UFR IM²AG

UNIVERSITÉ Grenoble Alpes

# UE INF404 - Projet Logiciel

## Bilan de la calculette

## Extensions

## Présentation du Projet

L2 Informatique

Année 2022 - 2023

# Au menu

# Au menu

# Définition d'un langage

4 niveaux :

1. **Alphabet** = ensemble fini de **caractères**

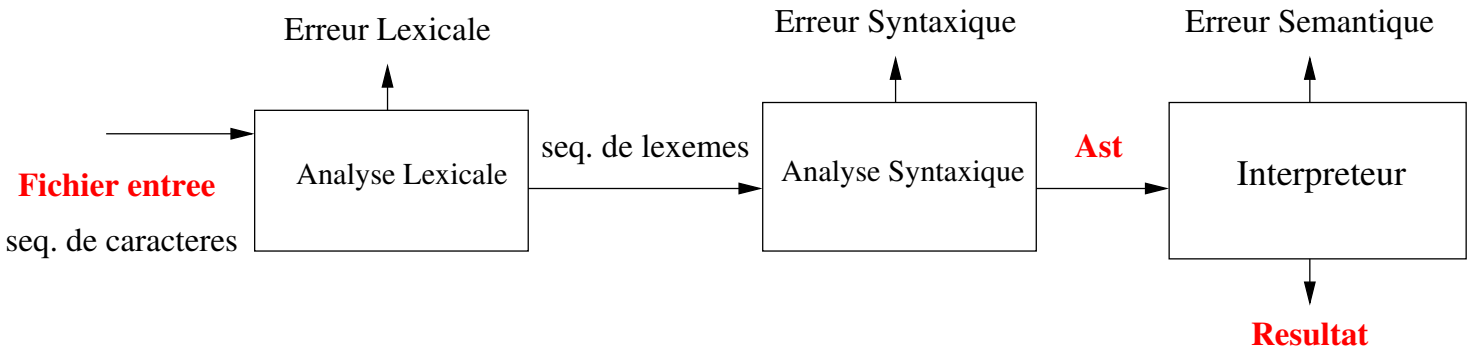
2. **Lexique** = ensemble (fini ou non) de **lexèmes**
   définies par une expression régulière/automate
   ex : ENTIER = chiffre.chiffre*

3. **Syntaxe** = ensemble de "phrases bien construites"
   phrase = séquence de lexèmes
   - langage régulier : expression régulière / automate
   - langage hors-contexte : grammaire hors-contexte

4. **Sémantique** = **sens** des phrases (syntaxiquement correctes)

# Structure d'un interpréteur

2 étapes : analyse et interprétation
(avec construction éventuelle d'un **arbre abstrait** (Ast))




Analyse lexicale : programmation d'un automate
Analyse syntaxique :

- langage régulier → programmation d'un automate
- langage hors-contexte → programmation d'une grammaire hors-contexte

⇒ algos **"systématiques"** (peuvent être produits automatiquement !)

# Au menu

# Extension 1 : noms d'opérateurs en "toutes lettres" (1)

**Exemples :**

`plus`, `moins`, `mul`, `div`
`pow`, `mod`, `abs`, ...
`cos`, `sin`, `tan`, `log`, `exp` ...

