Rapport - Compilation : Petit Scala, première partie

Nathanaël Courant, Noémie Cartier

5 décembre 2015

1 Choix techniques

1.1 Analyseur lexical

Nous avons choisi, pour des raisons de performance et de minimisation du nombre d'états de l'automate de l'analyseur lexical, de stocker les mots-clef et le lexème correspondant dans une table de hachage. En revanche, comme il est possible de placer plusieurs opérateurs à la suite sans séparateur (mais que tous les opérateurs ne sont pas réduits à un signe) nous avons réalisé leur reconnaissance en écrivant un cas par opérateur.

1.2 Analyseur syntaxique

Nous avons beaucoup utilisé la bibliothèque de Menhir, en particulier les fonctions `option`, `delimited`, `list` et `separated_list`. La fonction `ioption` nous a permis de résoudre un conflit.

Les constantes valant précisément -2^{31} nous ont posé des problèmes : en effet, elles ne peuvent pas être reconnues immédiatement par l'analyseur lexical, puisque celui-ci n'est pas capable d'identifier si le symbole - devant la constante est celui de la constante, ou est un opérateur binaire. De plus, nous n'avons pas réussi à écrire une grammaire qui permettrait de détecter ces constantes négatives lors de l'analyse syntaxique. Ce que nous avons dont fait a été de rajouter une phase de traitement après l'analyse syntaxique, où les - unaires précédant un entier sont transformés en constantes entières négatives, et où les débordements de constantes sont détectés.

1.3 Typeur

Dans la définition d'un type (visible au début du fichier `type_ast.ml`), nous avons créé trois constructeurs différents afin de mieux gérer la portée des différentes variables de type.

De la même façon, nous avons choisi d'utiliser une structure de type `Map` au lieu de `Hashtbl` pour avoir des environnements persistants.

Dans le calcul de la variance, nous avons dû prendre un double paramètre en permanence : la classe (ou le type, selon les arguments nécessaires pour chaque fonction) créée par le typeur et celle fournie par l'analyseur syntaxique, les premiers portant les information de typage qui a déjà été fait (portée des variables de type, type des champs lorsque celui-ci n'est pas précisé en particulier), et les deuxième portant la localisation à afficher en cas d'erreur.

2 Tests supplémentaires

Nous nous sommes que, bien que notre typeur passe déjà tous les tests fournis pour le projet, il ne se comportait pas comme il aurait dû dans certains cas particuliers. Nous avons donc créé quelques nouveaux tests (ceux dont le nom est de la forme `extra*.scala`).

Ces tests supplémentaires vérifient ces cas particuliers :

— Vérification du fait que la constante -2^{31} est acceptée
— Vérification de la variance dans un champ d'une classe dont le type n'est pas précisé dans le programme
— Vérification de la bonne gestion des variables de type lorsque leur nom est commun à un paramètre de méthode et à un paramètre de classe, ou à une classe définie précédemment
— Vérification de la bonne gestion de variables de type nommées comme une classe standard (`Int`, `Unit`, ...)