CSC4251_4252 : Compléments — Outils JFlex et CUP

Pascal Hennequin, J. Paul Gibson, Denis Conan

Novembre 2024
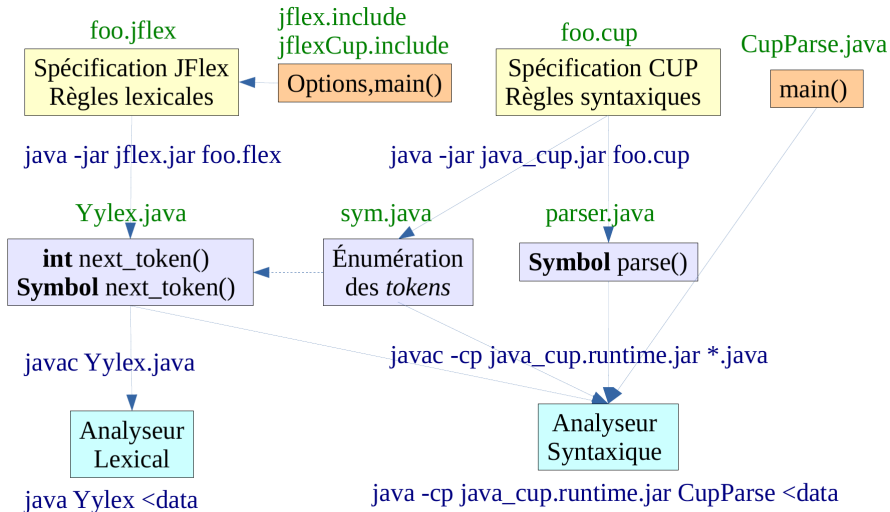
Sommaire

2 Format des spécifications

Jflex

```
package cours.compilation
import cours.compilation.common
%%
%include Jflex.include
%include JflexCup.include
%{
%}
%init{
%init}
%eof{
%eof}
ANY   = [^]
BLANC = [ \t]
%%
[a-zA-Z]  [a-zA-Z0-9]* { ECHO("ID"); }
[ \t] | {NL}    { ECHO(); }
{ANY}          { WARN("Invalid char"); }
```

Cup

```
package cours.compilation
import cours.compilation.*
parser code {:  /* ... */ :}
action code {:  int i ; :};
init with {: i=0; :}
terminal int TOK ;
terminal TIK, TAK ;
nonterminal int sym, var;
precedence left TIK;
precedence right TAK ;
start with sym,;
sym ::=  sym:s TOK:n  {: RESULT=s+n; :}
    |  var:v          {: RESULT=v; :}
;
var ::= /* vide */   {: RESULT=0;  :}
    | TIK           {: RESULT=42 ;  i++; :}
;
```

- CUP-*dump* :

```
terminal TOK;
nonterminal list;
list ::= /* vide */
    | TOK
    | list TOK
;
```

```
===== Terminals =====
[0]EOF [1]error [2]TOK
===== Non terminals ==
[0]list
===== Productions ===
[0] list ::=
[1] $START ::= list EOF
[2] list ::= TOK
[3] list ::= list TOK
```

```
===== Viable Prefix Recognizer ===
---lalr_state [0]:
  [list ::= (*) TOK , {EOF TOK }]
  [$START ::= (*) list EOF , {EOF }]
  [list ::= (*) list TOK , {EOF TOK }]
  [list ::= (*) , {EOF TOK }]
transition on TOK to state [2]
transition on list to state [1]
---lalr_state [1]:
  [$START ::= list (*) EOF , {EOF }]
  [list ::= list (*) TOK , {EOF TOK }]
transition on TOK to state [4]
transition on EOF to state [3]
---lalr_state [2]:
  [list ::= TOK (*) , {EOF TOK }]
---lalr_state [3]:
  [$START ::= list EOF (*) , {EOF }]
---lalr_state [4]:
  [list ::= list TOK (*) , {EOF TOK }]
```

```
Warning : *** Shift/Reduce conflict
              found in state #0
  between list ::= (*)
  and    list ::= (*) TOK
  under symbol TOK
```

■ **Même exemple avec « `CUP` *Eclipse Plugin* »**

lalr_state [3]:
[$START ::= list EOF (*) , {EOF }]

lalr_state [4]:
[list ::= list TOK (*) , {EOF TOK }

R1

R3

EOF

TOK

R2

lalr_state [2]:
[list ::= TOK (*) , {EOF TOK }]]

lalr_state [1]:
[$START ::= list (*) EOF , {EOF }]
[list ::= list (*) TOK , {EOF TOK }]

list

Conflit shift/reduce

TOK

R0

lalr_state [0]:
[list ::= (*) TOK , {EOF TOK }]
[$START ::= (*) list EOF , {EOF }]
[list ::= (*) list TOK , {EOF TOK }]
[list ::= (*) , {EOF TOK }]