Compilateur purscript : rendu 1

6 décembre 2023

Le lexer

Le lexer est codé dans le fichier purscript_lexer.mll et est compatible avec ocamllex.

La plupart des lexems sont reconnus directement par la règle principale. Il y a quelques cas où l'on utilise une règle différente :

— les commentaires sur plusieurs lignes
— les chaînes de caracteres
— les chaînes de caracteres entre les symboles "\\"

Le parser

Il est codé dans le fichier purscrip_parser.mly et est compatible avec menhir.

Il reprend globalement la grammaire proposée pour le purscript. Mais doit gérer quelques cas délicats.

Le premier problème vient de la règle :

$$\langle \text{tdecl} \rangle ::= \langle \text{lident} \rangle :: (\langle \text{ntype} \rangle =>)^* \ (\langle \text{type} \rangle - >)^* \ \langle \text{type} \rangle$$

Dans ce cas, il n'est pas possible de juste reconnaître grâce à la règle suivante :

$$\text{list}(\text{ntype}, =>) \ \text{list}(\text{type}, - >) \ \text{type}$$

En effet, cela provoque un conflit car le parser ne sait pas quand passer d'une liste à l'autre.

La solution est donc de remplacer par 3 régles qui s'appellent en chaîne et qui permettent de placer nous-même les limites.

Le deuxième problème vient de deux dérivations possibles dans la grammaire. En effet, si l'on souhaite reconnaitre on objet de 'type' et que l'on lit un 'uident', alors les deux dérivations suivantes sont possibles :

$$\left[\begin{array}{l} \text{type} \rightarrow \text{atype} \rightarrow \text{uident} \\ \text{type} \rightarrow \text{ntype} \rightarrow \text{uident} \end{array}\right.$$

Mais pour la suite, ces deux dérivations sont équivalentes, donc nous avons forcé le parser à reconnaitre l'un des deux (en l'occurence le premier).