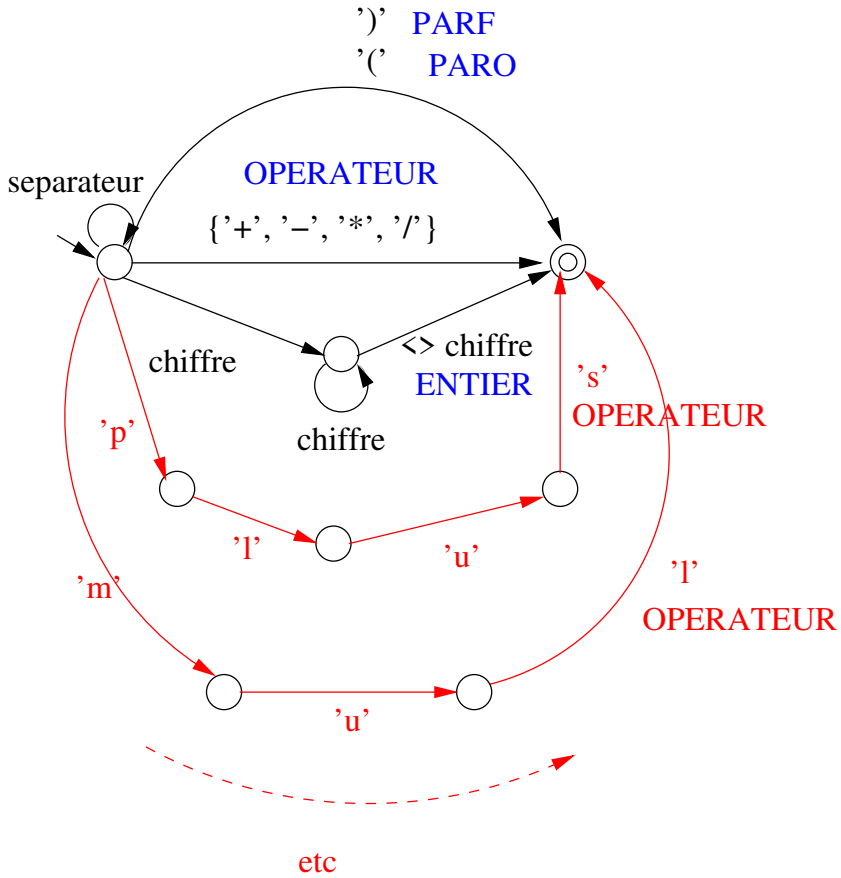
$\Rightarrow$ Etendre l'analyse lexicale ...
$\hookrightarrow$ modifier l'automate de reconnaissance des lexèmes !

Deux solutions :

1. reconnaitre individuellement chaque opérateur
2. reconnaitre une **chaîne de caractères**, puis la "filtrer" (un crible)

# Extension 1 : noms d'opérateurs en "toutes lettres" (2)

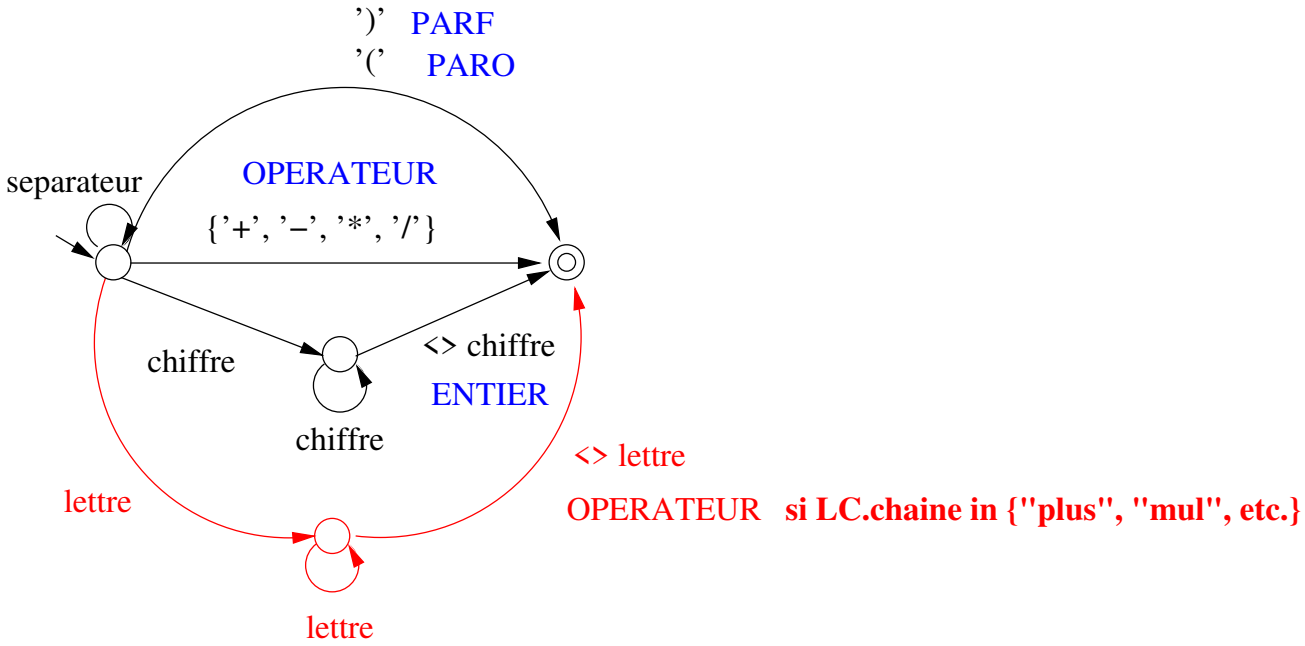
Reconnaissance individuelle de chaque opérateur :




Fastidieux à programmer et à modifier . . .

