TP5bis - Le simplificateur de "Glob patterns", acte II

(Ne commencer le TP5bis que si le TP5 a été rendu)

Fichiers du projet : `TP5bis_ACF.zip`, `tp5bis.thy`.

Le TP est à rendre **sur le MOODLE du cours** au plus tard le *Mardi 3/11 à 14h00*.
Les TPs devant être mis en ligne pour être évalués en même temps, **aucun retard ne sera possible**.
Déposez sur MOODLE une archive au format **ZIP** dont le nom sera de la forme **Nom1_Nom2_TP5bis_ACF.zip** et contenant :

1. **un JAR** de votre projet `TP5bis` au format `Nom1_Nom2_TP5bis.jar`. Pour générer le `.jar`, dans le shell SBT, tapez `package`. Le `.jar` doit être généré le répertoire `target/scala-2.13` du projet. **Renommez** le `.jar` avec le nom de votre binôme `Nom1_Nom2_TP5bis.jar`. Remarque : le `.jar` n'a pas besoin d'être exécutable.
2. **le répertoire de projet TP5bis complet**.

   Remarque : Tout **votre** code Scala devra être contenu dans le fichier `Simplify.scala`.
3. **la théorie Isabelle** complétée : `tp5bis.thy`.

---

1 Préliminaires

Décompressez l'archive `/share/m1info/ACF/TP5/TP5bis_ACF.zip` dans votre répertoire ACF. Dans votre éditeur Scala, ouvrez le projet `TP5bis_ACF`. Dans Isabelle/HOL, ouvrez le fichier `tp5bis.thy`.

2 Objectif

L'objectif de ce TP est le même que celui du TP précédent (TP5) mais on va exploiter Isabelle/HOL pour obtenir un simplificateur correct et générer le code Scala correspondant.

3 Marche à suivre

1. Comme dans le TP précédent, dans `TP5bis`, vous devez remplacer la chaîne `LE_NOM_DE_VOTRE_BINOME` par le nom de votre binôme à plusieurs endroits : dans les noms de répertoires et dans quelques fichiers sources. Certains éditeurs permettent de remplacer un nom de package rapidement ("refactoring").
2. Dans `tp5bis.thy`, vous trouverez la définition des patterns/globs ainsi que la fonction `accept` qui dit quand un mot est accepté par un pattern. Elle vous servira plus tard pour écrire le lemme de correction.
3. Définissez la fonction `simplify` de simplification des patterns.
4. Écrivez le lemme de correction qui assure qu'un pattern simplifié accepte les mêmes mots que le pattern original en vous servant de la fonction `accept`.

5. Utilisez les générateurs de contre-exemples sur le lemme de correction pour trouver des erreurs éventuelles dans votre fonction `simplify` (ou dans votre lemme).
6. Utilisez les générateurs de contre-exemples sur le lemme de **minimalité** pour voir si votre fonction `simplify` retourne bien le pattern simplifié minimal.
7. (Bonus) Prouvez les lemmes intermédiaires et le lemme de correction.
8. Générez le code Scala de la fonction `simplify` et intégrez-le dans le fichier `Simplify.scala`. La classe `Simplify` doit implanter le trait `Simplifier` et la méthode `simplify`. En particulier, **il est interdit de changer le nom de cette méthode dans le code Scala** car c'est cette méthode qui sera appelée par le serveur sur lequel seront déployés tous vos TPs.
9. Testez votre implantation en exécutant l'objet exécutable `Application` qui lit un glob et donne la version simplifiée avec votre implantation du `Simplifier`.