Compilation – TP 1 : OCamlLex et Menhir

Université Paris Diderot – Master 1

(2021–2022)

Les objectifs de cette séance de travaux pratiques sont :

- de comprendre les rôles respectifs de l'analyse lexicale et grammaticale dans l'analyse syntaxique ;
- de prendre connaissance des formats d'entrée des outils OCamllex et Menhir ;
- de comprendre la sémantique des spécifications OCamllex ;
- de comprendre la sémantique des spécifications Menhir ;
- de résoudre un conflit d'analyse grammaticale en spécifiant des priorités ;
- de résoudre un conflit d'analyse grammaticale en réécrivant une grammaire.

Les fichiers sources utilisés pour ces travaux pratiques sont sur le dépôt GIT à l'emplacement `tp/tp-menhir/code`. Pour réaliser ces travaux pratiques, votre environnement de travail doit inclure :

— `dune >= 1.6`
— `menhir >= 20181113`
— `ocaml >= 4.10.0`

Si vous travaillez depuis les machines de l'UFR, faites depuis la racine de votre compte.

```
ln -s /ens/guatto/.opam
```

La commande `eval $(opam config env)` dans votre shell courant devrait configurer correctement votre environnement de travail. Rajoutez cette ligne à la fin de votre `.bashrc`.

Exercice 1 (Prise en main du code source Marthe)

1. *Quel est le rôle de chaque fichier source fourni ?*
2. *Compilez le code source.*
3. *Expliquez les avertissements produits par la compilation.*
4. *Après avoir lu* `marthe.ml`*, expliquez ce que produise les entrées suivantes :*
   — `37`
   — `1 + 2`
   — `1 +`
   — `1 + 2 + 3`

□

Exercice 2 (Compléter l'analyseur lexical)

1. *Rajoutez une règle d'analyse lexicale pour reconnaître le caractère* `*` *comme le lexème* `STAR`*. Même chose pour les parenthèses gauches et droites.*
2. *Rajoutez une règle d'analyse lexicale pour reconnaître le mot-clé* `sum`*. Comment s'assurer que la chaîne* `sum` *est bien reconnu comme ce mot-clé et non comme un identificateur ?*

□

Exercice 3 (Compléter l'analyseur grammatical)

1. *Enlevez* `%left PLUS` *puis recompilez. Quel est l'effet de ce changement ?*

2. *Remplacez* `%left PLUS` *par* `%right PLUS`. *Quel est l'effet de ce changement ?*
3. *Introduisez une règle pour reconnaître une multiplication. En étudiant* `_build/target/parser.conflicts`, *expliquez pourquoi cette introduction provoque un conflit "shift/reduce".*
4. *Après avoir rappelé les règles de spécification de priorité en Menhir, résoudre le conflit introduit par la question précédente. Comment savoir si vous avez correctement résolu le conflit ?*
5. *Complétez la grammaire pour reconnaître les expressions parenthésées.*
6. *Complétez la grammaire pour reconnaître les expressions de la forme* `sum (x, start, stop, body)`. *Pourquoi cette nouvelle règle ne rentre-t-elle pas en conflit avec la règle introduite par la question précédente alors qu'elles partagent des lexèmes ?*

□

Exercice 4 (Pour les plus rapides)

1. *Dans le premier cours, la syntaxe de Marthe était stratifiée à l'aide de plusieurs non terminaux (factor, term, expression). Modifiez votre grammaire Menhir pour suivre cette grammaire stratifiée. Pourquoi cette stratification fait-elle disparaître les conflits ? Est-ce une grammaire équivalente à la grammaire précédente ?*
2. *Modifiez la fonction* `eval` *pour qu'elle calcule l'entier correspondant à l'expression Marthe.*
3. *Comment testez l'expression de l'utilisateur avant de l'évaluer pour être sûr que son évaluation ne va pas échouée ? Ecrivez une fonction* `check` *qui réalise ce test et produit un message d'erreur explicatif si une erreur potentielle peut se produire. Comment se comporte votre fonction sur les entrées suivantes :*
   — `x`
   — `sum (x, 0, 10, y)`
   — `sum (x, 0, -10, y)`
   ?

□