------------------------------------------
diviseurs(n) == { m \in 1..maxi:  n % m = 0 }
premier(n) ==  diviseurs(n) = {1,n} /\ n # 1
pre == x0=2^p /\ y0=2^(p+1) /\ x0*y0=2^(2*p+1)
--------------------------------------------------------------
ASSUME premier(p) /\ pre
(*

--algorithm  test  {
variables x=x0,y=y0;
{
l1:assert x=2^p /\ y=(2^p)*2 /\  x*y=2^(2*p+1);
x:=y+x+(2^x);
l2:assert x=5*(2^p) /\ y=2^(p+1) ;
}
}
*)
\* BEGIN TRANSLATION (chksum(pcal) = "31a0954b" /\ chksum(tla) = "3211f152")
VARIABLES x, y, pc

vars == << x, y, pc >>

Init == (* Global variables *)
        /\ x = x0
        /\ y = y0
        /\ pc = "l1"

l1 == /\ pc = "l1"
      /\ Assert(x=2^p /\ y=(2^p)*2 /\  x*y=2^(2*p+1),
                "Failure␣of␣assertion␣at␣line␣15,␣column␣4.")
      /\ x' = y+x+(2^x)
      /\ pc' = "l2"
      /\ y' = y

l2 == /\ pc = "l2"
      /\ Assert(x=5*(2^p) /\ y=2^(p+1),
                "Failure␣of␣assertion␣at␣line␣17,␣column␣4.")
      /\ pc' = "Done"
      /\ UNCHANGED << x, y >>

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

Next == l1 \/ l2
           \/ Terminating

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

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

\* END TRANSLATION

--------------------------------------------------------------
MAX == 32768  (* 16 bits *)
D == 0..32768
DD(X) == ( X \in D)
DDP(X) ==  (X # defaultInitValue) => (X \in D)
Safety_absence ==  DD(x)  /\ DD(y)
Inv ==
    /\ pc \in {"l1","l2","Done"}
    /\ x \in Int /\ y \in Int
    /\ pc="l1" => x=2^p /\ y=2^p*2 /\  x*y=2^(2*p+1)
    /\ pc="l2" =>    x=5*2^p /\ y=2^(p+1)

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

Listing 4: TLC configuration appex4_1_2.cfg

```
SPECIFICATION Spec
\* Add statements after this line.
\* SPECIFICATION
\* Uncomment the previous line and provide the specification name if it's declared
\* in the specification file. Comment INIT / NEXT parameters if you use SPECIFICATION.
```

```
CONSTANTS
    greeting = "Hello"
    p = 2
    x0 = 4
    y0 = 8
    maxi  = 89
    defaultInitValue = 0


\* PROPERTY
\* Uncomment the previous line and add property names

\* INVARIANT
\* Uncomment the previous line and add invariant names
```

Question 1.3

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

✔ Solution of exercise 1

Listing 5: TLA$^+$ module appex4_1_3.tla

```
--------- MODULE appex4_1_3 --------
EXTENDS Integers,TLC
-----------------------------------------------------------------
CONSTANTS x0,y0
-----------------------------------------------------------------
pre == x0=1 /\ y0=12

-----------------------------------------------------------------
(*

--algorithm  test  {
variables x=x0,y=y0;
{
l1:assert x=1 /\ y=12;
x:=2*y;
l2:assert x=24 /\ y=12;
}
}
*)
\* BEGIN TRANSLATION (chksum(pcal) = "fdb997c5" /\ chksum(tla) = "97306b1b")
VARIABLES x, y, pc

vars == << x, y, pc >>

Init == (* Global variables *)
        /\ x = x0
        /\ y = y0
        /\ pc = "l1"

l1 == /\ pc = "l1"
      /\ Assert(x=1 /\ y=12, "Failure␣of␣assertion␣at␣line␣14,␣column␣4.")
      /\ x' = 2*y
      /\ pc' = "l2"
      /\ y' = y

l2 == /\ pc = "l2"
      /\ Assert(x=24 /\ y=12, "Failure␣of␣assertion␣at␣line␣16,␣column␣4.")
      /\ pc' = "Done"
      /\ UNCHANGED << x, y >>

(* Allow infinite stuttering to prevent deadlock on termination. *)
```

```
Terminating == pc = "Done" /\ UNCHANGED vars

Next == l1 \/ l2
           \/ Terminating

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

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

\* END TRANSLATION

-----------------------------------------------------------------
MAX == 32768  (* 16 bits *)
D == 0..32768
DD(X) == ( X \in D)
DDP(X) ==  (X # defaultInitValue) => (X \in D)
-----------------------------------------------------------------
Safety_absence ==  DD(x)  /\ DD(y)
Inv ==
    /\ pc \in {"l1","l2","Done"}
    /\ x \in Int /\ y \in Int
    /\ pc="l1" =>  x=1 /\ y=12
    /\ pc="l2" =>  x=1 /\ y=24

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

Listing 6: TLC configuration appex4_1_3.cfg

