TP5 - Le TP0 en Isabelle/HOL et Scala

L'objectif de ce TP est de re-programmer en Isabelle la fonction du TP0 : une fonction `subSeq` qui détermine si une liste `la` est une sous-liste de `lb` en autorisant qu'au plus un élément de `la` n'apparaîsse pas dans `lb`.

1 Formalisation et vérification de subSeq en Isabelle

1. Complétez le fichier `/share/m1info/ACF/TP5/tp5.thy` avec la fonction `subSeq` en Isabelle/HOL ;
2. Définissez le(s) lemmes définissant la propriété à prouver sur `subSeq` ;
3. Vérifier vos lemmes à l'aide des générateurs de contre-exemple (`nitpick` et `quickcheck [tester=narrowing]`) en augmentant le `timeout` et la taille des contre-exemples recherchés.
4. Quand vos lemmes résistent à une génération de contre-exemple poussée, vous pouvez passer à la partie suivante.

Remarque : Pour définir la propriété, il est possible de programmer au préalable une fonction plus simple `exactSubSeq` qui détermine si une liste `la` est (exactement) une sous-liste de `lb`. Une façon de formaliser la propriété attendue sur `subSeq` est d'utiliser `exactSubSeq` et l'opérateur `@`. L'opérateur `@` de concaténation de listes permet, par exemple, de dire qu'il existe une liste `l3` identique à la liste `la` moins un élément en utilisant la formalisation suivante : `∃x l1 l2 l3. la=l1@[x]@l2 ∧ l3=l1@l2`. **Inspirez-vous** de cette forme pour formaliser la propriété attendue sur `subSeq`. Pour programmer `subSeq` il n'est pas nécessaire d'utiliser `exactSubSeq`.

2 Exportation et intégration du code Isabelle vérifié

Dans Eclipse Scala, importer le projet se trouvant dans l'archive `/share/m1info/ACF/TP5/TP5ACF.zip` :

File>Import>General>Existing Projects into Workspace>Select archive file

2.1 Exportation du code Scala depuis Isabelle/HOL

Afin de produire le code Scala, utilisez la directive `export_code` dans votre théorie après la définition de `subSeq` :

```
export_code subSeq in Scala
```

Ceci génère le code directement dans Isabelle/HOL. Pour générer le code dans un fichier `tp5.scala` la directive est :

```
export_code subSeq in Scala
   file "tp5.scala"
```

Remarque : le fichier `tp5.scala` est généré dans le repertoire depuis lequel Isabelle/HOL a été lancée.

2.2 Intégration du code Scala généré dans le projet Eclipse TP5

Le fichier Scala généré doit débuter par `package TP5`. Ajoutez-le au projet Eclipse. L'objectif est, ensuite, de compléter la classe `Sequence` du projet `TP5` ainsi que le contenu de la méthode `subSeq` de cette classe : `Sequence` :

```
def subSeq(t1: Array[Object], t2: Array[Object]): Boolean = {
    // Conversion des tableaux Java en listes Scala
    val sl1= t1.toList
    val sl2= t2.toList

    /* TODO */
    true
}
```

Pour l'instant cette méthode rend toujours `true`. A la place de `true` vous devez appeler la fonction Scala générée à partir du code Isabelle sur les listes `sl1` et `sl2` qui sont les listes Scala correspondant aux tableaux Java `t1` et `t2`. De plus, la fonction Isabelle `subSeq` travaille sur des listes polymorphes d'éléments de type `'a` pour lesquels l'égalité est définie. La fonction `subSeq` **générée** en Scala travaille sur des listes polymorphes d'éléments de type `A` ayant le trait `HOL.equal` :

```
def subSeq[A : HOL.equal](la: List[A], lb: List[A]): Boolean
```

Depuis Scala, on ne pourra appeler cette fonction qu'avec des listes d'objets de type `T` ayant le trait `HOL.equal`. Ce trait garantit simplement que la méthode `equal` testant l'égalité entre deux objets de type `T` est définie. La façon la plus simple de définir automatiquement ce trait pour tout type `T` est d'ajouter la définition *implicite* suivante :

```
implicit def equal_t[T]: HOL.equal[T] = new HOL.equal[T] {
    val `HOL.equal` = (a: T, b: T) => a==b
}
```

Celle-ci définit la fonction de conversion implicite `equal_t[T]` qui traduit à la volée tout type `T` en un type ayant le trait `HOL.equal[T]`. Pour l'implantation de la fonction `equal` de ce trait, on choisit simplement l'égalité Scala '`==`'.

2.3 Tests

Pour tester votre fonction `subSeq` lancez l'application Java `Main.java` qui est identique à l'oracle de test que vous avez utilisé au TP0.