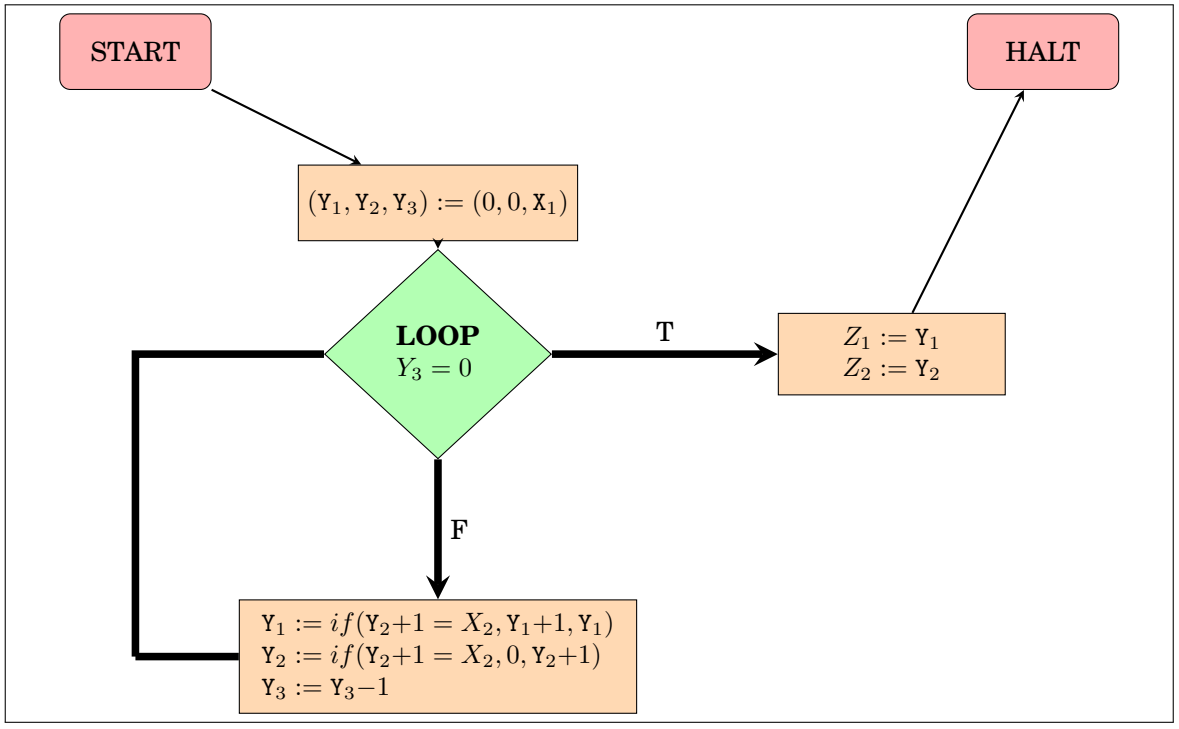

FIGURE 1 – Organigramme de calcul de la division entière

Cours MOdélisation, Vérification et EXpérimentations
Exercices
Série 1 Annotation, modélisation, vérification - Validation en TLA$^+$
par Dominique Méry
10 janvier 2025

TD1

Exercice 1 *(malgtd1ex1)*
Le PGCD de deux nombres vérifie les propriétés suivantes :

- $\forall a, b \in \mathbb{N}.pgcd(a, b) = pgcd(b, a)$
- $\forall a, b \in \mathbb{N}.pgcd(a, a{+}b) = pgcd(a, b)$
- *Ecrire une spécification TLA$^+$ calculant le PGCD de deux nombres donnés.*
- *Donner une explication ou une justification de la correction de cette solution*

Exercice 2 *(malgtd1ex2)*
L'accès à une salle est contrôlé par un système permettant d'observer les personnes qui entrent ou qui sortent de cette salle. Ce système est un ensemble de capteurs permettant d'identifier le passage d'une personne de l'extérieur vers l'intérieur et de l'intérieur à l'extérieur. Le système doit garantir qu'au plus `max` *personnes soient dans la salle. Ecrire un module TLA$^+$ permettant de modéliser un tel système respectant la propriété attendue.*

Exercice 3 *(malgtd1ex3)*
On considère l'algorithme suivant décrit par un organigramme ou flowchart de la figure 1. Cet algorithme calcule le reste et le quotient de la division de x_1 par x_2 : $0 \leq z_2 \leq x_2 \wedge x_1 = z_1 \cdot x_2 + z_2$. On suppose que x_1 et x_2 sont positifs et non nuls.

