

Cours MOdélisation, Vérification et EXpérimentations  
Exercices  
Sémantique des langages de programmation  
par Dominique Méry  
13 mai 2025

## Exercices sur Frama-c et wp

**Exercice 1 Question 1.1** Soit le petit programme suivant annoté mais incomplet.

```
/*@ requires A(x,y,z) ;
   ensures \result == 49 ;
*/

int q6(int x, int y, int z){

    z = y*(x+y);
    y = x*y;
    x=x*x;
    z = z+x+y;
   /*@ assert z == 49; */

    return z;
}
```

En utilisant l'opérateur wp, proposer des assertions pour  $A(x, y, z)$ , afin que le contrat soit correct.

**Question 1.2** Soit le petit programme suivant annoté mais incomplet.

```
/*@ requires A(x,y,z);
   ensures \result == 144 ;
*/

int q6(int x, int y, int z){
    int u;
    u = x+y+z;
    x=x*x;
    /*@ assert x == 9; */
    y=y*y;
    /*@ assert y == 16; */
    z=z*z;
    u = u*u;
    return u;
}
```

En utilisant l'opérateur wp, proposer une assertion pour  $A$ , afin que le contrat soit correct. Les annotations indiquées sont correctes et font partie des données du problème.

**Exercice 2** Nous étudions ce petit algorithme qui calcule quelque chose et nous avons exécuté cet algorithme de 0 et 10 pour obtenir la suite suivante :

0 --> 0, 1 --> 1, 2 --> 3, 3 --> 7, 4 --> 15, 5 --> 31, 6 --> 63, 7 --> 127, 8 --> 255,

```
#ifndef _A_H
#define _A_H
// Definition of the mathematical function mathpower2
/*@ axiomatic mathpower {
```

```
@ logic integer mathpower(integer n, integer m);
@ axiom mathpower_0: \forall integer n; n >= 0 ==> mathpower(n,0) == 1;
@ axiom mathpower_in: \forall integer n,m; n >= 0 && m >= 0
==> mathpower(n,m+1) == mathpower(n,m)*n;
@ } */
```

```
int inv1(int x);
#endif
```

```
#include <limits.h>
#include <qmathiinv1.h>
```

```
int inv1(int x)
{ int u=0;
  int k=0;
  while (k < x)
  { u=2*u+1;
    k=k+1;
  };
  return(u);
}
```

Si on utilise la fonction power2, on obtient la suite suivante :

0 --> 1, 1 --> 2, 2 --> 4, 3 --> 8, 4 --> 16, 5 --> 32, 6 --> 64, 7 --> 128, 8 --> 256, 9 -->

**Question 2.1** On comprend que l'algorithme calcule la suite  $u_n$  d'entiers telle que  $\forall n \in \mathbb{N}$  :  $u_n = 2^n - 1$ . En particulier,  $u_0 = 0$ .

Donner une définition de  $u_{n+1}$  en fonction de  $u_n$  en calculant le rapport  $\frac{u_{n+1}+1}{u_n+1}$

**Question 2.2** Ecrire un contrat pour cette algorithme en précisant la clause requires et la clause ensures.

**Question 2.3** Proposer un invariant de boucle en vous aidant de la suite  $u_n$  et montrer qu'il est correct par cette preuve de correction.

**Question 2.4** Exprimer la terminaison de cet algorithme et justifier qu'il termine pour la précondition choisie.

**Exercice 3** On dit que  $S1$  est équivalent à  $S2$  et on note  $S1 \equiv S2$ , si pour tous les états  $s$  et  $s'$ ,  $(S1, s) \xrightarrow{\text{nat}} s'$  si, et seulement si,  $(S2, s) \xrightarrow{\text{nat}} s'$ .

**Question 3.1** Montrer que  $\text{while } b \text{ do } S \text{ od} \equiv \text{if } b \text{ then } S; \text{while } b \text{ do } S \text{ od else skip fi}$

