Spécifications ACSL, premiers pas

Polytech Nice-Sophia
Département informatique
Preuves de programmes
Sylvain Lippi

Nous allons utiliser la plate-forme `Frama-c` avec le plugin `jessie` afin de prouver des spécifications (si possibles complètes !) sur des petites fonctions écrites en *C*. En pratique, on ajoutera des spécifications écrites en ACSL sur les fonctions et on lancera `frama-c -jessie` sur le source ainsi annoté.

Exercice 1 *Le max de deux entiers*
On considère la fonction triviale suivante :

```
1   int max(int x, int y)
2   {
3       return x>y? x : y;
4   }
```

1. *Une propriété de cette fonction est que le résultat est supérieur aux deux arguments de cette fonction. Ajouter la spécification ACSL correspondant et la vérifier en lançant* `frama-c -jessie`.
2. *La spécification précédente est correcte mais pas complète. Quelle propriété supplémentaire doit vérifier la fonction* `max` *?*

Exercice 2 *Le max de deux entiers avec des pointeurs*
On propose une variante avec des pointeurs qui rend la valeur zéro en cas de succès et -1 *sinon.*

```
1   int max(int *r, int *x, int *y)
2   {
3       if (!r) return -1;
4       *r = (*x > *y) ? *x : *y;
5       return 0;
6   }
```

Annoter cette version. On pourra éventuellement utiliser `behaviour` *et* `assumes` *afin de considérer deux « scénarios » d'exécution distincts : ok et ko et ainsi alléger l'écriture des spécifications.*

Exercice 3 *Le max d'un tableau d'entiers*
On considère maintenant une fonction plus générale qui rend le plus petit indice correspondant à l'élément maximal d'un tableau d'entiers.

```
1   int max_element(const int *t, int n)
2   {
3       int max=0;
4
5       for (int i=0; i<n; i++)
6           if (t[max]<t[i]) max=i;
7
8       return max;
9   }
```

1. *Prouver que la fonction termine en utilisant un variant de boucle* ie *une valeur positive qui décroit à chaque itération.*
2. *Prouver l'absence d'accès d'un tableau en dehors de ses bornes. Pour cela, on supposera que l'argument* `n` *est strictement positif et que le tableau* `t` *est valide de l'indice 0 à* $n-1$.
   Indication. *Utiliser les invariants de boucle.*
3. *Prouver que la valeur retournée est comprise entre 0 et* $n-1$.
4. *Prouver que l'indice retourné correspond à un élément supérieur ou égal à tous les autres.*
   Indication. *Utiliser les invariants de boucle ainsi que le quantificateur* `\forall` *et l'implication logique* `==>`.
5. *Prouver que si plusieurs ont la même valeur maximale, l'indice retourné est le plus petit. En déduire, que tous les éléments d'indice strictement inférieur à l'indice retourné sont strictement plus petits que l'élément maximal.*