# Extension 1 : noms d'opérateurs en "toutes lettres" (3)

Reconnaissance à l'aide d'un **crible** :




Mise en oeuvre dans `reconnaitre_lexeme()` :

1. reconnaitre un lexème `L` de type *séquence de lettres*
2. si `L.chaine` $\in$ {`"plus"`, `"mul"`, `etc`} alors
   `OPERATEUR`

   sinon
   Erreur_Lexicale

# Extension 2 : le “moins unaire”

**Exemple :** `-5 + -(2*3) - 9 -- 7`

$\Rightarrow$ Etendre l'analyse syntaxique . . .donc modifier la **grammaire** !

$$
\begin{aligned}
\textit{eag} &\rightarrow \textit{seq\_terme} \\
\textit{seq\_terme} &\rightarrow \textit{terme}\ \textit{suite\_seq\_terme} \\
\textit{suite\_seq\_terme} &\rightarrow \textit{op1}\ \textit{terme}\ \textit{suite\_seq\_terme} \\
\textit{suite\_seq\_terme} &\rightarrow \varepsilon \\
\textit{terme} &\rightarrow \textit{seq\_facteur} \\
\textit{seq\_facteur} &\rightarrow \textit{facteur}\ \textit{suite\_seq\_facteur} \\
\textit{suite\_seq\_facteur} &\rightarrow \textit{op2}\ \textit{facteur}\ \textit{suite\_seq\_facteur} \\
\textit{suite\_seq\_facteur} &\rightarrow \varepsilon \\
\textit{facteur} &\rightarrow \textit{ENTIER} \\
\textit{facteur} &\rightarrow \texttt{PARO}\ \textit{eag}\ \texttt{PARF} \\
\textit{facteur} &\rightarrow \texttt{MOINS}\ \textit{facteur} \\
\textit{op1} &\rightarrow \texttt{PLUS} \\
\textit{op1} &\rightarrow \texttt{MOINS} \\
\textit{op2} &\rightarrow \texttt{MUL}
\end{aligned}
$$

# Extension 2 : le “moins unaire” (mise en oeuvre)

1. Modifier la construction de l’**arbre abstrait** :
   - définir un nouveau noeud de type “moins unaire”, avec un seul fils
   - écrire une fonction de construction `creer_op_unaire(...)`
   - modifier `Rec_facteur()` pour prendre en compte la nouvelle règle

```
Rec_facteur (resultat A : AsT) =
   A1 : Ast ;
   selon LC.nature
    cas ENTIER :  ...
    cas PARO : ...
        cas MOINS : Avancer ; Rec_facteur (A1) ;
                                  A := creer_op_unaire (MOINS, A1) ;
    autre : Erreur
   fselon
fin
```

2. Modifier la **fonction d’évaluation** (`int Evaluer(Ast A)`)

# Au menu

# Objectifs du projet

## Ecrire un interpréteur

1. choisir ce que l'on veut interpréter ...
2. définir le langage d'entrée
   alphabet, lexique, syntaxe, sémantique
3. écrire les fonctions d'analyse (lexicale et syntaxique)
4. définir et produire l'Ast
5. écrire le "traitement" de l'Ast

$\Rightarrow$ même démarche que pour la calculette

(et réutilisation partielle possible de certains modules !)

# Quelques pistes possibles . . .

- calculatrice étendue . . .
- exécution/interprétation
  langage de programmation, langage graphique, . . .
- traduction
- vérification de type
- simulation
  robot, système physique, “jeu”, . . .
- etc. . . .

# Des exemples concrets

- interpréteur mini langage Python
- simulateur mini assembleur ARM
- interpréteur programme Logo
- traduction langage L $\rightarrow$ C
- traduction langage L $\rightarrow$ HTML
- langage “graphique”
  composition de figures élémentaires
- etc. . . .

Démos !