Question 3.1 *Donner la précondition et la postcondition associées à cet algorithme.*

Question 3.2 *Traduire cet algorithme sous forme d'un module TLA$^+$.*

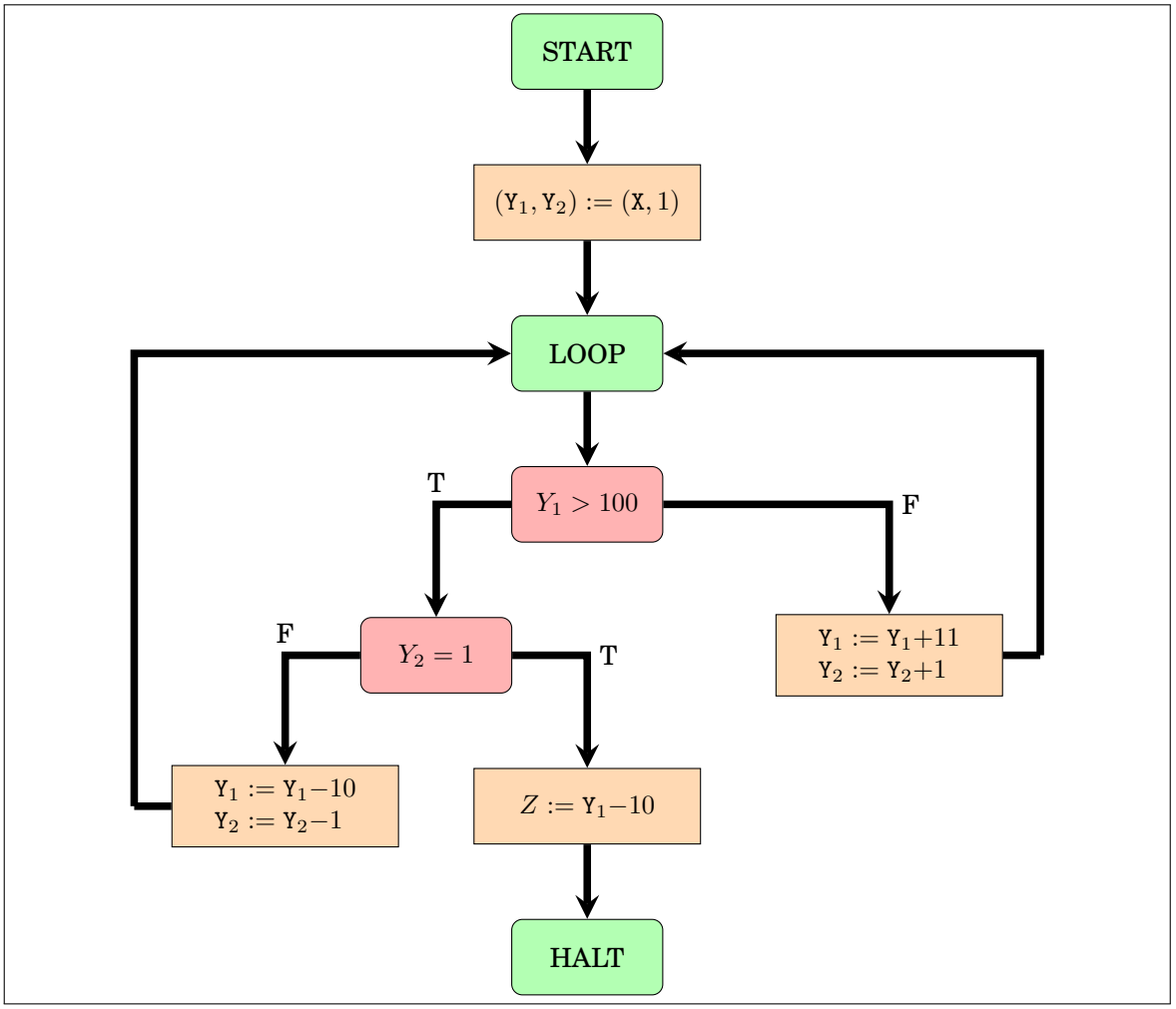

FIGURE 2 – Flowchart du calcul de la fonction de McCarthy

Question 3.3 *Tester les valeurs des variables à l'exécution.*

Question 3.4 *Montrer que cet algorithme est partiellement correct par rapport à sa précondition et à sa postcondition qu'il faudra énoncer.*

TD2

Exercice 4 *(malgtd1ex4)*
La fonction de McCarthy $f91$ *est définie pour tout entier* x $f91(x) = if\ x > 100\ then\ x-10\ else\ 91\ fi$.

