TD5

ANALYSE DESCENDANTE RÉCURSIVE

On part d'une grammaire **non** récursive gauche.

- E $\rightarrow$ TR ▶️ *une expression E est une suite de termes*
- R $\rightarrow$ + TR| ε
- T $\rightarrow$ FS ▶️ *un terme est un produit de facteurs*
- S $\rightarrow$ * FS | ε
- F $\rightarrow$ (E) | 0 | 1 | ... | 9

On note qu'intuitivement, on aurait plutôt écrit une grammaire récursive gauche :

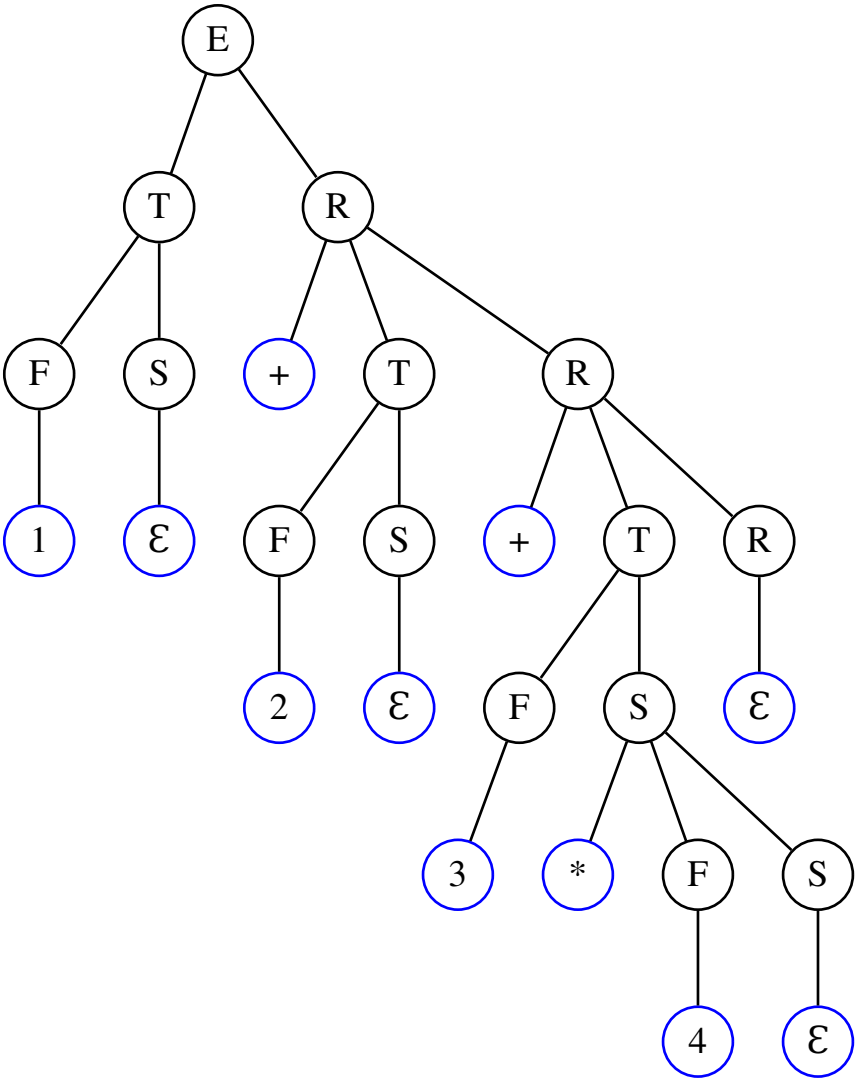
- $E \rightarrow E+E \mid E*E \mid (E) \mid 0 \ldots \mid 9$
  ou
- $E \rightarrow E+R$
- $R \rightarrow R * F$
- $F \rightarrow (E) \mid 0 \mid 1 \mid \ldots \mid 9$

On part d'une grammaire **non** récursive gauche.

- E $\rightarrow$ TR ▶️ *une expression E est une suite de termes*
- R $\rightarrow$ + TR| ε
- T $\rightarrow$ FS ▶️ *un terme est un produit de facteurs*
- S $\rightarrow$ * FS | ε
- F $\rightarrow$ (E) | 0 | 1 | ... | 9

ARBRE DE DÉRIVATION POUR 1+2+3*4.