```
\* SPECIFICATION
\* Uncomment the previous line and provide the specification name if it's declared
\* in the specification file. Comment INIT / NEXT parameters if you use SPECIFICATION.

CONSTANTS
    greeting = "Hello"

INIT Init
NEXT Next

\* PROPERTY
\* Uncomment the previous line and add property names

\* INVARIANT
\* Uncomment the previous line and add invariant names
```

Question 1.4

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

✔ Solution of exercise 1

Listing 7: TLA$^+$ module appex4_1_4.tla

```
--------- MODULE appex4_1_4 --------
EXTENDS Integers,TLC
-----------------------------------------------------------------

(*
--algorithm  test  {
variables x=11,y=13,z;
{
l1:assert x=11 /\ y=13;
z:=x;x:=y;y:=z;
l2:assert x=26 \div 2 /\ y=33 \div 3;

l3:print <<x,y>>;
```

```
}
}
*)
\* BEGIN TRANSLATION
CONSTANT defaultInitValue
VARIABLES x, y, z, pc

vars == << x, y, z, pc >>

Init == (* Global variables *)
        /\ x = 11
        /\ y = 13
        /\ z = defaultInitValue
        /\ pc = "l1"

l1 == /\ pc = "l1"
      /\ Assert(x=11 /\ y=13, "Failure␣of␣assertion␣at␣line␣10,␣column␣4.")
      /\ z' = x
      /\ x' = y
      /\ y' = z'
      /\ pc' = "l2"

l2 == /\ pc = "l2"
      /\ Assert(x=26 \div 2 /\ y=33 \div 3,
                "Failure␣of␣assertion␣at␣line␣12,␣␣column␣4.")
      /\ pc' = "l3"
      /\ UNCHANGED << x, y, z >>

l3 == /\ pc = "l3"
      /\ PrintT(<<x,y>>)
      /\ pc' = "Done"
      /\ UNCHANGED << x, y, z >>

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

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

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

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

\* END TRANSLATION

-----------------------------------------------------------------
MAX == 32768  (* 16 bits *)
D == 0..32768
DD(X) == ( X \in D)
DDP(X) ==  (X # defaultInitValue) => (X \in D)
---------------------------------------
Safety_absence ==  DD(x)  /\ DD(y)
Inv ==
    /\ pc \in {"l1","l2","Done"}
    /\ x \in Int /\ y \in Int
    /\ pc="l1" =>  x=11 /\ y=13
    /\ pc="l2" =>  x=26 \div 2 /\ y=33 \div 3

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

Listing 8: TLC configuration appex4_1_4.cfg

```
\* SPECIFICATION
\* Uncomment the previous line and provide the specification name if it's declared
\* in the specification file. Comment INIT / NEXT parameters if you use SPECIFICATION.

CONSTANTS
    greeting = "Hello"

INIT Init
NEXT Next

\* PROPERTY
```

```
\* Uncomment the previous line and add property names

\* INVARIANT
\* Uncomment the previous line and add invariant names
```

✎ Exercise 2

Annotate each algorithm and translate it with the PlusCal to TLA$^+$ translator.