Question 4.1 *Définir le contrat é'tablissant la correction partielle de l'algorithme* $ALG91$ *de la figure 2 qui est réputé calculer la fonction* $f91$

Question 4.2 *Construire un module TLA*$^+$ *modélisant les différents pas de calcul.*

Question 4.3 *Evaluer l'algorithme en posant des questions de sûreté suivantes :*

1. *l'algorithme est partiellement correct.*
2. *l'algorithme n'a pas d'erreurs à l'exécution.*

Exercice 5 *(malgtd1ex5 et inmalgtd1ex5)*

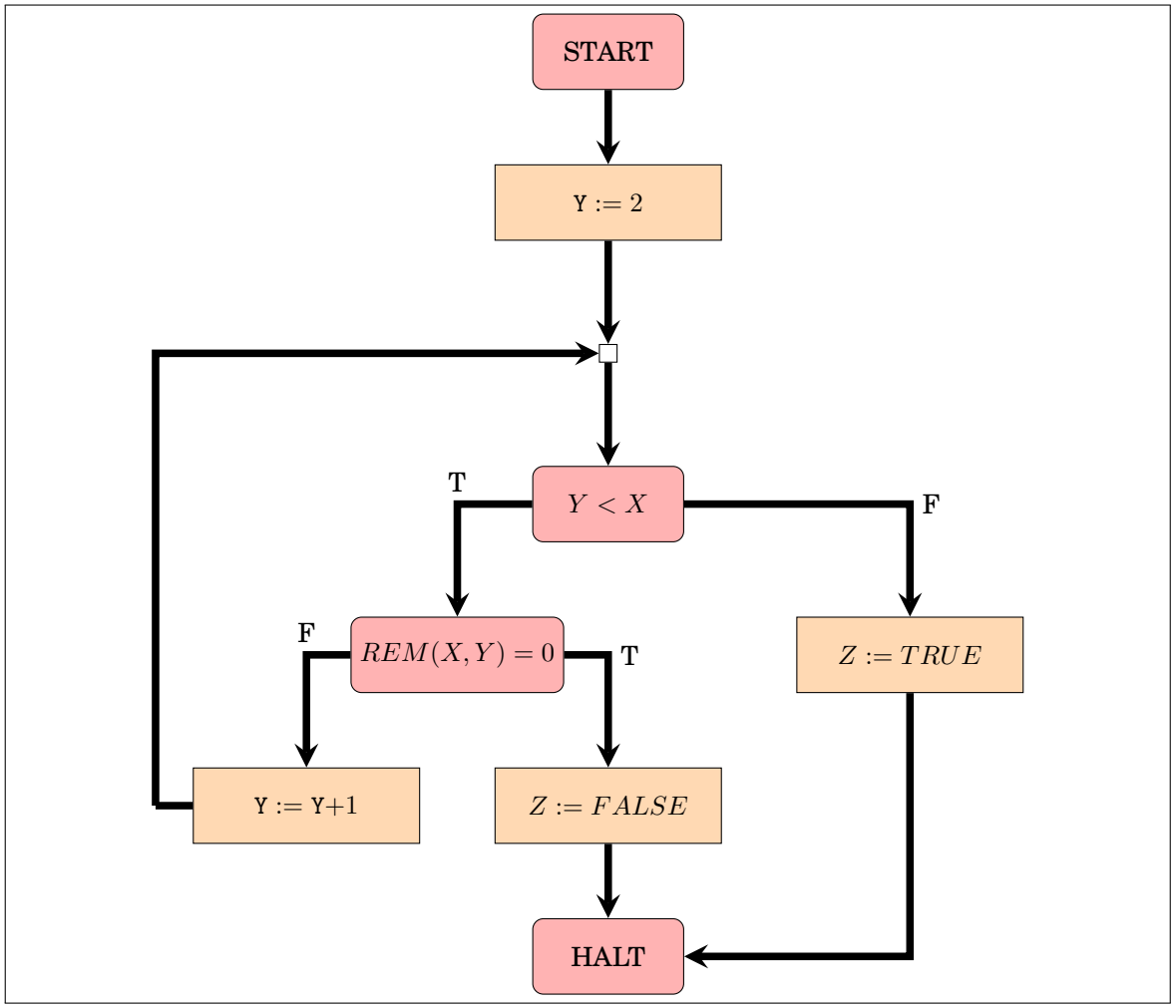

FIGURE 3 – Flowchart pour le test de primalité

Soit le schéma de la figure 3 définissant un calcul déterminant, si un nombre entier naturel est premier ou non.

