

# Au menu de l'UE Compilation

1

## Introduction

- I - Programmation dirigée par la syntaxe
- II - Analyse lexicale - Analyse syntaxique
- III – Description formelle d'un langage
  - III.1 – Expressions régulières (ou rationnelles) – Automates finis
  - III.2 – Grammaires algébriques (ou hors-contexte)

## A - Programmation par automates finis

- I – Analyse lexicale par automate fini
  - I.1 – Reconnaissance des items lexicaux
  - I.2 – Analyse lexicale et actions
  - I.3 – Analyse lexicale et erreurs
- II – Analyse syntaxique par automate fini
  - II.1 – Reconnaissance des données licites
  - II.2 – Analyse syntaxique et actions
  - III.3 - Analyse syntaxique et erreurs
- III - Programmation d'un automate fini déterministe
  - III.1 – Programmation directe
  - III.2 – Programmation par interpréteur de tables

III.3 – Traitement des erreurs dans l'analyse syntaxique

## B - Analyse syntaxique descendante de gauche à droite (DGD)

- I - Limite des automates finis
- II - Analyseur DGD procédural - Points de génération

## C – Construction d'un compilateur

- I - Compilateur
- II - Table des symboles - Compilation des déclarations
- III - Compilation des expressions - Calcul de type
- IV - Compilation des instructions
- V - Compilation des procédures
- VI - Compilation séparée - Édition de liens

## D - Automates à pile - Grammaires LL(1)

- I – Analyse DGD par automate à pile
- II - Analyse DGD et grammaire LL(1)
- III – Analyseur associé à une grammaire LL(1)

# II-Analyse syntaxique par automate fini

## II.1 – Reconnaissance des données licites

2

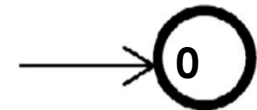
### ► Objectifs

- Contrôle de validité de la chaîne d'entrée
- Mais aussi traitement sur cette chaîne

### ► Par convention, les états sont :

- Numérotés consécutivement, de 0 à  $N_{\max}$
- 0 = état initial
- $N_{\max}$  = état final
- $N_{\max}-1$  = état erreur

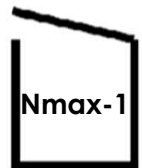
état initial



état final



état erreur



# II-Analyse syntaxique par automate fini

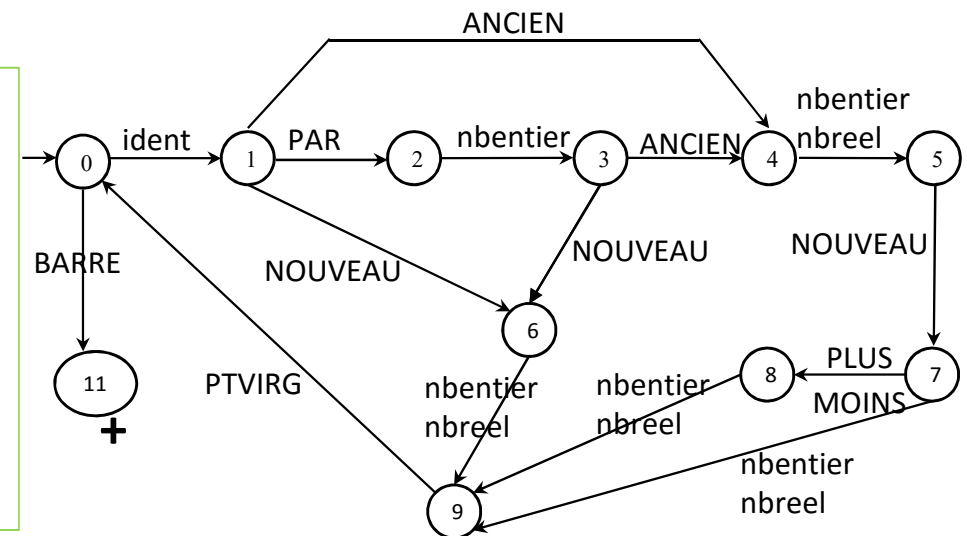
## II.1 – Reconnaissance des données licites

3

Automate : contrôle de validité de la chaîne d'entrée (reconnaissance)

**Automate fini du langage Monnaie (état d'erreur omis):**

```
( ident ( PAR nbentier )0/1  
  ( ANCIEN ( nbentier | nbréel ) NOUVEAU ( PLUS | MOINS )0/1 ( nbentier | nbréel )  
  |  
  NOUVEAU ( nbentier | nbréel )  
  )  
  PTVIRG  
  ) * BARRE
```



# II-Analyse syntaxique par automate fini

## II.2 – Analyse syntaxique et actions

4

Automate avec traitements sur la chaîne d'entrée (actions)

