*Transformation de la grammaire :*

1. La couleur rouge l’élimination de la récursivité à gauche.
2. La couleur bleue l’élimination de l’ambiguïté.

P → program id ;

Dcl

Inst\_composée.

Proc p() :

Accepter(program) ;

Accepter(id) ;DCL() ;inst\_composee() ;accepter(.) ;

Fin

~~Dcl → Dcl var Liste\_id : Type ; | Ꜫ~~

Dcl →dcl

dcl → var Liste\_id : Type ; dcl | Ꜫ

proc DCL() :

dcl() ;

fin

proc dcl() :

si symbole = var alors

accepter(var) ;List\_id() ;accepter( :) ;type() ;accepter( ;) ;accepter( ;) ;dcl() ;

fin

~~Liste\_id → id | Liste\_id , id~~

Liste\_id → id listid

listid → ,id listid| Ꜫ

proc List\_id():

accepter(id);

listed();

fin

proc listid():

si symbole = , alors

accepter(,);accepter(id) ;listid() ;

fin

Type →integer | char

Proc type():

Si symbole = integer alors

Accepter(interger) ;

Sinonsi symbole = char alors

Accepter (char) ;

Else

Error() ;

Finsi

Fin

Inst\_composée → begin

Inst

End

Proc inst\_composee():

Accepter(begin);inst();accepter(end);accepter(.);

Fin

inst → Liste\_inst | Ꜫ

~~Liste\_inst → I | Liste\_inst ; I~~

Liste\_inst → I listinst

listinst→ ; I listinst| Ꜫ

Proc List\_inst() :

I();listinst();

Fin

Proc listinst ():

Si symbole = ; alors

Accepter(;);I();listinst();

Finsi

fin

I → Id := Exp\_simple | if Exp then I else I |

while Exp do I

read(id) | readln(id)| write(id) | writeln(id)

~~Exp → Exp\_simple | Exp\_simple oprel Exp\_simple~~

Exp → Exp\_simple eexp

eexp → oprel Exp\_simple | Ꜫ

proc Exp() :

Exp\_simple() ;eexp() ;

Fin

Proc eexp() :

Si symbole = operel alors

Accepter(symbole.att) ;Exp\_simple() ;

Fin

Proc Exp\_simple() :

Terme() ;exp\_simple() ;

Fin

~~Exp\_simple → Terme | Exp\_simple opadd Terme~~

Exp\_simple → Terme exp\_simple

Exp\_simple → opadd Terme exp\_simple | Ꜫ

Proc exp\_simple() :

Si symbole = opeadd alors

Accepter(symbole .att) ;Terme() ;exp\_simple() ;

Finsi

Fin

~~Terme → Facteur | Terme opmul Facteur~~

Terme → Facteur terme

terme → opmul Facteur terme | Ꜫ

Proc Terme () :

Facteur() ;terme() ;

Fin

Proc terme() :

Si symbole = opmul alors

Accepter(symbole.att) ;Facteur() ;terme() ;

Finsi

fin

~~Facteur → Id |nb | (exp\_simple) | Facteur~~

Facteur → Id |nb | (exp\_simple)

Proc Facteur() :

Si symbole = id alors

Accepter(id) ;

Sinonsi symbole = nb alors accpeter(nb) ;

Sinonsi symbole = ( alors accepter (() ;Exp\_simple() ;accepter()) ;

Finsi

fin