Question 5.1 *Ecrire un module TLA/TLA$^+$ modélisant ce schéma de calcul et montrer que le modèle est sans blocage.*

Question 5.2 *Définir la propriété prime(x) qui est vraie si x est premier et faux sinon.*

Question 5.3 *Ecrire le contrat présumé du calcul du flowchart de la figure 3*

Question 5.4 *Vérifier la correction partielle*

Question 5.5 *Vérifier l'absence d'erreurs à l'exécution.*

Exercice 6 *Dans cet exercice, il est question de découvrir les modules de base de TLA Toolbox comme TLC, Integers, Naturals . . . afin de découvrir les fonctions qui sont prédéfinies.*

TD3

Exercice 7 *(Utilisation de ToolBox et TLA pour un labyrinthe, malgtd1ex7)*
Le module truc permet de résoudre un problème très classique en informatique : trouver un chemin entre un sommet input et des sommets output supposés être des sommets de sortie.

Question 7.1 *Pour trouver un chemin de input à l'un des sommets de output, il faut poser une question de sûreté à notre système de vérification. Donner une question de sûreté à poser permettant de trouver un chemin de input vers un sommet de output.*

Question 7.2 *On désire utiliser cette technique pour trouver un chemin dans un labyrinthe. Un labyrinthe est représenté par une matrice carrée de taille n. On définit ensuite pour chaque élément $<< i, j >>$ de la matrice les voisins communiquant à l'aide de la fonction lab qui associe à $<< i, j >>$ les éléments qui peuvent être atteints en un coup. Par exemple, le mouvement possible à partir de $<< 1, 1 >>$ est $<< 2, 1 >>$, ou le mouvement possible à partir de $<< 2, 1 >>$ est $<< 2, 2 >>$ ou $<< 1, 1 >>$, ou le mouvement possible à partir de $<< 2, 2 >>$ est $<< 2, 3 >>$ ou $<< 3, 2 >>$ ou $<< 2, 1 >>$, . . .*

```
lab == [<<x,y>> \in (nodes \X nodes) |->
                      IF x=1 /\ y=1  THEN {<<2,1>>} ELSE
                      IF x=2 /\ y=1  THEN {<<2,2>>}
                      IF x=1 /\ y=2  THEN {} ELSE
                      IF x=2 /\ y=2  THEN {<<3,2>>,<<2,3>>} ELSE
                      ELSE {}

                      ]
```

Modifier le module truc pour traiter ce problème et donner la question à poser pour trouver une sortie.

MODULE *truc*

EXTENDS *Integers*, *TLC*
VARIABLES p
CONSTANTS *input*, *output*

$n \triangleq 10$
nodes $\triangleq 1..n$
$l \triangleq [i \in 1..n \mapsto$ IF $i = 1$ THEN $\{4, 5\}$ ELSE
IF $i = 2$ THEN $\{6, 7, 10\}$ ELSE

