Projet de compilation : mini-langage orienté objets
Partie 2: Typage et évaluation

Ce document décrit la seconde partie du projet d'analyse syntaxique et compilation. L'objet de cette partie du projet est de réaliser un analyseur statique de bon typage et un évaluateur pour le mini-langage à objets décrit en cours.

Le document de cours décrit la grande majorité des éléments nécessaires à la réalisation de cette partie du projet. Nous détaillons uniquement les différences ajoutées au langage par commodité.

1 Détails pratiques

1.1 Remarques générales

Votre programme devra gérer correctement tous les cas d'erreurs: vous devez utiliser dans la mesure du possible toutes les erreurs de typage et d'évaluation définies dans le fichier `utils.ml`, à l'aide des fonctions `raise_typing_error` et `raise_eval_error`. Les erreurs qui y sont définies devraient vous donner une bonne indication de tous les cas que vous devez gérer.

1.2 Typage

Le typage du mini-langage à objets a été vu en cours et est détaillé à la section 7.3 des notes de cours. La commande `print "chaine"` est toujours bien typée, et l'expression `fail` a tous les types. Pour simplifier, on pourra supposer que `fail` a un type $\bot$, et utiliser la règle de sous-typage supplémentaire $\bot \leq C$ pour tout C. On a ainsi, pour toute classe C:

$$\bot \leq C \leq \texttt{Object}$$

1.3 Évaluation

L'évaluation du mini-langage à objets à été vue en cours et est détaillée à la section 7.2 des notes de cours. La commande `print "chaine"` devra afficher sur la sortie standard `chaine`, et l'évaluation de l'expression `fail` devra générer l'erreur d'évaluation `Abort`.

2 Consignes

1. Recopiez sur votre compte le répertoire:
   `/info/ens/meyera/Public/Compil/projet/squelette/partie2`.
   Dans celui-ci, vous trouverez un squelette du code de la partie 2, dans lequel vous insérerez vos fichiers `parser.mly` et `lexer.mll` modifiés dans la partie 1. Vous trouverez également un fichier `utils.ml` contenant un ensemble de fonctions auxiliaires pour la manipulation de l'environnement, de la mémoire et des erreurs.
2. Complétez les fichiers `typage.ml` et `eval.ml` en suivant les règles de typage et d'évaluation fournies dans le polycopié du cours ainsi que les règles ci-dessus, afin de créer un analyseur de type et un évaluateur pour le langage. En principe, il n'est pas nécessaire de modifier les autres fichiers fournis. **Attention :** l'afficheur (fichier `print.ml`) et le programme principal (`main.ml`) ont été mis à jour depuis la partie 1.
3. Expérimentez votre analyseur syntaxique / typeur / évaluateur sur des exemples de programmes, comme la classe des entiers naturels en unaire programmée en partie 1 ou tout autre programme que vous aurez réalisé.