

Compilateur purscript : rendu 1

9 décembre 2023

Le lexer

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

La plupart des lexemes 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 caractères
- les chaînes de caractères 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 reconnaître $\langle \text{tdecl} \rangle$ avec 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 règles 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 reconnaître un 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 reconnaître l'un des deux (en l'occurrence le premier).

Le typage : point de vue général

Bonjour

Makefile / dune / tests

Le projet contient également des scripts permettant d'automatiser la compilation et de tester rapidement sur de nombreux tests.

La plupart des tests sont fournis mais nous avons rajouté quelques tests afin de vérifier le comportement sur quelques cas particuliers.

Dune s'occupe de produire le fichier `purscript_main.exe`. Et Make s'occupe de lancer dune, de renommer l'exécutable, et de lancer les tests.