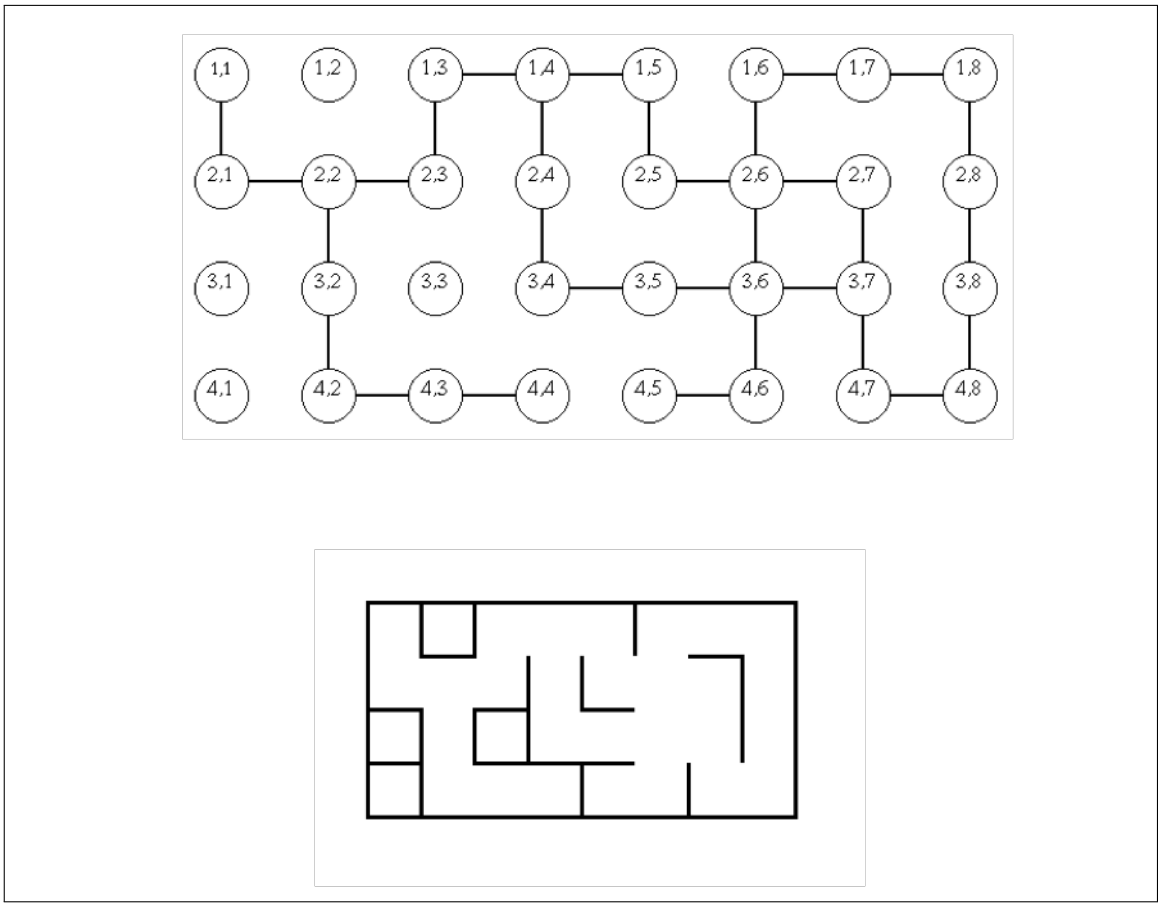

FIGURE 4 – Labyrinthe

IF $i = 4$ THEN $\{7, 8\}$ ELSE
IF $i = 5$ THEN $\{\}$ ELSE
IF $i = 6$ THEN $\{4\}$ ELSE
IF $i \ = \ 7$ THEN $\{5\}$ ELSE
IF $i \ = \ 8$ THEN $\{5,\ 2\}$ ELSE
$\{\}$
]

Init $\triangleq\ p\ =\ 1$
$M(i)\ \triangleq\ \wedge i\ \in\ l[p]$
$\wedge\ p' = i$
Next $\triangleq\ \exists\, i\ \in\ 1..n:\ M(i)$

Exercice 8 *(malgtd1ex10,malgtd1ex10bis,malgtd1ex10ter,malgtd1ex10last)*
Pour montrer que chaque annotation est correcte ou incorrecte, on propose de procéder comme suit :

— *Traduire cette annotation sous la forme d'un contrat.*
— *Vérifier les conditions de vérification du contrat*

$\ell_1 : P_{\ell_1}(v)$
v := f(v, c)
$\ell_2 : P_{\ell_2}(v)$

variables v
requires $pre(v_0)$
ensures $post(v_0, v_f)$
begin
$\ell_1 : Q_1(v_0, v)$
v := f(v, c)
$\ell_2 : Q_2(v_0, v)$
end

— $pre(v_0) \equiv P_{\ell_1}(v_0)$
— $post(v_0, v_f) \equiv P_{\ell_2}(v_f)$.
— $Q_{\ell_1}(v_0, v) \equiv P_{\ell_1}(v) \wedge v = v_0$
— $Q_{\ell_2}(v_0, v) \equiv P_{\ell_2}(v)$

On rappelle qu'un contrat est valide si les trois conditions suivantes sont valides :

— *(init)* $pre(v_0) \wedge v = v_0 \Rightarrow Q_1(v_0, v)$
— *(concl)* $pre(v_0) \wedge Q_2(v_0, v) \Rightarrow post(v_0, v)$
— *(induct)* $pre(v_0) \wedge Q_1(v_0, v) \wedge cond_{\ell_1,\ell_2}(v) \wedge v' = f(v, c) \Rightarrow Q_2(v_0, v')$

Les deux propriétés (init) et (concl) sont valides par construction et la seule propriété à montrer correcte ou incorrecte est la propriété (induct).

