Interprétation des programmes – TP 3 : Données d'ordre supérieur

Université Paris Diderot – Master 1

(2014-2015)

Nous avons introduit un nouveau langage en cours. Il s'agit de HOPIX, un langage qui étend DATIX avec des fonctions de première classe.

Cette séance de travaux pratiques a pour objectifs :
- de continuer votre apprentissage de l'écriture d'un interprète à travers l'écriture de celui de HOPIX ;
- de vous faire lire et étendre le code source d'un vérificateur de types simples ;
- de vous faire concevoir une passe de compilation qui traduit l'ordre supérieur au premier ordre.

Le code source correspondant à ces travaux pratiques se trouve sur le GIT, dont on rappelle l'URL :

`http://moule.informatique.univ-paris-diderot.fr:8080/Yann/compilation-m1`

On rappelle que vous devez faire des *commits* réguliers (à chaque modification de votre code) pour que nous puissions suivre votre avancement.

Exercice 1 *HOPIX est une extension de DATIX*

1. *Quelles différences notez-vous entre HOPIX et DATIX ?*
2. *Dans l'interprète de HOPIX, pourquoi les valeurs sont-elles représentées par un type paramétré ?*
3. *Dans l'interprète de HOPIX, quels cas de l'interprète de DATIX sont réutilisables ?*

□

Exercice 2 *Vérificateur de types de HOPIX*

1. *Pourquoi n'y-a-t-il plus de notion de signature pour typer les fonctions de HOPIX ?*
2. *Comment le vérificateur de type doit-il traiter le cas où le type d'un argument d'une fonction est manquant ?*
3. *Complétez le vérificateur de type de HOPIX.*

□

Exercice 3 *Les fonctions anonymes de HOPIX*

Dans un premier temps, on ignore les fonctions récursives, elles seront l'objet de l'exercice suivant.

1. *Dans l'interprète de HOPIX, complétez les cas correspondant aux constructions de DATIX, puis ceux des fonctions anonymes et des applications.*
2. *Complétez la fonction* `HopixAST.free_variables`*. À quoi servira-t-elle d'après vous ?*
3. *Dans le compilateur de HOPIX vers DATIX, complétez les fonctions* `closure_conversion` *et* `hoist` *pour qu'elles traitent l'intersection des langages HOPIX et DATIX.*
4. *Dans le compilateur de HOPIX vers DATIX, complétez la fonction* `closure_conversion` *pour qu'elle traite les fonctions anonymes et les applications. Vous pouvez tester votre fonction en produisant un programme HOPIX intermédaire et en vérifiant qu'il s'exécute comme le programme source.*
5. *Dans DATIX et FOPIX, rajoutez les pointeurs sur fonctions. Pour cela, il faudra rajouter une nouvelle construction dans l'AST et étendre l'ensemble des fonctions sur ce type pour traiter ce nouveau cas.*
6. *Dans le compilateur de HOPIX vers DATIX, étendre la fonction* `hoist` *pour qu'elle prenne en compte les fonctions anonymes et les applications.*

□