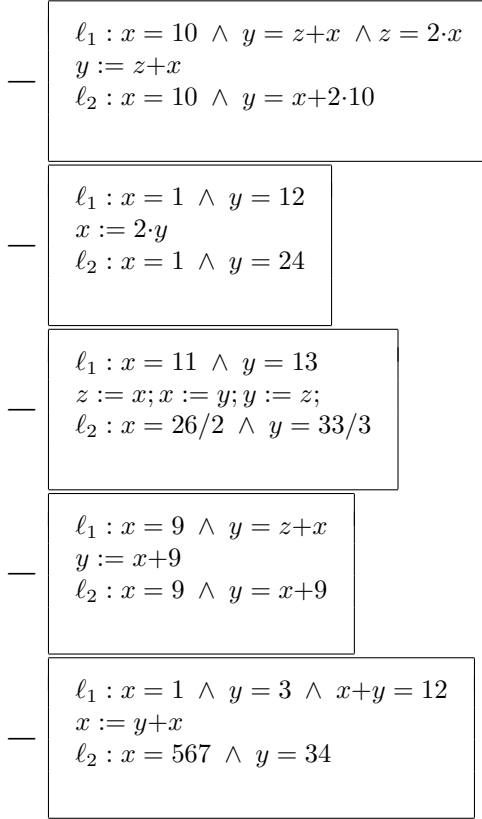
**Question 3.2** Etendre la fonction sémantique pour l'instruction  $\text{repeat } S \text{ until } b$ .

**Question 3.3** Montrer que  $\text{repeat } S \text{ until } b \equiv S; \text{if } b \text{ then skip else repeat } S \text{ until } b \text{ fi}$

**Exercice 4** On rappelle que  $wp(X := E)(P(x)) = P[e(x)/x]$  et que  $\{A(x)\}X := E\{B(x)\}$  est définie par  $A \Rightarrow wp(X := E)(B)$ . On peut assez naturellement appliquer cette définition pour

```
ℓ1 : A(x)
X := E(X)
ℓ2 : B(x)
```

Montrer la correction des triplets suivants et vérifier avec Frama-C en examinant les conditions de vérification engendrées :



**Exercice 5** Calculer  $wp(S)(P)$  dans les cas suivants :

1.  $wp(X := E(X); Y := F(X))(P(x, y))$
2.  $wp(X := Y, Y := X)(P(x, y))$
3.  $wp(\text{while } TRUE \text{ do } X := E(X) \text{ od})(P(x, y))$
4.  $wp(\text{while } FALSE \text{ do } X := E(X) \text{ od})(P(x, y))$
5.  $wp(\text{while } x < 20 \text{ do } X := X+1 \text{ od})(TRUE)$

## Sémantique naturelle et sémantique SOS

**Exercice 6**

$n ::= 0 \mid 1 \mid n0 \mid n1$   
 $e ::= n \mid x \mid e1+e2 \mid e1-e2 \mid e1 \cdot e2$   
 $b ::= tt \mid ff \mid e1 = e2 \mid e1 \neq e2 \mid e1 \leq e2 \mid e1 \geq e2 \mid e1 < e2 \mid e1 > e2 \mid \neg b \mid b1 \ \&\& \ b2$   
 $S ::= x := e \mid skip \mid S1; S2 \mid (\text{if } b \text{ then } S_1 \text{ else } S_2 \text{ fi} \mid \text{while } b \text{ do } S \text{ od})$

**Question 6.1** Définir une fonction sémantique pour la catégorie syntaxique des chaînes numériques  $NUM$  à valeurs dans  $\mathbb{Z} : \mathcal{N} \in NUM \rightarrow \mathbb{Z}$ .

**Question 6.2** Evaluer les valeurs suivantes :

- $\mathcal{N}(11)$
- $\mathcal{N}(101)$
- $\mathcal{N}(0100)$

**Question 6.3** Montrer que  $\mathcal{N}$  est bien définie pour toutes les expressions.

**Exercice 7** On définit l'ensemble des états  $States = Var \rightarrow \mathbb{Z}$  où  $Var$  est l'ensemble des variables.

**Question 7.1** Une expression arithmétique  $e \in \text{Exp}$  est évaluée dans un état  $s$  par la fonction sémantique  $\mathcal{E} \in \text{Exp} \rightarrow (\text{States} \rightarrow \mathbb{Z})$ . Définir  $\mathcal{E}$  par induction sur la syntaxe.

**Question 7.2** Soit  $s \in \text{States}$  tel que  $s(x) = 2$  et  $s(y) = 3$  où  $x, y \in \text{Var}$  et  $s \in \text{States}$ . Evaluer les expressions suivantes en  $s$  :  $x+y+101$ ,  $x \cdot y$ .

**Question 7.3** Une expression logique  $b \in \text{Bexp}$  est évaluée dans un état  $s$  par la fonction sémantique  $\mathcal{B} \in \text{Bexp} \rightarrow (\text{States} \rightarrow \mathbb{B})$ . Définir  $\mathcal{B}$  par induction sur la syntaxe.