### Traitements sur la chaîne :

► **Objectif :** Extraction d'information

- Ex : calcul du cours le + haut / monnaie

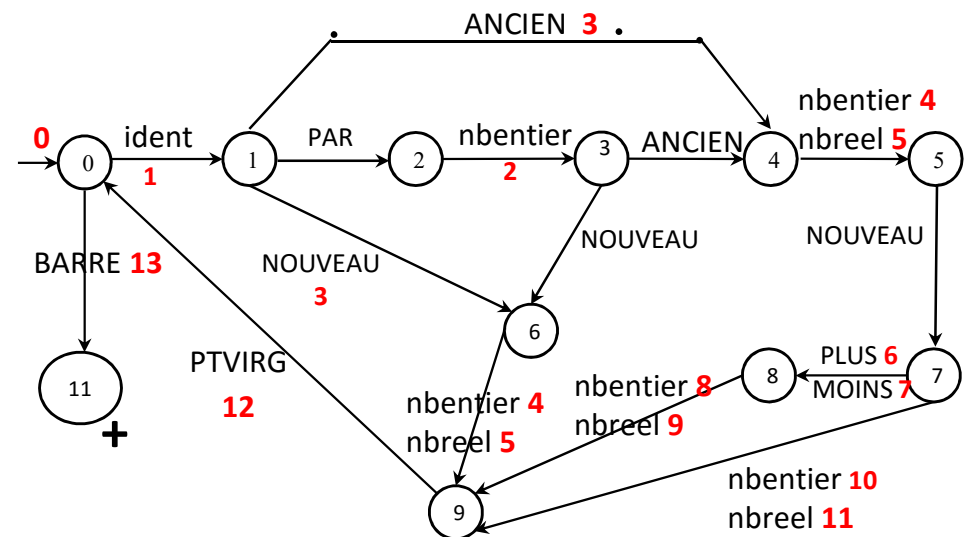
► **Principe:**

- **Action = code à exécuter sur une transition**
- **Traitement complet de la donnée = enchaînement des actions selon le parcours de l'automate**

► **Mise en œuvre :**

- Numérotation des actions conventionnelle
  - -1 = action vide (pas d'action)
  - 0 = initialisations nécessaires
  - $\geq 1$  = autres actions, pas nécessairement consécutives
- Association (N° action - transition)
- Définir le code à exécuter pour chaque action

### Automate avec actions du langage Monnaie :



# II-Analyse syntaxique par automate fini

5

## II.2 – Analyse syntaxique et actions

### Automate et traitement sur la chaîne d'entrée

#### Traitements sur la chaîne :

##### ► Action = partie de traitement :

- **Nécessité de connaître certaines valeurs**
- La reconnaissance syntaxique a besoin des codes d'items, pas des valeurs
- Les actions ont besoin des valeurs associées à certains items (attributs lexicaux)

⇒ **Définition d'attributs lexicaux dans l'analyseur lexical**

⇒ **Donner accès à ces attributs pour les actions de l'analyseur syntaxique**

#### Rappel attributs lexicaux (Monnaies) :

##### Définitions et mis à jour par Lex

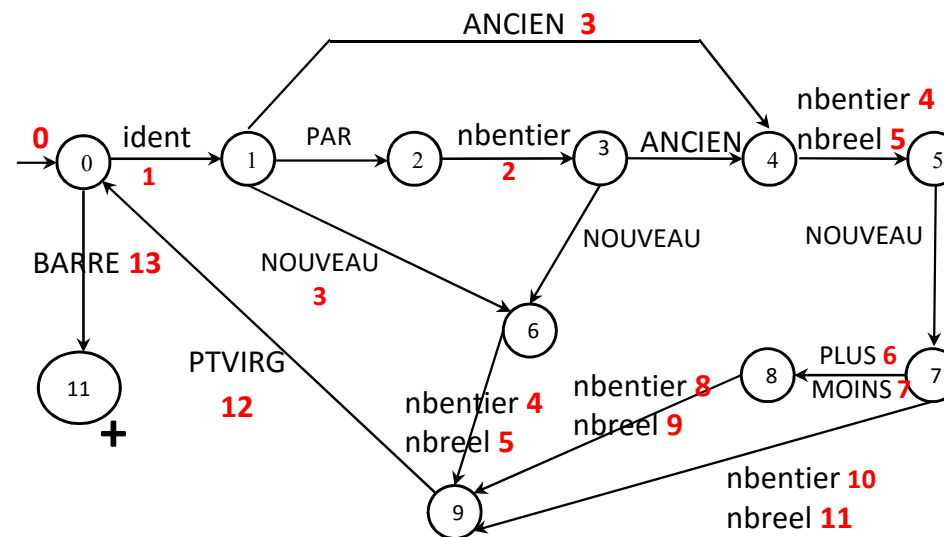
- **int valEnt** = valeur de l'item **nbentier**
- **double valReel** = valeur de l'item **nbreel**
- **int numIdCourant** = index dans une table d'identificateurs du dernier **ident** lu.
  - `String[] tabIdent` locale à l'analyseur lexical
  - **String chainIdent**(int numIdent) = chaîne de l'**ident** qui a pour code numIdent

