INFO0947 : TAD

Groupe 02 : Simon Lorent, Corentin Jemine

Avril/Mai 2015

1 Introduction

Dans ce troisième projet nous avons définit deux structures de données, List et Array, pour lesquelles nous avons implémenté des fonctions de base. Ces fonctions sont des constructeurs, des observateurs ou des transformateurs. Nous nous pencherons sur une définition théorique de ces structures ainsi que leurs avantages et inconvénients respectifs.

2 Définition du type abstrait

2.1 Signature

Type :
Multi [1]
Utilise :
Natural, Boolean, Element [2]
Opérations :
create_empty : → Multi
is_empty : Multi → Boolean
count : Multi → Natural
occurrences : Element x Multi → Natural
part_of : Element x Multi → Boolean
equals : Multi x Multi → Boolean
join : Multi x Multi → Multi
add_to : Element x Multi → Multi
remove_from : Element x Multi → Multi

2.2 Sémantique

Préconditions :
Aucune [3]
Axiomes :
Notations : #m désigne le nombre d'Elements dans m
m[i] désigne le i-ème Element de m
Remarque : le signe d'égalité entre 2 Multis suit la définition de la fonction equals()

∀ m, m' ∈ Multi, ∀ e ∈ Element :

is_empty(create_empty()) = True

count(m) = #m

$$\text{occurrences(e, m)} = \sum_{i=1}^{count(m)} (\text{m[i]} == \text{e})$$

part_of(e, m) = (occurrences(e, m) > 0)

$$\text{equals(m, m')} = (\text{count(m)} == \text{count(m')}) \;\&\& \prod_{i=1}^{count(m)} ((\text{occurrences(m[i], m)} == \text{occurrences(m[i], m')})$$

1. Multi désigne soit le type List, soit le type Array
2. Element désigne une type générique
3. remove_from est définit sur un Multi vide mais n'aura aucun effet

occurrences(e, join(m, m')) = occurrences(e, m) + occurrences(e, m')

add_to(e, m) = join(m, add_to(e, create_empty()))

count(add_to(e, m)) = count(m) + 1

is_empty(add_to(e, m)) = False

Si part_of(e, m) **alors** m = remove_from(e, add_to(e, m))

Si ¬ part_of(e, m) **alors** equals(m, remove_from(e, m))

2.3 Jusitification des axiomes