# Des exemples concrets

- interpréteur mini langage Python
- simulateur mini assembleur ARM
- interpréteur programme Logo
- traduction langage L $\rightarrow$ C
- traduction langage L $\rightarrow$ HTML
- langage “graphique”
  composition de figures élémentaires
- etc. . . .

Démos !

# Au menu

# Interpréteur pour un langage de programmation

**En entrée :** un programme en langage “source”, des options
**En sortie :**

- le résultat de l'exécution de ce programme, avec :
  - la valeur finale des variables
  - un mode “pas-à-pas” ?
  - etc.
- . . .ou un message d'erreur explicite . . .

**Exemples :**

- interpréteur Python
- interpréteur Caml
- interpéteur pour un langage de votre choix !

# Etape 1 : enrichir le langage des EAG

- notion de variable/identificateur
  $+$ *affectation* ou liaison nom $\leftrightarrow$ valeur
- structures de contrôle (if, while, ...)
- types de données
- fonctions / procédures ($+$ récursivité ?)
- etc.

# Approche “incrémentale”

1. calculette (L0)

```
12 - 5 * (2+3)
```

2. affectations (L1)

```
X = 2 ;
Y = X + 2 ;
X = 3*Y - X ;
```

3. instructions conditionnelles et entrée-sorties (L2)

```
lire (X) ;
if X>0 then Y = X + 2 else Y = 3 ;
ecrire (Y) ;
```

4. instruction itérative (L3)
5. types, fonctions, etc.

# Approche “incrémentale”

1. calculette (L0)

```
12 - 5 * (2+3)
```

2. affectations (L1)

```
X = 2 ;
Y = X + 2 ;
X = 3*Y - X ;
```

3. instructions conditionnelles et entrée-sorties (L2)

```
lire (X) ;
if X>0 then Y = X + 2 else Y = 3 ;
ecrire (Y) ;
```

4. instruction itérative (L3)
5. types, fonctions, etc.

# Approche “incrémentale”

1. calculette (L0)

```
12 - 5 * (2+3)
```

2. affectations (L1)

```
X = 2 ;
Y = X + 2 ;
X = 3*Y - X ;
```

3. instructions conditionnelles et entrée-sorties (L2)

```
lire (X) ;
if X>0 then Y = X + 2 else Y = 3 ;
ecrire (Y) ;
```

4. instruction itérative (L3)
5. types, fonctions, etc.

# Approche “incrémentale”

1. calculette (L0)

```
12 - 5 * (2+3)
```

2. affectations (L1)

```
X = 2 ;
Y = X + 2 ;
X = 3*Y - X ;
```

3. instructions conditionnelles et entrée-sorties (L2)

```
lire (X) ;
if X>0 then Y = X + 2 else Y = 3 ;
ecrire (Y) ;
```

4. instruction itérative (L3)
5. types, fonctions, etc.

# Approche “incrémentale”

1. calculette (L0)

```
12 - 5 * (2+3)
```

2. affectations (L1)

```
X = 2 ;
Y = X + 2 ;
X = 3*Y - X ;
```

3. instructions conditionnelles et entrée-sorties (L2)

```
lire (X) ;
if X>0 then Y = X + 2 else Y = 3 ;
ecrire (Y) ;
```

4. instruction itérative (L3)
5. types, fonctions, etc.

# Important pour la suite . . . !

## Cette semaine en TP

**fin calculette ! ! !** . . .ou **début du projet ...** (TP5)

## Sans oublier :

Chaque **binôme** dépose sur Moodle :

- les sources de la calculette (achevée ou non)
- servira d'**inscription** pour le projet !

## Et enfin ...

Choisir un **sujet de projet**

- soit suivre les cours et TPs proposés
  (interpréteur pour langage de programmation impératif)
- soit définir son **propre sujet** dans le cadre prévu . . .

# Dernière info : partiel la semaine du 13 mars !

- **Durée :** une heure
- **Coefficient :** 20% de la note de l'UE INF404
- **Documents autorisés :** une feuille A4 recto/verso
- **Compétences attendues :**
  - définition d'un langage (lexique, syntaxe, sémantique)
  - analyse lexicale, automate
  - analyse syntaxique,
    écriture **systématique** d'un analyseur à partir d'une grammaire hors-contexte
  - arbre de dérivation, arbre abstrait
  - parcours (récursif) d'arbre

Exemples de sujets (et corrigés) disponibles sur Moodle . . .