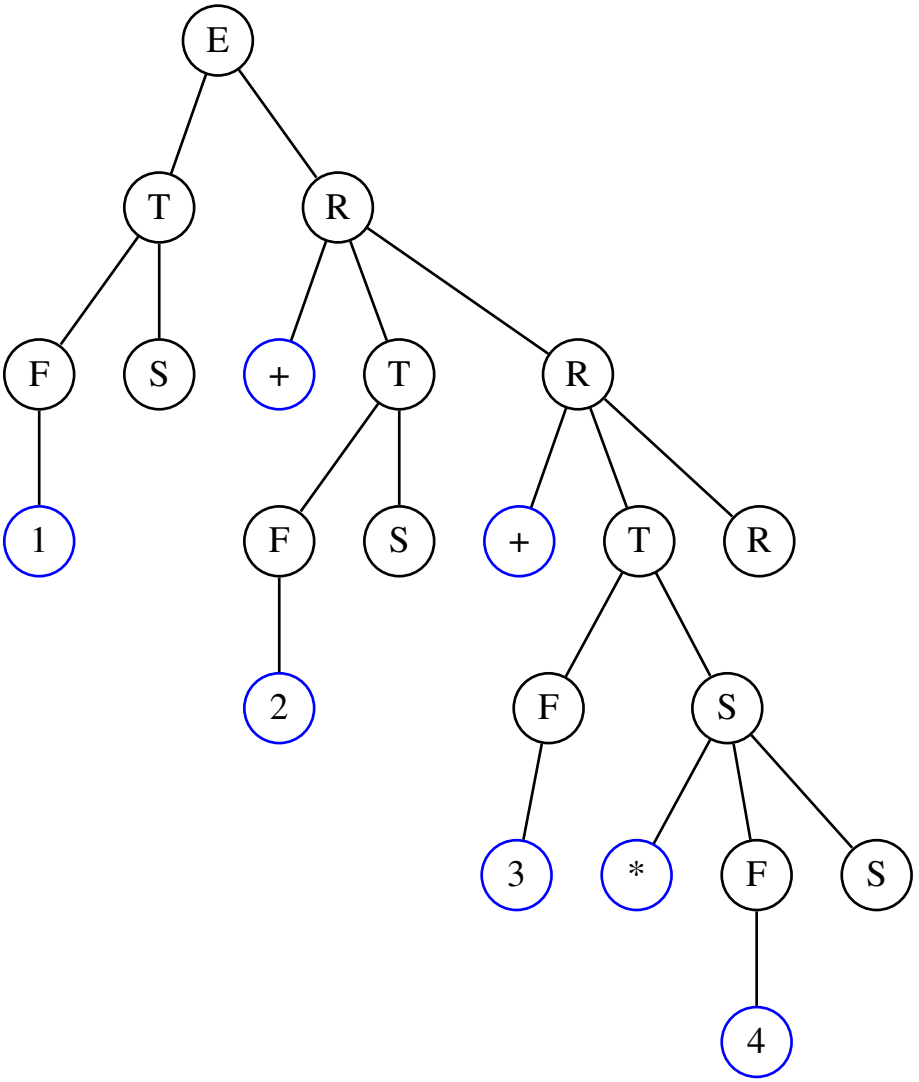
E $\rightarrow$ TR ; R $\rightarrow$ + TR| ε ; T $\rightarrow$ FS ; S $\rightarrow$ * FS | ε ; F $\rightarrow$ (E) | 0 | 1 | ... | 9

