LINFO1122 - Méthodes de Conception de Programmes
Devoir 2 – Structures de données

Charles Pecheur

Automne 2022

L'objectif de ce devoir est de définir et spécifier des *tables de hachage*. Une table de hachage (hash table) est une structure de données qui permet de représenter une association clés-valeurs, avec un accès rapide à une clé donnée au moyen d'une *fonction de hachage* (hash function). Le hachage de la clé donne l'index de l'élément correspondant dans la table de hachage. Pour plus d'informations, consultez le livre de référence du cours d'Algorithmes et structures de données, chapitre 3.4, [1] et les pages Wikipédia. [2]

Vous devez construire une représentation d'ensemble de chaînes de caractères sous forme de table de hachage.(le type des éléments n'est pas important ; les `string` sont pris pour réduire le problème). Les collisions seront traitées par chaînage, chaque entrée de la table étant une liste chaînée d'éléments. Vous vous baserez sur les types de données suivants :

```
type HashSet = array<List>

datatype List = Nil | Cons(string, List)
```

- La fonction de hachage `hash`() sur les `string` (pour une taille de table donnée) ;
- L'invariant de représentation `ok`() sur `HashSet` ;
- La fonction d'abstraction `abs`() sur `HashSet` ;
- la spécification formelle et complète et l'implémentation des procédures suivantes :
  - `create(n: int) returns (a: HashSet)` qui crée une table de taille `n`, initialement vide ;
  - `isin(s: string, a: HashSet) returns (result: bool)` qui teste si `s` est présent dans le tableau ;
  - `add(a: HashSet, s: string)` qui ajoute le string `s` dans le tableau ;
  - `remove(a: HashSet, s: string)` qui supprime le string `s` dans le tableau ;
  - ...et toutes les procédures auxiliaires définies pour celles-ci ;
- la preuve de correction de votre implémentation de `isin`.

Vous devrez nécessairement définir aussi une fonction de hachage, mais seule sa spécification importe : l'implémentation d'une fonction de hachage sur les `string` pertinente et efficace ne fait pas partie de l'exercice ; une implémentation triviale suffira tant qu'elle satisfasse aux spécifications. Votre solution doit contenir les éléments suivants :

La définition de `ok` et `abs` peut nécessiter d'introduire des fonctions et/ou prédicats auxiliaires. Si vos procédures sur les listes sont définies par induction sur la structure, la terminaison est assurée et la preuve ne nécessite pas d'invariants ni de variants.

Vous pouvez écrire votre implémentation dans votre environnement Dafny, mais il n'est **pas** nécessaire, et même déconseillé, de tenter de valider les spécifications et de faire les preuves dans Dafny. Pour ce devoir, ce n'est pas nécessaire, et cela vous ferait perdre beaucoup de temps. Préférez pour ces parties un éditeur de texte adapté, ou une copie manuscrite comme pour le devoir 1.

Pour pouvoir tester votre code dans Dafny, vous pouvez définir une fonction `Main` et exécuter votre programme (dans VS-Code, clic droit et "Dafny : Compile and Run (F5)"). Par exemple :

1. Sedgewick, R., & Wayne, K. (2007). Algorithms and data structures. Princeton University, COS. `https://algs4.cs.princeton.edu/32bst`
2. `https://fr.wikipedia.org/wiki/Table_de_hachage`, `https://en.wikipedia.org/wiki/Hash_table`

```
method Main() {
    var ht := create(10) ;
    add(ht, "coucou") ;
    print ht ;
}
```

Enfin, vous aurez certainement besoin de l'instruction `match` en Dafny. Pensez à consulter les ressources disponibles sur Moodle.

Les réponses doivent être soumises sous forme d'une archive zip, contenant un document PDF reprenant vos solutions, et un fichier source avec le code de votre programme. Ce travail est à rendre sur Moodle pour le **vendredi 25 novembre 2022**.