Question 2.1 `(pluscal_loop)`

Listing 9: Fragment

```
         variables x=x0;
{
 while (0<x)
  {
     x := x - 1;
   };
}
}
```

Question 2.2 `(pluscal_max)`

Listing 10: Fragment

```
  variables x=a,y = b,z;
{
 if ( x < y ) {
  z:=y;
  }
else
{
 z:= x;
 };
 }
}
```

✎ Exercise 3 `appex4_3.tla`

We consider the following algorithm:

START
$\{x_1 \geq 0 \wedge x_2 > 0\}$
$(y_1, y_2, y_3) \leftarrow (x_1, 0, x_2)$;
while $y_3 \leq y_1$ **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 $y_3 \leq y_1$ **then** $(y_1, y_2) \leftarrow (y_1 - y_3, y_2 + 1)$
  end;
$(z_1, z_2) \leftarrow (y_1, y_2)$
$\{0 \leq z_1 < x_2 \wedge x_1 = z_2 x_2 + z_1\}$
HALT

Question 3.1 Show that this algorithm is partially correct with respect to its precondition and its postcondition, which you will state. To do so, translate this algorithm into a module from the PlusCal language.

Question 3.2 Show that it is free of run-time errors.

✔ Solution of exercise 3

Listing 11: TLA$^+$ module appex4_3.tla

```
---------------------- MODULE   appex4_3 --------------------------------
EXTENDS Naturals, Integers, TLC

CONSTANT MAXINT,u,v

(*
--algorithm gcdscm {
  variables x1 =u; (* 1st integer *)
            x2 = v;  (* 2nd integer *)
            y1;
            y2;
            y3;
            y4;
            z1,z2;


{
     print <<x1, x2>>;

    y1:=x1;
    y2:=x2;
    y3:=x2;
    y4:=0;
    while (y1 /= y2) {

      while  ( y1 > y2) {
        y1:=y1-y2;
        y4:=y4+y3;
      };
      while  ( y2 > y1) {
        y2:=y2-y1;
        y3:=y4+y3;
      };

    };
    z1 := y1;
    z2 := y3+y4;
    print <<x1, x2,z1,z2>>
}

}
*)
\* BEGIN TRANSLATION
CONSTANT defaultInitValue
VARIABLES x1, x2, y1, y2, y3, y4, z1, z2, pc

vars == << x1, x2, y1, y2, y3, y4, z1, z2, pc >>

Init == (* Global variables *)
        /\ x1 = u
        /\ x2 = v
        /\ y1 = defaultInitValue
        /\ y2 = defaultInitValue
        /\ y3 = defaultInitValue
        /\ y4 = defaultInitValue
        /\ z1 = defaultInitValue
        /\ z2 = defaultInitValue
        /\ pc = "Lbl_1"

Lbl_1 == /\ pc = "Lbl_1"
         /\ PrintT(<<x1, x2>>)
         /\ y1' = x1
         /\ y2' = x2
         /\ y3' = x2
         /\ y4' = 0
         /\ pc' = "Lbl_2"
         /\ UNCHANGED << x1, x2, z1, z2 >>

Lbl_2 == /\ pc = "Lbl_2"
```

```
          /\ IF y1 /= y2
                THEN /\ pc' = "Lbl_3"
                     /\ UNCHANGED << z1, z2 >>
                ELSE /\ z1' = y1
                     /\ z2' = y3+y4
                     /\ PrintT(<<x1, x2,z1',z2'>>)
                     /\ pc' = "Done"
          /\ UNCHANGED << x1, x2, y1, y2, y3, y4 >>

Lbl_3 == /\ pc = "Lbl_3"
          /\ IF y1 > y2
                THEN /\ y1' = y1-y2
                     /\ y4' = y4+y3
                     /\ pc' = "Lbl_3"
                ELSE /\ pc' = "Lbl_4"
                     /\ UNCHANGED << y1, y4 >>
          /\ UNCHANGED << x1, x2, y2, y3, z1, z2 >>

Lbl_4 == /\ pc = "Lbl_4"
          /\ IF y2 > y1
                THEN /\ y2' = y2-y1
                     /\ y3' = y4+y3
                     /\ pc' = "Lbl_4"
                ELSE /\ pc' = "Lbl_2"
                     /\ UNCHANGED << y2, y3 >>
          /\ UNCHANGED << x1, x2, y1, y4, z1, z2 >>

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

Next == Lbl_1 \/ Lbl_2 \/ Lbl_3 \/ Lbl_4
           \/ Terminating

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

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

\* END TRANSLATION

Q1 == pc # "Done"
(* Qpc == pc ="Done" => z1=gcd(x1,x2) /\ z2=scm(x1,x2) *)
D == 0..MAXINT\cup {defaultInitValue}
Qpc1 == pc = "Done" => ( x1 % z1  = 0 ) /\ (x2 % z1 = 0)
Qpc2 == pc = "Done" => ( z2 % x1  = 0 ) /\ (z2 % x2 = 0)
Qproperty1 == pc = "Done" => x1*x2=z1*z2
Qef == x1 \in D
=============================================================================
\* Modification History
\* Last modified Thu Nov 19 20:42:02 CET 2020 by mery
\* Created Wed Sep 09 17:02:47 CEST 2015 by mery
```