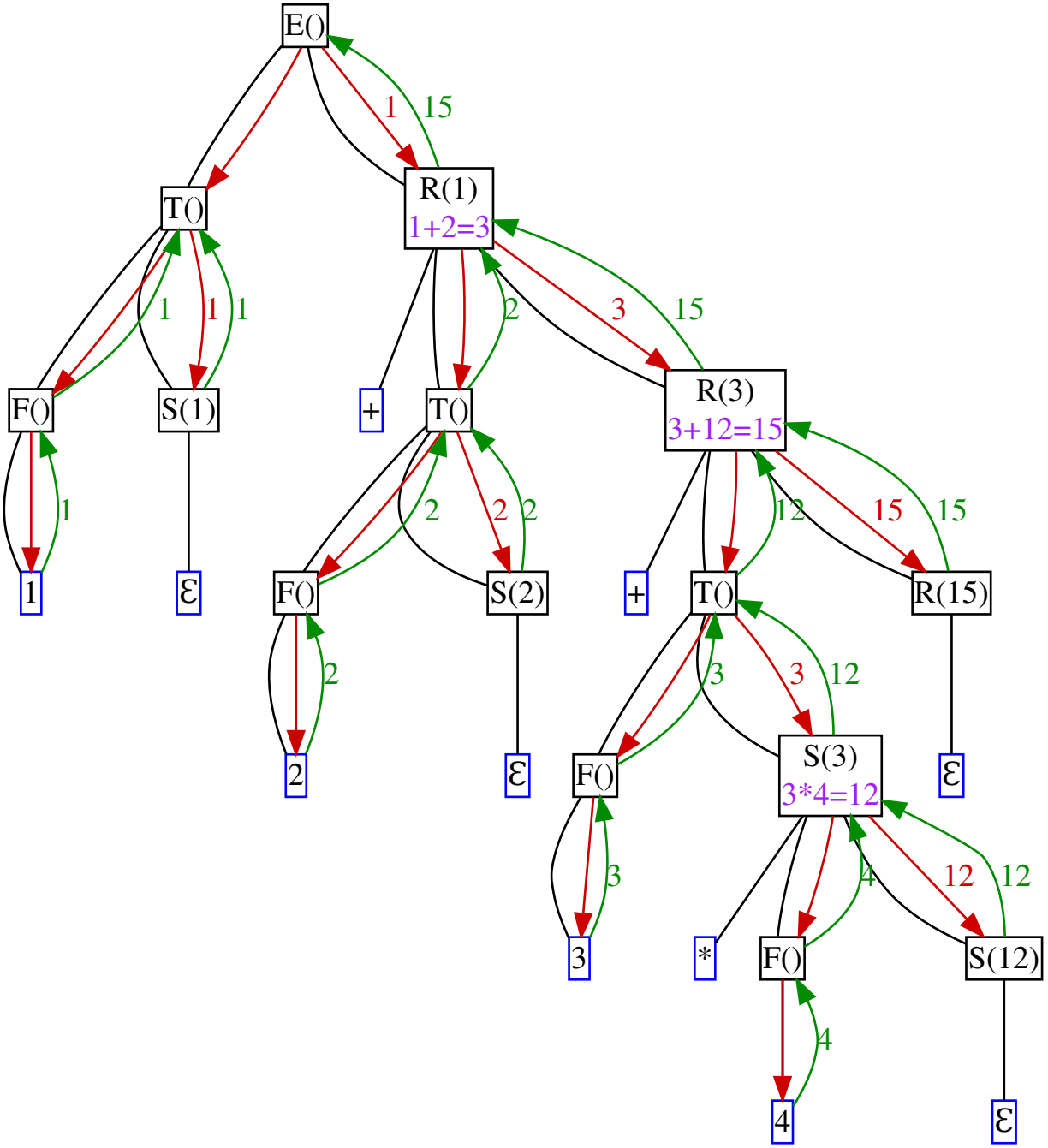


```
int jeton;                          /* caractère co
int numcar=0;                       /* numero du ca

void E(void){T();  R();} /* regle : E->TR */

void R(void){
  if (jeton=='+') {                 /* regle : R->+T
    AVANCER T(); R();}
  else ;                            /* regle : R->ep
}
void T(void){ F(); S();}            /* regle : T->FS

void S(void){
  if (jeton=='*') {                 /* regle : S->*F
    AVANCER  F(); S(); }
  else ;                            /* regle : S->ep
}
void F(void){
  if (jeton=='(') {                 /* regle : F->(E
    AVANCER  E(); TEST_AVANCE(')') }
  else if (isdigit(jeton))   AVANCER      /* reg
    else ERREUR_SYNTAXE
}
```

ARBRE DES APPELS POUR 1+2+3*4

(C'EST LE MÊME)




E()
T()
R(1)
1+2=3
F()
S(1)
+
T()
R(3)
3+12=15
1
ε
F()
S(2)
+
T()
R(15)
2
ε
F()
S(3)
3*4=12
ε
3
*
F()
S(12)
4
ε