

# Interprétation des programmes – TP 3 :

## De Hopix à Hobix

Université Paris Diderot – Master 1

(2016-2017)

Cette séance de travaux pratiques a pour objectifs de :

- vous faire écrire votre première passe de compilation ;
- vous faire comprendre et traduire la sémantique du *pattern matching* ;
- vous faire expliciter les représentations en machine des types de données de types sommes.

Remarque : Si vous n'avez pas terminé le TP3, vous pouvez désactiver la vérification des types de HOPIX à l'aide de l'option `--typechecking false` de flap.

### Exercice 1 (Preliminaires)

1. Quelles sont les différences syntaxiques entre HOPIX et HOBIX ?
2. Pourquoi n'est-il pas nécessaire de typer HOBIX ?
3. Qu'apporterait néanmoins le typage de HOBIX en termes de génie logiciel ?
4. Quelle est la sémantique des opérations de HOBIX qui ne sont pas dans HOPIX ?
5. Comment peut-on tester une passe de compilation ?
6. Lisez le fichier `HopixToHobix` et compléter la fonction `HopixToHobix.component`.

□

### Exercice 2 (Traduction des types sommes)

1. Comment est représentée une valeur construite (i.e un ensemble de valeurs muni d'une étiquette) ?
2. Écrire sur papier le code compilé HOBIX correspondant au programme suivant :

```
type l = { N | C : int * l }.
val n = N.
val a = C (1, C (2, N)).
val b = C (3, a).
val zero = a ? { N => 0 | _ => 1 }.
val one = a ? { N => 0 | C (i, _) => i }.
val two = a ? { N | C (_, _) => 0 | C (_, C (i, _)) => i }.
val three = b ? { C (i, _) & C (_, C (_, c)) => C (i, c) | _ => N }.
```

3. Compléter le cas de la déclaration des types sommes dans la fonction `HopixToHobix.type_definition`.
4. Compléter le cas de construction d'une valeur d'un type somme dans la fonction `HopixToHobix.expression`.
5. Compléter la fonction `HopixToHobix.pattern` en supposant que le motif ne contient pas de POr.
6. Utiliser la fonction de la question précédente pour compléter le cas du *pattern matching* dans la fonction `HopixToHobix.expression`.
7. Après avoir lu le fichier `utilities/ListMonad.mli`, compléter la fonction `HopixToHobix.expand_or_patterns_in_branch`. Utiliser cette fonction pour faire une passe sur les branches qui garantit qu'il n'y a pas de POr dans les motifs.

□