Question 8.1 *(malgtd1ex10)*

$\ell_1 : x = 3\ \wedge\ y = z{+}x\ \wedge z = 2{\cdot}x$
$y := z{+}x$
$\ell_2 : x = 3\ \wedge\ y = x{+}6$

Question 8.2 *(malgtd1ex10bis)*
Pour les deux exemples qui suivent, on considère dex cas et on doit donner une interprétation.

$\ell_1 : x = 2^4\ \wedge\ y = 2\ \wedge\ x{\cdot}y = 2^6$
$x := y{+}x{+}2^x$
$\ell_2 : x = 2^{10}\ \wedge\ y = 2$

$\ell_1 : x = 2^4 \ \wedge \ y = 2 \ \wedge \ x{\cdot}y = 2^5$
$x := y{+}x{+}2^x$
$\ell_2 : x = 2^{10} \ \wedge \ y = 2$

Question 8.3 *(malgtd1ex10ter.tla)*

$\ell_1 : x = 1 \ \wedge \ y = 12$
$x := 2{\cdot}y{+}x$
$\ell_2 : x = 1 \ \wedge \ y = 25$

Question 8.4 *(malgtd1ex10last.tla)*

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

Exercice 9 *(malgtd1ex11)*

Variables : X,Y,Z
Requires : $x_0, y_0 \in \mathbb{N} \ \wedge z_0 \in \mathbb{Z}$
Ensures : $z_f = max(x_0, y_0)$

$\ell_0 : \{\ldots\}$
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: maximum de deux nombres non annotée

Question 9.1 *Ecrire un module TLA$^+$ qui traduit la relation de transition de cet algorithme selon les instructions.*

Question 9.2 *Compléter l'algorithme 9 en l'annotant.*

Question 9.3 *Vérifier que l'annotation est correcte.*

Question 9.4 *Enoncer et vérifier la correction partielle et montrer que le contrat de correction partielle est satisfait.*

Question 9.5 *Compléter le module TLA$^+$ en définissant l'invariant construit avec les annotations et vérifier le contrat.*

Exercice 10 *Montrer que chaque annotation est correcte ou incorrecte selon les conditions de vérifications énoncées comme suit*
$\forall x, y, , x', y'.P_\ell(x, y) \ \wedge \ cond_{\ell,\ell'}(x, y) \ \wedge \ (x', y') = f_{\ell,\ell'}(x, y) \Rightarrow P_{\ell'}(x', y')$

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

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

— *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 11 *Montrer que chaque annotation est correcte ou incorrecte selon les conditions de vérifications énoncées comme suit*
$\forall x, y, , x', y'.P_\ell(x, y) \ \wedge \ cond_{\ell,\ell'}(x, y) \ \wedge \ (x', y') = f_{\ell,\ell'}(x, y) \Rightarrow P_{\ell'}(x', y')$

— *(1)* $\ell_1 : x = 9 \ \wedge \ y = z{+}x$
$y := x{+}9$
$\ell_2 : x = 9 \ \wedge \ y = x{+}9$

— $\ell_1 : x = 3 \ \wedge \ y = 3$
$x := y{+}x$
$\ell_2 : x = 6 \ \wedge \ y = 3$

— *(2)* $\ell_1 : x = 1 \ \wedge \ y = 3 \ \wedge \ x{+}y = 12$
$x := y{+}x$
$\ell_2 : x = 567 \ \wedge \ y = 34$

— $\ell_1 : x = 1 \ \wedge \ y = 3$
$z := x; x := y; y := z;$
$\ell_2 : x = 3 \ \wedge \ y = 1$

TD4

Exercice 12 *(malgtd1ex12) Dans l'algorithm 12, on calcule le maximum d'une suite de valeurs entières. On vous demande :*

- *Définir la précondition et la postcondition.*
- *Annoter cet algorithme*
- *Vérifier les conditions de vérification pour la correction partielle*
- *Vérifier les conditions pour l'absence d'erreurs à l'exécution*
- *Ecrire un module TLA pour valider ce qui a été prouvé.*

Exercice 13 *(malgtd1ex13)*
Soit l'algorithme 13 de la boucle bornée. On demande

1. *de définir le contrat de cet algorithme*
2. *d'annoter l'algorithme.*
3. *de vérifier les conditions de vérification.*
4. *de proposer un modèle TLA$^+$ pour vérifier les annotations et la correction partielle*

On rappelle qu'un contrat pour la correction partielle d'un petit programme est donné par les éléments ci-dessou en colonne de gauche et que les conditions de vérification associées sont définies par le texte de la colonne de droite.