Listing 12: TLC configuration appex4_3.cfg

```
\* SPECIFICATION
\* Uncomment the previous line and provide the specification name if it's declared
\* in the specification file. Comment INIT / NEXT parameters if you use SPECIFICATION.

CONSTANTS
    greeting = "Hello"
    MAXINT = 1000
    u = 67
    v = 45
    defaultInitValue = 2000

INIT Init
NEXT Next

\* PROPERTY
\* Uncomment the previous line and add property names

INVARIANT
\* Uncomment the previous line and add invariant names
Qpc1
```

```
Qpc2
Qef
```

✎ Exercise 4 appex4_4.tla

We consider the following algorithm:

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, their product divided by their greatest common divisor.

Question 4.1 Show that this algorithm is partially correct with respect to its precondition and its postcondition, which you will state. To do so, translate this algorithm into a module from the PlusCal language.

Question 4.2 Show that it is free of run-time errors.

✔ Solution of exercise 4

Listing 13: TLA$^+$ module appex4_4.tla

```
---------------------- MODULE appex4_4 --------------------------------
EXTENDS Naturals, Integers, TLC
CONSTANT MAXINT,x10,x20

(*
--algorithm gcdscm {
  variables x1 =x10; (* 1st integer *)
            x2 = x20;  (* 2nd integer *)
            y1;
            y2;
            y3;
            y4;
            z1,z2;


{
     print <<x1, x2>>;
    y1:=x1;
    y2:=x2;
    y3:=x2;
    y4:=0;
    while (y1 /= y2) {

      while  ( y1 > y2) {
        y1:=y1-y2;
        y4:=y4+y3;
      };
      while  ( y2 > y1) {
        y2:=y2-y1;
        y3:=y4+y3;
      };

    };
    z1 := y1;
```

```
    z2 := y3+y4;
    print <<x1, x2,z1,z2>>
}

}
*)
\* BEGIN TRANSLATION (chksum(pcal) = "ab42efb3" /\ chksum(tla) = "9f011883")
CONSTANT defaultInitValue
VARIABLES x1, x2, y1, y2, y3, y4, z1, z2, pc

vars == << x1, x2, y1, y2, y3, y4, z1, z2, pc >>

Init == (* Global variables *)
        /\ x1 = x10
        /\ x2 = x20
        /\ y1 = defaultInitValue
        /\ y2 = defaultInitValue
        /\ y3 = defaultInitValue
        /\ y4 = defaultInitValue
        /\ z1 = defaultInitValue
        /\ z2 = defaultInitValue
        /\ pc = "Lbl_1"

Lbl_1 == /\ pc = "Lbl_1"
         /\ PrintT(<<x1, x2>>)
         /\ y1' = x1
         /\ y2' = x2
         /\ y3' = x2
         /\ y4' = 0
         /\ pc' = "Lbl_2"
