Au menu

Au menu

Définition d'un langage

4 niveaux :

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

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

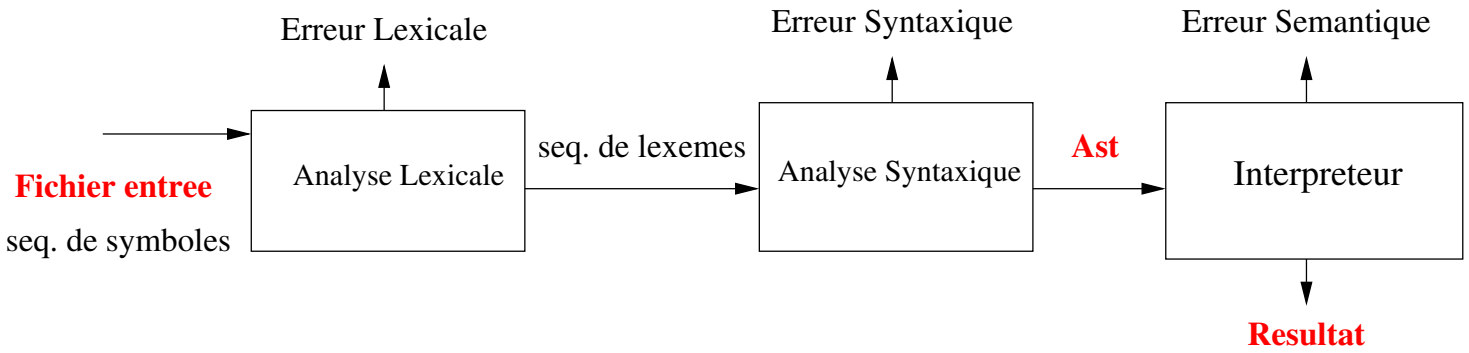
3. **Syntaxe** = ensemble de "phrases bien construites"
   phrase = séquence de lexèmes
   - langage régulier : expression régulière / automate
   - langage hors-contexte : grammaire hors-contexte

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 $\rightarrow$ programmation d'un automate
- langage hors-contexte $\rightarrow$ programmation d'un grammaire (LL(1))

$\Rightarrow$ algos **"systématiques"** (peuvent être produits automatiquement !)

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 . . .

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

Des exemples concrets

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

Au menu

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.

Langage L1 : séquences d'affectations

Exemple

```
X := 12 * 3 ;
Y := X + 5 ;
X := X * 2 ;
```

Analyse lexicale : 3 nouveaux lexèmes . . .

- IDF = seq. non vide de lettre/chiffre (débutant par lettre)
- AFF = opérateur d'affectation (`:=`)
- SEPAFF = `';'` (et fin_de_ligne devient un séparateur)

Analyse syntaxique : étendre la grammaire

$$
\begin{aligned}
Seq_Aff &\rightarrow Aff\ Seq_Aff \\
Seq_Aff &\rightarrow \varepsilon \\
Aff &\rightarrow \texttt{IDF AFF}\ Eag\ \texttt{SEPAFF}
\end{aligned}
$$

et ajouter $Facteur \rightarrow \texttt{IDF}$

Langage L1 : séquences d'affectations

Exemple

```
X := 12 * 3 ;
Y := X + 5 ;
X := X * 2 ;
```

Analyse lexicale : 3 nouveaux lexèmes . . .

- IDF = seq. non vide de lettre/chiffre (débutant par lettre)
- AFF = opérateur d'affectation (:=)
- SEPAFF = ';' (et fin_de_ligne devient un séparateur)

Analyse syntaxique : étendre la grammaire

$$\begin{aligned} Seq_Aff &\rightarrow Aff\ Seq_Aff \\ Seq_Aff &\rightarrow \varepsilon \\ Aff &\rightarrow \texttt{IDF AFF}\ Eag\ \texttt{SEPAFF} \end{aligned}$$

et ajouter $Facteur \rightarrow \texttt{IDF}$

Langage L1 : séquences d'affectations

Exemple

```
X := 12 * 3 ;
Y := X + 5 ;
X := X * 2 ;
```

Analyse lexicale : 3 nouveaux lexèmes . . .

- IDF = seq. non vide de lettre/chiffre (débutant par lettre)
- AFF = opérateur d'affectation (:=)
- SEPAFF = ';' (et fin_de_ligne devient un séparateur)

Analyse syntaxique : étendre la grammaire

$$
\begin{aligned}
\mathit{Seq_Aff} &\rightarrow \mathit{Aff}\ \mathit{Seq_Aff} \\
\mathit{Seq_Aff} &\rightarrow \varepsilon \\
\mathit{Aff} &\rightarrow \texttt{IDF AFF}\ \mathit{Eag}\ \texttt{SEPAFF}
\end{aligned}
$$

et ajouter $\mathit{Facteur} \rightarrow \texttt{IDF}$

Evaluation ?

```
X := 12 * 3 ;
Y := X + 5 ;
X := X + Y ;
```

1. calculer `12 * 3` (c'est une EAG !)
2. mémoriser le résultat (36) dans la variable `X`
3. calculer `X + 12` (presque une EAG ?)
   $\rightarrow$ utiliser la valeur 36 pour `X`
4. mémoriser le résultat (41) dans la variable `Y`
5. calculer `X + Y`
   (en utilisant les valeurs 36 et 41 pour `X` et `Y`)
6. mémoriser le résultat (77) dans la variable `X`