Variables : F,N,M,I

Requires : $\left(\begin{array}{l} n_0 \in \mathbb{N} \wedge \\ n_0 \neq 0 \wedge \\ f_0 \in 0 \mathinner{..} n_0{-}1 \rightarrow \mathbb{N} \end{array}\right)$

Ensures : $\left(\begin{array}{l} m_f \in \mathbb{N} \wedge \\ m_f \in ran(f_0) \wedge \\ (\forall j \cdot j \in 0 \mathinner{..} n_0{-}1 \Rightarrow f_0(j) \leq m_f) \end{array}\right)$

$M := F(0)$;
$I := 1$;
while $I < N$ **do**
  if ***F(i)>M*** **then**
    $M := F(I)$;
  ;
  $I{+}{+}$;
;

Algorithme 2: Algorithme du maximum d'une liste non annotée

Variables : X
Requires : $x_0 \in \mathbb{N}$
Ensures : $x_f = 0$

$\ell_0 : \{\ldots\}$
while $0 < X$ **do**
  $\ell_1 : \{\ldots\}$
  $X := X{-}1$;
  $\ell_2 : \{\ldots\}$
;
$\ell_3 : \{\ldots\}$

Algorithme 3: Exemple de la boucle bornée

Contrat de la correction partielle

variables $type\ X$

---

definitions

$def1 \stackrel{def}{=} text1$

---

requires $pre(x_0)$

---

ensures $post(x_0, x_f)$

---

begin
$0 : P_0(x_0, x)$
$\texttt{instruction}_0$
$1 : P_i(x_0, x)$
$\texttt{instruction}_1$
$f : P_f(x_0, x)$
end

Conditions de vérification

- $pre(x_0) \wedge x = x_0 \Rightarrow P_0(x_0, x)$
- $pre(x_0) \wedge P_f(x_0, x) \Rightarrow post(x_0, x)$
- Pour toutes les paires ℓ, ℓ', telles que $\ell \longrightarrow \ell'$, on vÃ©rifie que, pour toutes les valeurs $x, x' \in$ MEMORY
$$\left(\begin{array}{l} \left(\begin{array}{l} pre(x_0) \wedge P_\ell(x_0, x)) \\ \wedge cond_{\ell,\ell'}(x) \wedge x' = f_{\ell,\ell'}(x) \end{array} \right) \\ \Rightarrow P_{\ell'}(x_0, x') \end{array} \right),$$

Exercice 14 *(6 points)*
Soit le contrat suivant qui met en jeu les variables X,Y, Z,C,R.

VARIABLES $\texttt{int}\ X, Y, Z, C, R$

REQUIRES $x_0, y_0, z_0, c_0, r_0 \in \mathbb{Z}$

ENSURES $r_f = 0$

BEGIN
$0 : x = x_0 \wedge y = y_0 \wedge z = z_0 \wedge c = c_0 \wedge r = r_0 \wedge x_0, y_0, z_0, c_0, r_0 \in \mathbb{Z}$
$(X, Z, Y) := (49, 2{\cdot}C, (2{\cdot}C{+}1){\cdot}(2{\cdot}C{+}1));$
$1 : x = 49 \wedge z = 2{\cdot}c \wedge y = (z{+}1){\cdot}(z{+}1)$
$Y := X{+}Z{+}1;$
$2 : x = 49 \wedge z = 2{\cdot}c \wedge y = (c{+}1){\cdot}(c{+}1)$
END

Question 14.1 *Ecrire les conditions de vérification associée au contrat ci-dessus en vous aidant du rappel de la définition de ces conditions de vérification.*

Question 14.2 *Simplifier les conditions de vérification et préciser les conditions que doivent vérifier les valeurs initiales des variables X,Y,Z,C,R pour que les conditions de vérification soient toutes vraies. En particulier, il faudra s'assurer que la prÃ©condition est satisfaisable.*

Exercice 15 *(6 points)*

On considère le petit programme se trouvant à droite de cette colonne. Nous allons poser quelques questions visant à compléter les parties marquées en gras et visant à définir la relation de calcul.
On notera $pre(n_0, x_0, b_0)$ *l'expression* $n_0, x_0, b_0 \in \mathbb{Z}$ *et* $in(n, b, n_0, x_0, b_0)$ *l'expression* $n = n_0 \wedge b = b_0 \wedge pre(n_0, x_0, b_0)$

