Rapport - Compilation : Petit Scala, deuxième partie

Nathanaël Courant, Noémie Cartier

11 janvier 2016

Choix techniques

Production de code

Nous avons choisi de compiler les expressions en plaçant leur résultat dans `%rax` : ainsi, `compile_expr` prend en arguments une expression, la représentation des classes en mémoire (voir plus loin), et le nombre d'arguments de la fonction actuelle, et compile cette expression en écrivant son résultat dans `%rax`.

Les expressions égales à `()` ne remettent pas `%rax` à 0, la valeur de `()` n'étant jamais utilisée. En effet, les règles de typage interdisent toute utilisation de la valeur d'un objet de type `Unit`, celui-ci ne pouvant ni être converti en une valeur d'un autre type, ni testé pour son égalité (avec un autre objet de type `Unit`).

Le valeur de retour de `compile_expr` est une paire de type `X86_64.text * X86_64.data`, représentant l'assembleur à écrire dans le segment de texte et de données, respectivement.

L'appel d'une fonction se fait en passant tous les paramètres de celle-ci sur la pile, en poussant le premier paramètre (l'objet lui-même) en premier.

L'appel du constructeur d'une classe se déroule en commençant par créer l'objet (i.e. l'allouer et remplir l'entête avec l'adresse du descripteur de classe), puis écrire les paramètres du constructeur à leur place respective dans l'objet, là où ils sont stockés plus tard. On termine par appeler le constructeur comme une autre fonction. Lorsque la classe a une classe parente, le constructeur calcule donc les paramètres qui sont passés au parent et les stocke dans l'objet, puis appelle le constructeur de la classe parente, et enfin calcule les nouveaux champs qui sont définis dans cette classe.

Ainsi, si classe B étend la classe A, la représentation mémoire de B est la suivante :

descripteur de B
paramètres du constructeur de A
champs définis dans A
paramètres du constructeur de B
champs définis dans B

Notre code initial évaluait les expressions binaires arithmétiques de droite à gauche, pour simplifier le code produit. Cependant, nous n'avons pas pu trouver d'informations sur l'ordre d'évaluation des paramètres des opérateurs binaires en Scala, et les tests effectués avec `scalac` semblaient indiquer que celui-ci était de gauche à droite ; nous avons donc changé cela dans notre compilateur.

Finalement, le calcul de la représentation mémoire des objets et des descripteurs de classes se fait au tout début, dans l'ordre de définition des classes.

La fonction gérant la production de code n'a pas besoin de faire de l'analyse de portée ou de détermination des méthodes appelées, ceci étant déjà fait au cours du typage.