TP4 Bis - Décider l'équivalence d'expressions arithmétiques simples

Fichiers de TP : `tp4bis.thy`, `table.thy`

Placez le fichier `tp4bis.thy` et le fichier `table.thy` dans le même répertoire. Ouvrez ces deux fichiers dans Isabelle/HOL.

1 Objectif du TP

Dans ce TP, vous allez construire un algorithme permettant de décider si deux expressions arithmétiques sont équivalentes. Les expressions que l'on considère contiennent des constantes, des variables et les opérateurs binaires '+' et '-'.

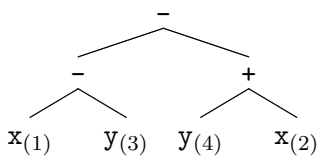
Deux expressions sont équivalentes, si pour toutes valeurs des variables, elles s'évaluent à la même valeur entière. Par exemple, l'expression `(0 + (x + 0)) + y` est équivalente à l'expression `y + x`. L'expression `(x - y) + (y - x)` est équivalente à l'expression `0`. A l'inverse l'expression `x + y` n'est pas équivalente à l'expression `y` car il existe des valeurs de `x` et de `y` telles que `x + y` et `y` s'évaluent de façon différente.

Dans la théorie `tp4bis`, vous trouverez le type algébrique `expression` ainsi qu'une fonction `evalE` :: `expression` ⇒ `symTable` ⇒ `int` permettant d'évaluer une expression à l'aide d'une table associant des variables à des valeurs entières. La fonction `evalE` n'est, a priori, pas utile pour décider l'équivalence d'expressions mais vous servira de base de confiance lors de l'écriture des lemmes de correction.

2 Marche à suivre pour décider l'équivalence d'expressions

Pour décider de l'équivalence, on propose de compiler les expressions sous la forme `compiledExp` : une paire composée d'un entier (la somme des constantes de l'expression, en tenant compte des opérateurs '`+`' et '`-`' rencontrés en la parcourant) et d'une table associant une variable à un entier (la multiplicité de chaque variable dans une expression : la différence entre les occurrences positives et négatives de l'expression). En effet, deux expressions sont équivalentes si on a les deux propriétés suivantes :

1. Les sommes des constantes (tenant compte des '`+`' et '`-`') des expressions sont identiques. Par exemple, la somme des constantes dans l'expression `(x + 1) - (y + 1)` est `0`, la somme des constantes dans l'expression `(x - 1) - (y - 1)` est également `0`.
2. La multiplicité de chaque variable concorde entre les deux expressions. Par exemple, dans l'expression `(x - y) - (y + x)` dont la représentation est l'arbre



l'occurrence de `x` à la position (1) est positive, alors que l'occurrence à la position (2) est négative. La multiplicité de `x` est donc 0. Ainsi, quelle que soit la valeur de la variable `x`, la somme de ces valeurs s'annulera lors de l'évaluation de l'expression. L'occurrence de `y` à la position (3) est négative et il en est de même pour l'occurrence (4). La multiplicité de `y` est donc de -2. Ainsi, pour toute valeur de la variable `y`, cette valeur sera multipliée par `-2` lors de l'évaluation de l'expression.

Ainsi, l'expression `(x - y) - (y + x)` sera équivalente aux expressions `0 - (y + y)`, `(z - y) - (y + z)`, `(1 - y) - (1 - (0 - y))`, ...car elles se compilent vers la même `compiledExp`.

1. Définissez la fonction de compilation des expressions `compile` :: `expression` ⇒ `compiledExp` ;
2. Définissez le prédicat d'équivalence d'expressions `equiv` :: `expression` ⇒ `expression` ⇒ `bool`.

3 Lemmes de corrections

Voici quelques propriétés attendues, à vous de les formaliser avec des lemmes Isabelle/HOL pour découvrir des erreurs éventuelles dans votre code.

— Lemme de correction de la compilation d'une expression : "Pour toute expression et toute valeur des variables, l'évaluation de l'expression et l'évaluation de l'expression compilée concordent". Pour écrire ce lemme, il vous sera nécessaire de définir la fonction d'évaluation des expressions compilées `evalCompiled` :: `compiledExp` ⇒ `symTable` ⇒ `int` ;
— Lemme de correction de `equiv` : "Si deux expressions satisfont `equiv` alors les résultats de leur évaluations (par `evalE`) concordent pour toutes valeurs des variables" ;
— Lemme de complétude de `equiv` : "Pour deux expressions, si les résultats de leur évaluations (par `evalE`) concordent pour toutes valeurs des variables alors elles doivent satisfaire `equiv`".

Remarque : sur vos propriétés, cherchez des contre-exemples en poussant les temps de calcul des générateurs de contre-exemples. Les preuves ne sont pas demandées donc **ne passez à la preuve que s'il vous reste du temps en fin de TP**.