Analyse lexicale

C'est quoi une grammaire?

Exemple

```
Statement = "if" "(" Condition ")" Statement ["else" Statement].
```

Quatre composantes

Symboles terminaux	Sont atomiques	"if", ">=", ident, number, ...
Symboles non terminaux	Sont dérivés en unités	Statement, Expr, Type, ...
productions	Règles donnant la décomposition des non terminaux	Statement = Designator "=" Expr ";". Designator = ident ["." ident]. ...
Symbole de départ	Non terminal axiome	Depart

Notation BNF ordinaire

Symboles terminaux *Sont écrits sans quottes (Ex. : ident, +, -)*
Symboles non terminaux sont écrits entre < *et* > (Ex. : <Expr>, <Term>)
Membres d'une production sont séparés par ::=

Grammaire BNF pour les expressions arithmétiques

```
<Expr>    ::=  <Sign> <Term>
<Expr>    ::=  <Expr> <Addop> <Term>

<Sign>    ::=  +
<Sign>    ::=  -
<Sign>    ::=

<Addop>   ::=  +
<Addop>   ::=  -

<Term>    ::=  <Factor>
<Term>    ::=  <Term> <Mulop> <Factor>

<Mulop>   ::=  *
<Mulop>   ::=  /

<Factor>  ::=  ident
<Factor>  ::=  number
<Factor>  ::=  ( <Expr> )
```

- Alternatives sont transformées en productions séparées
- Répétition doivent être exprimée par récursion

Avantages

- Sans méta symboles (|, (), [], {})
- Plus facile à construire un arbre syntaxique

Inconvénient

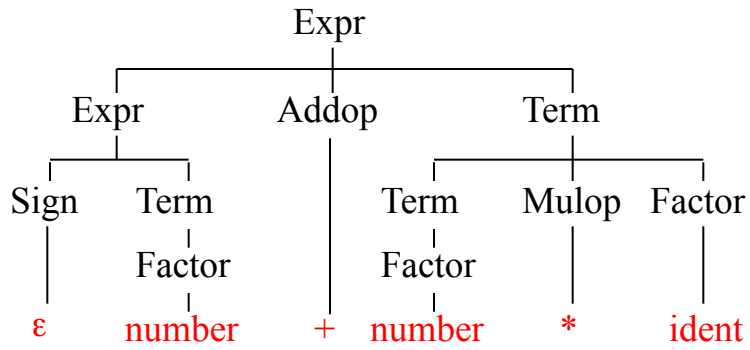
- Lourdeur

Arbre syntaxique

Montre la structure d'une phrase particulière

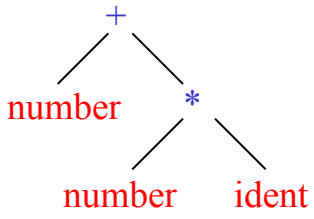
Ex. pour 10 + 3 * i

Arbre syntaxique concret (Arbre de l'analyseur)



Reflète les priorités des opérateurs : de bas en haut dans l'arbre.

Arbre syntaxique abstrait (feuilles = opérandes, nœuds internes = opérateurs)



Souvent utilisé comme une représentation interne d'un programme; Utilisé pour les optimisations.

Taches d'un scanner

1. **Délivre les symboles terminaux (unités lexicales)**

Source *Unités lexicales*

2. **Saute les caractères non significatifs**
   - blancs
   - Caractères de tabulation
   - Caractères de fin de lignes (CR, LF)
   - Commentaires

Les unités lexicales ont une structure syntaxique

```
ident =    letter { letter | digit }.
number =   digit { digit }.
if =       "i" "f".
eql =      "=" "=".
...
```