Preuve de programmes impératifs
TPs - Why3

Quentin Peyras

4 décembre 2023

1 Introduction à Why3

/\	et	forall i :int. ...	quantification universelle
\/	ou	exists i :int. ...	quantification existentielle
->	implique	old a	valeur de a avant application de la fonction
<->	équivalent	result	valeur de retour de la fonction

FIGURE 1 – Syntaxe des formules en Why3

Le langage utilisé par Why3 est un langage multi-paradigme appelé WhyML. Par exemple, on peut écrire une version impérative de l'algorithme de division euclidienne comme ceci :

```
let division (a b: int) : int
  =
    let ref q = 0 in
    let ref r = a in
    while r >= b do
      q <- q + 1;
      r <- r - b
    done;
    q
```

Une fonctionnalité importante de Why3 est de pouvoir annoter les programmes avec des post-conditions, des pré-conditions, des invariants et des variants. Par exemple on peut annoter notre fonction division comme suit :

```
let division (a b: int) : int
    requires { a >= 0 /\ b >= 0 }
    ensures  { exists r: int. a = b * result + r /\ 0 <= r < b }
  =
    let ref q = 0 in
    let ref r = a in
    while r >= b do
      invariant { true }
```

```
    variant {r}
    q <- q + 1;
    r <- r - b
  done;
  q
```

La syntaxe des annotations en Why3 est donnée par la figure **??**. Vous pouvez utiliser la version web de Why3 à l'adresse : https://why3.lri.fr/try/

2 Programmes avec types de base

2.1 Division euclidienne

Question 1. Ouvrez le fichier "div_euclid.mlw".

Question 2. Complétez l'invariant pour prouver la post-condition sans prouver la terminaison.

Question 3. Complétez les annotations pour prouvez la terminaison du programme.

2.2 Multiplication

Question 4. Ouvrez le fichier "mult.mlw", complétez les preconditions et postcondidions de la fonction.

Question 5. Complétez les annotations pour prouver la terminaison et correction du programme.

2.3 Racine carrée

Question 6. Ouvrez le fichier "sqrt.mlw", complétez les preconditions et postcondidions de la fonction.

Question 7. Complétez les annotations pour prouver la terminaison et correction du programme.

3 Listes

3.1 Tri booléen

Question 8. Ouvrez le fichier "two_way.mlw", complétez les preconditions et postcondidions de la fonction.

Question 9. Prouvez la correction puis la terminaison de la fonction en complétant l'invariant et le variant.

3.2 Recherche dichotomique

Question 10. Ouvrez le fichier "binary_search.mlw", complétez les preconditions et postcondidions de la fonction.

Question 11. Prouvez la correction puis la terminaison de la fonction en complétant l'invariant et le variant.

3.3 Tri par sélection

Question 12. Ouvrez le fichier "selection_sort.mlw", complétez les preconditions et postcondidions de la fonction.

Question 13. Prouvez la correction puis la terminaison de la fonction en complétant l'invariant et le variant.

3.4 Tri bulle

Question 14. Ouvrez le fichier "bubble_sort.mlw", complétez les preconditions et postcondidions de la fonction.

Question 15. Prouvez la correction puis la terminaison de la fonction en complétant l'invariant et le variant.