Evaluation ?

```
X := 12 * 3 ;
Y := X + 5 ;
X := X + Y ;
```

1. calculer `12 * 3` (c'est une EAG !)
2. mémoriser le résultat (36) dans la variable `X`
3. calculer `X + 12` (presque une EAG ?)
   $\rightarrow$ utiliser la valeur 36 pour `X`
4. mémoriser le résultat (41) dans la variable `Y`
5. calculer `X + Y`
   (en utilisant les valeurs 36 et 41 pour `X` et `Y`)
6. mémoriser le résultat (77) dans la variable `X`

Evaluation ?

```
X := 12 * 3 ;
Y := X + 5 ;
X := X + Y ;
```

1. calculer `12 * 3` (c'est une EAG !)
2. mémoriser le résultat (36) dans la variable `X`
3. calculer `X + 12` (presque une EAG ?)
   $\rightarrow$ utiliser la valeur 36 pour `X`
4. mémoriser le résultat (41) dans la variable `Y`
5. calculer `X + Y`
   (en utilisant les valeurs 36 et 41 pour `X` et `Y`)
6. mémoriser le résultat (77) dans la variable `X`

Evaluation ?

```
X := 12 * 3 ;
Y := X + 5 ;
X := X + Y ;
```

1. calculer `12 * 3` (c'est une EAG !)
2. mémoriser le résultat (36) dans la variable `X`
3. calculer `X + 12` (presque une EAG ?)
   $\rightarrow$ utiliser la valeur 36 pour `X`
4. mémoriser le résultat (41) dans la variable `Y`
5. calculer `X + Y`
   (en utilisant les valeurs 36 et 41 pour `X` et `Y`)
6. mémoriser le résultat (77) dans la variable `X`

Evaluation ?

```
X := 12 * 3 ;
Y := X + 5 ;
X := X + Y ;
```

1. calculer `12 * 3` (c'est une EAG !)
2. mémoriser le résultat (36) dans la variable `X`
3. calculer `X + 12` (presque une EAG ?)
   $\rightarrow$ utiliser la valeur 36 pour `X`
4. mémoriser le résultat (41) dans la variable `Y`
5. calculer `X + Y`
   (en utilisant les valeurs 36 et 41 pour `X` et `Y`)
6. mémoriser le résultat (77) dans la variable `X`

Evaluation ?

```
X := 12 * 3 ;
Y := X + 5 ;
X := X + Y ;
```

1. calculer `12 * 3` (c'est une EAG !)
2. mémoriser le résultat (36) dans la variable `X`
3. calculer `X + 12` (presque une EAG ?)
   $\rightarrow$ utiliser la valeur 36 pour `X`
4. mémoriser le résultat (41) dans la variable `Y`
5. calculer `X + Y`
   (en utilisant les valeurs 36 et 41 pour `X` et `Y`)
6. mémoriser le résultat (77) dans la variable `X`

Evaluation ?

```
X := 12 * 3 ;
Y := X + 5 ;
X := X + Y ;
```

1. calculer `12 * 3` (c'est une EAG !)
2. mémoriser le résultat (36) dans la variable `X`
3. calculer `X + 12` (presque une EAG ?)
   $\rightarrow$ utiliser la valeur 36 pour `X`
4. mémoriser le résultat (41) dans la variable `Y`
5. calculer `X + Y`
   (en utilisant les valeurs 36 et 41 pour `X` et `Y`)
6. mémoriser le résultat (77) dans la variable `X`

Memoriser `IDF` $\leftrightarrow$ Valeur

Structure de données de “Table des Symboles”

Interface (`table_symbole.h`)

- initialiser une table vide
- insérer/remplacer un couple (IDF, valeur)
- rechercher la valeur de IDF
- afficher la table

Implémentation (`table_symbole.c`)

tableau, liste chainée, etc.

Interpréter une séquence d'affectation ?

Lors de l'analyse syntaxique :

- fonction `rec_AFF()` :

$$\mathit{Aff} \rightarrow \texttt{IDF AFF } \mathit{Eag} \texttt{ SEPAFF}$$

  - `Rec_Eag` calcule la valeur v de la partie droite
  - insérer/remplacer (IDF,v) dans la table des symboles

- fonction `Rec_Facteur` :

$$\mathit{Facteur} \rightarrow \texttt{IDF}$$

  rechercher la valeur de IDF dans la table des symboles

Important pour la suite . . . !

Cette semaine en TP

fin calculette ! ! !
ou TP5 (séquences d'affectations)

La semaine après le partiel

Chaque **binôme** envoie un mail à son enseignant de TP :

- la composition du binôme
- son idée pour le projet !