         /\ UNCHANGED << x1, x2, z1, z2 >>

Lbl_2 == /\ pc = "Lbl_2"
         /\ IF y1 /= y2
               THEN /\ pc' = "Lbl_3"
                    /\ UNCHANGED << z1, z2 >>
               ELSE /\ z1' = y1
                    /\ z2' = y3+y4
                    /\ PrintT(<<x1, x2,z1',z2'>>)
                    /\ pc' = "Done"
         /\ UNCHANGED << x1, x2, y1, y2, y3, y4 >>

Lbl_3 == /\ pc = "Lbl_3"
         /\ IF y1 > y2
               THEN /\ y1' = y1-y2
                    /\ y4' = y4+y3
                    /\ pc' = "Lbl_3"
               ELSE /\ pc' = "Lbl_4"
                    /\ UNCHANGED << y1, y4 >>
         /\ UNCHANGED << x1, x2, y2, y3, z1, z2 >>

Lbl_4 == /\ pc = "Lbl_4"
         /\ IF y2 > y1
               THEN /\ y2' = y2-y1
                    /\ y3' = y4+y3
                    /\ pc' = "Lbl_4"
               ELSE /\ pc' = "Lbl_2"
                    /\ UNCHANGED << y2, y3 >>
         /\ UNCHANGED << x1, x2, y1, y4, z1, z2 >>

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

Next == Lbl_1 \/ Lbl_2 \/ Lbl_3 \/ Lbl_4
           \/ Terminating

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

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

\* END TRANSLATION


```

```
 Property == pc="Done"=> z1*z2 = x1*x2


Q1 == pc # "Done"
(* Qpc == pc ="Done" => z1=gcd(x1,x2) /\ z2=scm(x1,x2) *)
D == 0..MAXINT\cup {defaultInitValue}
Qpc1 == pc = "Done" => ( x1 % z1  = 0 ) /\ (x2 % z1 = 0)
Qpc2 == pc = "Done" => ( z2 % x1  = 0 ) /\ (z2 % x2 = 0)
Qproperty1 == pc = "Done" => x1*x2=z1*z2
Qef == x1 \in D /\ x2 \in D /\ y1 \in D /\ y3 \in D /\ z1 \in D /\ z2 \in D

=============================================================================
\* Modification History
\* Last modified Wed Nov 29 16:59:35 CET 2023 by mery
\* Created Wed Sep 09 17:02:47 CEST 2015 by mery
```

Listing 14: TLC configuration appex4_4.cfg

```

\* SPECIFICATION
\* Uncomment the previous line and provide the specification name if it's declared
\* in the specification file. Comment INIT / NEXT parameters if you use SPECIFICATION.

CONSTANTS
    greeting = "Hello"

x0 = 89
y0 = 789
min = 0
max = 888888
defaultInitValue = 99999999

INIT Init
NEXT Next

\* PROPERTY
\* Uncomment the previous line and add property names

\* INVARIANT
\* Uncomment the previous line and add invariant names
```

