

Cours MOdélisation, Vérification et EXpérimentations  
Exercices  
Utilisation d'un environnement de vérification Frama-c (I)  
par Dominique Méry  
5 mars 2025

## TD5

### Annotations en Frama-c

#### Exercice 1 *framac/ex1.c*

Vérifier l'annotation suivante :

```
l1: x== 10 && y == z+x && z==2*x;  
    y= z+x;  
l2: x== 10 && y == x+2*10;  
    x = x+1;  
l3:  x-1== 10 && y == x-1+2*10;
```

#### Exercice 2 *framac/ex2.c*

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

```
int q1() {  
    int c = val ;  
l1: x == 2;  
    int x;  
l2: c == 2;  
    x = 3 * c ;  
l3: x == 6;  
    return(0);  
}
```

#### Exercice 3 Vérifier l'annotation suivante :

```
int main()  
{  
    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 4 Vérifier l'annotation suivante :

```
int main()  
{  
    int z;  
    int a = 4;  
l1:  a == 4 ;  
    int b = 3;  
l2:  b == 3 && a == 4;  
    int c = a+b;  
l3:  b == 3 && c == 7 && a == 4 ; */
```

```
    a += c;
    b += a;
l4:  a == 11 && b == 14 && c == 7 ;
l5:  a + b == 25 ;
    z = a * b;
l6:  a == 11 && b == 14 && c == 7 && z == 154;
    return(0);
}
```

**Exercice 5** 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);
}
```

### Contrats en Frama-c

**Exercice 6** Vérifier l'annotation suivante :

```
/*@ requires x0 >= 0;
    assigns \nothing;
    ensures \result == x0+1;
@*/

int exemple(int x0) {
    int x=x0;
    //@ assert ...;
    x = x + 2;
    //@ assert x== ...;
    return x;
}
```

**Exercice 7** Vérifier l'annotation suivante :

```
/*@
    requires x < 3 && x > 8;
    ensures \false;
*/
void fonc(int x){
}
}
```

### TD8-IL

**Exercice 8** Analyser et compléter l'annotation suivante pour qu'elle soit valide :

```

int annotation(int a,int b)
{
    int x,y,z;
    x = a;
l1:  x == a; */
    y = b;
l2:  x == a && y == b; */
    z = a+b-2;
l3:  x == a && y == b && z==4; */
    return(z);
}

```

**Exercice 9** Définir une fonction *abs* avec son contrat.

```

int abs ( int x ) {
    if ( x >=0 ) return x ;
    return -x ; }

```

**Exercice 10** Définir une fonction *max* avec son contrat.

```

int max ( int x, int y ) {
    if ( x >=y ) return x ;
    return y ; }

```

**Exercice 11** Soit l'algorithme annoté comme suit :

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

```

 $\ell_0 : \{x = x_0 \wedge y = y_0 \wedge z = z_0 \wedge x_0, y_0 \in \mathbb{N} \wedge z_0 \in \mathbb{Z}\}$ 
if  $X < Y$  then
     $\ell_1 : \{x < y \wedge x = x_0 \wedge y = y_0 \wedge z = z_0 \wedge x_0, y_0 \in \mathbb{N} \wedge z_0 \in \mathbb{Z}\}$ 
    Z := Y;
     $\ell_2 : \{x < y \wedge x = x_0 \wedge y = y_0 \wedge x_0, y_0 \in \mathbb{N} \wedge z_0 \in \mathbb{Z} \wedge z = y_0\}$ 
else
     $\ell_3 : \{x \geq y \wedge x = x_0 \wedge y = y_0 \wedge z = z_0 \wedge x_0, y_0 \in \mathbb{N} \wedge z_0 \in \mathbb{Z}\}$ 
    Z := X;
     $\ell_4 : \{x \geq y \wedge x = x_0 \wedge y = y_0 \wedge x_0, y_0 \in \mathbb{N} \wedge z_0 \in \mathbb{Z} \wedge z = x_0\}$ 
;
 $\ell_5 : \{z = \max(x_0, y_0) \wedge x = x_0 \wedge y = y_0 \wedge x_0, y_0 \in \mathbb{N} \wedge z_0 \in \mathbb{Z}\}$ 

```

**Algorithme 1:** maximum de deux nombres non annotée

**Question 11.1** Ecrire un programme ACSL qui traduit ce contrat et qui le vérifie.

**Question 11.2** Ecrire un programme ACSL qui traduit ce contrat et qui le vérifie mais qui enlève les assertions.

## Contrats avec invariants de boucle et ghosts en Frama-c

**Exercice 12** *Ecrire un contrat pour la fonction factorielle*

```
int codefact(int n) {  
    int y = 1;  
    int x = n;  
    while (x != 1) {  
        y = y * x;  
        x = x - 1;  
    };  
    return y;  
}
```

**Exercice 13** *Ecrire un contrat pour la fonction calculant le reste de la division de a par b.*

```
int reste(int a, int b) {  
    int r = a;  
    int q = 0;  
    while (r >= b) {  
        r = r - b;  
        q = q + 1;  
    };  
    return r;  
}
```