Question 15.1 *Donner l'assertion Requires en complétant ce qui est déjà mentionné et en reportant le texte complet de cette assertion Requires dans votre copie.*

On rappelle que la relation de transition de ℓ *vers* ℓ'*, notée* $a(\ell, \ell')$*, est définie par une relation de la forme* $cond_{\ell,\ell'}(v) \wedge v' = f_{\ell,\ell'}(v)$*.*

Question 15.2 *Ecrire les relations de transition entre les étiquettes successives :* $a(\ell_0, \ell_1)$*,* $a(\ell_1, \ell_2)$*,* $a(\ell_2, \ell_3)$*,* $a(\ell_3, \ell_6)$*,* $a(\ell_1, \ell_4)$*,* $a(\ell_4, \ell_5)$*,* $a(\ell_5, \ell_6)$*.*

Exercice 16 *(8 points)*

VARIABLES $int\ N, X, B$

REQUIRES $n_0, x_0, b_0 \in \mathbb{Z}$

ENSURES $\begin{cases} n_0 < b_0 \Rightarrow x_f = \textbf{\textit{question1}} \\ n_0 \geq b_0 \Rightarrow x_f = \textbf{\textit{question1}} \\ n_f = n_0 \wedge b_f = b_0 \end{cases}$

BEGIN
ℓ_0 :
  $X := N;$
ℓ_1 :
IF $X < B$ ***THEN***
  ℓ_2 :
$X := X \cdot X + 2 \cdot B \cdot X + B \cdot B;$
  ℓ_3 :
ELSE
  ℓ_4 :
    $X := B;$
  ℓ_5 :
FI
ℓ_6 :
END

VARIABLES N, V, S, I

DEFINITIONS

$$pre(n_0, v_0, s_0, i_0) \stackrel{def}{=} \begin{cases} n_0 \in \mathbb{N} \wedge n_0 \neq 0 \\ v_0 \in 0..n_0{-}1 \longrightarrow \mathbb{Z} \\ s_0 \in \mathbb{Z} \wedge i_0 \in \mathbb{Z} \end{cases}$$

REQUIRES $\begin{cases} n_0 \in \mathbb{N} \wedge n_0 \neq 0 \\ v_0 \in 0..n{-}1 \longrightarrow \mathbb{Z} \end{cases}$

ENSURES $\begin{cases} s_f = \bigcup_{k=0}^{n_0-1} v_0(k) \\ n_f = n_0 \\ v_f = v_0 \end{cases}$

BEGIN

$\ell_0 : \begin{cases} pre(n_0, v_0, s_0, i_0) \\ (n, v, s, i) = (n_0, v_0, s_0, i_0) \end{cases}$

$S := V(0)$

ℓ_1 :?????????

$I := 1$

$\ell_2 : \begin{cases} pre(n_0, v_0, s_0, i_0) \\ s = \bigcup_{k=0}^{i-1} v(k) \wedge i = 1 \\ (n, v) = (n_0, v_0) \end{cases}$

WHILE $I < N$ ***DO***

$\ell_3 : \begin{cases} pre(n_0, v_0, s_0, i_0) \\ s = \bigcup_{k=0}^{i-1} v(k) \wedge i \in 1..n{-}1 \\ (n, v) = (n_0, v_0) \end{cases}$

$S := S \oplus V(I)$

ℓ_4 :?????

$I := I{+}1$

ℓ_5 :?????

OD;

ℓ_6 :?????

END

La notation $\bigcup_{k=i}^{j} v(k)$ *désigne la valeur maximale des éléments* $\{v(k) | k \in i..j\}$ *et on suppose que l'opérateur* $\oplus$ *renvoie pur deux valeurs entières, la valeut maximale.*

Question 16.1 *Compléter les annotations incomplètes* ℓ_1, ℓ_4 , ℓ_5 *et* ℓ_6.

Question 16.2 *Vérifier les conditions de vérification associées aux transitions suivantes :*

1. ℓ_0, ℓ_1
2. ℓ_2, ℓ_3
3. ℓ_3, ℓ_4
4. ℓ_5, ℓ_6

Question 16.3 *Donner et vérifier les points pour assurer la correction partielle de cet algorithme.*

Question 16.4 *Que faut-il faire pour vérifier que cet algorithme est bien annoté et qu'il est partiellement correct en utilisant TLA*$^+$ *? Expliquer simplement les éléments à mettre en œuvre et les propriétés de sûreté à vérifier.*

Fin de la série 1