Listing 15: TLA$^+$ module appex4_5.tla

```
---------------------- MODULE appex4_5 -------------------------------
EXTENDS Naturals, Integers, TLC
CONSTANT MAXINT,x10,x20

(*
--algorithm gcdscm {
  variables x1 =x10; (* 1st integer *)
            x2 = x20;  (* 2nd integer *)
            y1;
            y2;
            y3;
            y4;
            z1,z2;


{
     print <<x1, x2>>;
    y1:=x1;
    y2:=x2;
    y3:=x2;
    y4:=0;
    while (y1 /= y2) {

      while  ( y1 > y2) {
        y1:=y1-y2;
        y4:=y4+y3;
      };
      while  ( y2 > y1) {
        y2:=y2-y1;
```

```
        y3:=y4+y3;
      };

    };
    z1 := y1;
    z2 := y3+y4;
    print <<x1, x2,z1,z2>>
}

}
*)
\* BEGIN TRANSLATION (chksum(pcal) = "ab42efb3" /\ chksum(tla) = "9f011883")
CONSTANT defaultInitValue
VARIABLES x1, x2, y1, y2, y3, y4, z1, z2, pc

vars == << x1, x2, y1, y2, y3, y4, z1, z2, pc >>

Init == (* Global variables *)
        /\ x1 = x10
        /\ x2 = x20
        /\ y1 = defaultInitValue
        /\ y2 = defaultInitValue
        /\ y3 = defaultInitValue
        /\ y4 = defaultInitValue
        /\ z1 = defaultInitValue
        /\ z2 = defaultInitValue
        /\ pc = "Lbl_1"

Lbl_1 == /\ pc = "Lbl_1"
         /\ PrintT(<<x1, x2>>)
         /\ y1' = x1
         /\ y2' = x2
         /\ y3' = x2
         /\ y4' = 0
         /\ pc' = "Lbl_2"
         /\ UNCHANGED << x1, x2, z1, z2 >>

Lbl_2 == /\ pc = "Lbl_2"
         /\ IF y1 /= y2
               THEN /\ pc' = "Lbl_3"
                    /\ UNCHANGED << z1, z2 >>
               ELSE /\ z1' = y1
                    /\ z2' = y3+y4
                    /\ PrintT(<<x1, x2,z1',z2'>>)
                    /\ pc' = "Done"
         /\ UNCHANGED << x1, x2, y1, y2, y3, y4 >>

Lbl_3 == /\ pc = "Lbl_3"
         /\ IF y1 > y2
               THEN /\ y1' = y1-y2
                    /\ y4' = y4+y3
                    /\ pc' = "Lbl_3"
               ELSE /\ pc' = "Lbl_4"
                    /\ UNCHANGED << y1, y4 >>
         /\ UNCHANGED << x1, x2, y2, y3, z1, z2 >>

Lbl_4 == /\ pc = "Lbl_4"
         /\ IF y2 > y1
               THEN /\ y2' = y2-y1
                    /\ y3' = y4+y3
                    /\ pc' = "Lbl_4"
               ELSE /\ pc' = "Lbl_2"
                    /\ UNCHANGED << y2, y3 >>
         /\ UNCHANGED << x1, x2, y1, y4, z1, z2 >>

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

Next == Lbl_1 \/ Lbl_2 \/ Lbl_3 \/ Lbl_4
           \/ Terminating

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

```

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

\* END TRANSLATION


 Property == pc="Done"=> z1*z2 = x1*x2


Q1 == pc # "Done"
(* Qpc == pc ="Done" => z1=gcd(x1,x2) /\ z2=scm(x1,x2) *)
D == 0..MAXINT\cup {defaultInitValue}
Qpc1 == pc = "Done" => ( x1 % z1  = 0 ) /\ (x2 % z1 = 0)
Qpc2 == pc = "Done" => ( z2 % x1  = 0 ) /\ (z2 % x2 = 0)
Qproperty1 == pc = "Done" => x1*x2=z1*z2
Qef == x1 \in D /\ x2 \in D /\ y1 \in D /\ y3 \in D /\ z1 \in D /\ z2 \in D

=============================================================================
\* Modification History
\* Last modified Wed Nov 29 16:59:35 CET 2023 by mery
\* Created Wed Sep 09 17:02:47 CEST 2015 by mery
```

Listing 16: TLC configuration appex4_5.cfg

```

\* SPECIFICATION
\* Uncomment the previous line and provide the specification name if it's declared
\* in the specification file. Comment INIT / NEXT parameters if you use SPECIFICATION.

CONSTANTS
    greeting = "Hello"

x0 = 89
y0 = 789
min = 0
max = 888888
defaultInitValue = 99999999

INIT Init
NEXT Next

\* PROPERTY
\* Uncomment the previous line and add property names

\* INVARIANT
\* Uncomment the previous line and add invariant names
```