Grammatica LispKit

G_{LK 1} non LL(1)

```
Prog ::= let Bind in Exp end | letrec Bind in Exp end
Bind ::= var = Exp X

X ::= and Bind | epsilon
Exp ::= Prog | lambda(Seq_Var) Exp | ExpA | OPP(Seq_Exp) | if Exp then Exp else Exp
ExpA ::= T E1
E1 ::= OPA T E1 | epsilon
T ::= F T1
T1 ::= OPM F T1 | epsilon
F ::= var Y | exp_const | (ExpA)
Y ::= (Seq_Exp) | epsilon
OPA ::= + |
OPM ::= * | /
OPP ::= cons | car | cdr | eq | leq | atom
Seq_Exp ::= Exp Seq_Exp | epsilon
Seq_Var ::= var Seq_Var | epsilon
```

G_{LK} LL(1)

```
1 Prog ::= let Bind in Exp end | letrec Bind in Exp end
2 Bind ::= var = Exp X
3 X ::= and Bind | epsilon
4 Exp ::= Prog | lambda(Seq Var) Exp | ExpA | OPP(Seq Exp) | if Exp then Exp else Exp
5 ExpA ::= T E1
6 E1 ::= OPA T E1 | epsilon
7 T ::= F T1
8 T1 ::= OPM F T1 | epsilon
9 F ::= var Y | exp_const | (ExpA)
10 Y ::= (Seq_Exp) | epsilon
11 OPA ::= + |
12 OPM ::= * | /
13 OPP ::= cons | car | cdr | eq | leq | atom
14 Seq_Exp ::= Exp Separator | epsilon
15 Separator ::= , Exp Separator | epsilon
16 Seq_Var ::= var Seq_Var | epsilon
```