# II-Analyse syntaxique par automate fini

## II.2 – Analyse syntaxique et actions

6

- **Exemple** : Actions associées aux transitions de l'analyseur syntaxique du langage Monnaie
- **Objectif** : calculer le cours le plus haut pour chaque monnaie



Actions cf. feuille distribuée

# II-Analyse syntaxique par automate fini

## II.2 – Analyse syntaxique et actions

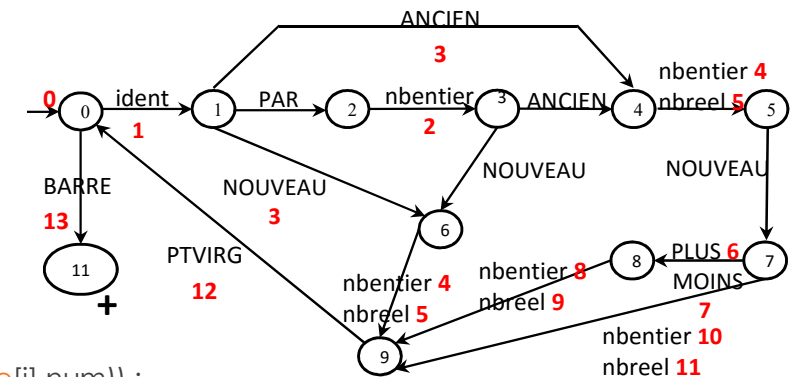
7

### Actions

```
private static final int MAXMONNAIE = 100 ; // nombre maximal de monnaies gérées
private Monnaie[] tabMonnaie = new Monnaie [MAXMONNAIE+1] ;
private int iTab = 0, numIdCourant, signe ;
private double coursCourant ; // cours maximal local de la monnaie courante
private double qte ;
0 : // action réservée à l'initialisation des variables
1 : numIdCourant = Lex.numIdCourant ;
2 : qte = Lex.valEnt ;
3 : qte = 1 ;
4 : coursCourant = Lex.valEnt / qte ;
5 : coursCourant = Lex.valReel / qte ;
6 : signe = 1 ;
7 : signe = -1 ;
8 : if (signe == 1) coursCourant = coursCourant + Lex.valEnt / qte ;
9 : if (signe == 1) coursCourant = coursCourant + Lex.valReel / qte ;
10 : if (Lex.valEnt / qte > coursCourant) coursCourant = Lex.valEnt / qte ;
11 : if (Lex.valReel / qte > coursCourant) coursCourant = Lex.valReel / qte ;
12 : int i = 1 ; while (i <= iTab && tabMonnaie[i].num != numIdCourant) i++ ;
    if (i <= iTab) { if (coursCourant > tabMonnaie[i].max) tabMonnaie[i].max = coursCourant ; }
    else if ( iTab == MAXMONNAIE) erreur("débordement de tabMonnaie") ;
    else { iTab = iTab + 1 ; tabMonnaie[iTab] = new Monnaie(numIdCourant, coursCourant) ; }
13 : for (int i = 1 ; i <= iTab ; i++) {
    System.out.println("cours max " + tabMonnaie[i].max + " pour " + Lex.chainIdent(tabMonnaie[i].num)) ;
}
```

```
public class Monnaie {
    public int num ;
    public double max ;
    public Monnaie (int num, double max) {
        this.num = num ;
        this.max = max ;
    }
}
```

### Automate



# II-Analyse syntaxique par automate finis

## II.2 – Analyse syntaxique et actions

8

- **Exercice 2** : Remplir la feuille distribuée pour l'analyse de la chaîne suivante

FSuisse par 100 ancien 62 nouveau 62.5 ; /



# II-Analyse syntaxique par automate fini

9

## II.2 – Analyse syntaxique et actions

**Exercice 3 :** Soit un langage de suite non vide d'expressions arithmétiques (sur des entiers positifs) dont les spécifications syntaxiques sont données par

$V_T = \{ \textit{nbentier}, \textit{PLUS}, \textit{ETOILE}, \textit{PTVIRG}, \textit{BARRE} \}$

$(\textit{nbentier} (\textit{PLUS} \textit{nbentier} (\textit{ETOILE} \textit{nbentier})^{0/1})^* \textit{PTVIRG})^+ \textit{BARRE}$

Q1) Donner un automate fini déterministe qui reconnaît ce langage.

Q2) Associer des actions à l'automate proposé en Q1 de manière à :

- calculer et afficher la valeur de chaque expression
- déterminer et afficher la plus grande valeur d'expression

Une suite d'expressions arithmétiques	Ex	Résultats des actions affichage suivant
	5;	Expression vaut 5
	5;	Expression vaut 5
	15+5*3;	Expression vaut 30
	1+5;	Expression vaut 6
	1+5*3;/	Expression vaut 16 Val max : 30