**Question 7.4** Soit  $s \in \text{States}$  tel que  $s(x) = 2$  et  $s(y) = 3$  où  $x, y \in \text{Var}$  et  $s \in \text{States}$ . Evaluer les expressions suivantes en  $s$  :  $x = y$ ,  $x \neq y$ ,  $x \leq y$ ,  $x < y$  &  $x + 6 \leq y$ .

**Question 7.5** On étend le langage des expressions logiques par les deux constructions  $b_1 \Rightarrow b_2$  et  $b_1 \Leftrightarrow b_2$ . Ce langage est noté  $\text{Bexp}_1$ . Montrer que pour tout expression  $b \in \text{Bexp}_1$ , il existe une expression  $b' \in \text{Bexp}$  telle que  $\mathcal{B}(b) = \mathcal{B}(b')$ .

**Exercice 8** Nous définissons deux opérations substitution et mise à jour. Ces deux opérations seront utilisées plus tard dans l'expression de la sémantique des instructions :

- la notation de substitution  $e[x \mapsto e_1]$  qui est la substitution de  $x$  par  $e_1$  dans  $e$ .
- la mise à jour pour un état  $s$  et on la note  $s[x \mapsto v]$  qui est le nouvel état obtenu par mise à jour de la valeur de  $x$  pour  $s$ .

**Question 8.1** Ecrire une définition inductive de  $e[x \mapsto f]$ .

**Question 8.2** Définir la mise à jour pour un état  $s$  et on la note  $s[x \mapsto v]$  qui est le nouvel état obtenu par mise à jour de la valeur de  $x$  pour  $s$ .

**Question 8.3** Montrer que  $s[x \mapsto v][y \mapsto w] = s[y \mapsto w][x \mapsto v]$  et que  $s[x \mapsto v][x \mapsto w] = s[x \mapsto w]$ .

**Question 8.4** Montrer que  $\mathcal{E}(e[x \mapsto f])(s) = \mathcal{E}(e)(s[x \mapsto \mathcal{E}(f)(s)])$ .

**Question 8.5** Définir la substitution pour les expressions booléennes  $b[x \mapsto e]$  où  $b$  est une expression booléenne de  $\text{BExp}$  et  $e$  est une expression arithmétique de  $\text{Exp}$ .



**Question 8.6**

Montrer que  $\mathcal{E}(b[x \mapsto e])(s) = \mathcal{E}(b)(s[x \mapsto \mathcal{E}(e)(s)])$ .

**Exercice 9**

On rappelle les règles définissant la sémantique naturelle du langage de programmation  $\mathcal{PL}$

Règles de définition selon la syntaxe

Axiome Ass  $(x := e, s) \xrightarrow{\text{nat}} s[x \mapsto \mathcal{E}(e)(s)]$

Axiome Skip  $(\text{skip}, s) \xrightarrow{\text{nat}} s$

Règle Comp Si  $(S_1, s) \xrightarrow{\text{nat}} s'$  et  $(S_2, s') \xrightarrow{\text{nat}} s''$ , alors  $(S_1; S_2, s) \xrightarrow{\text{nat}} s''$ .

Règle Iftt Si  $(S_1, s) \xrightarrow{\text{nat}} s'$  et  $\mathcal{B}(b)(s) = \text{TRUE}$ , alors  $(\text{if } b \text{ then } S_1 \text{ else } S_2 \text{ fi}, s) \xrightarrow{\text{nat}} s'$ .

Règle Iffff Si  $(S_2, s) \xrightarrow{\text{nat}} s'$  et  $\mathcal{B}(b)(s) = \text{FALSE}$ , alors  $(\text{if } b \text{ then } S_1 \text{ else } S_2 \text{ fi}, s) \xrightarrow{\text{nat}} s'$ .

Règle Whilett  $\downarrow$  Si  $(S, s) \xrightarrow{\text{nat}} s'$  et  $(\text{while } b \text{ do } S \text{ od}, s') \xrightarrow{\text{nat}} s''$  et  $\mathcal{B}(b)(s) = \text{TRUE}$ , alors  $(\text{while } b \text{ do } S \text{ od}, s) \xrightarrow{\text{nat}} s''$ .

Règle Whiefff  $\downarrow$  Si  $\mathcal{B}(b)(s) = \text{FALSE}$ , alors  $(\text{while } b \text{ do } S \text{ od}, s) \xrightarrow{\text{nat}} s$ .

**Question 9.1** Soit  $s$  tel que  $s(u) = 0$  et  $s(v) = 1$ .

- Evaluer  $(u := 11; v := u + 100; u := u + v, s)$  en sémantique naturelle.
- Evaluer  $(w := u; u := v; v := w, s)$  en sémantique naturelle.