Au menu de l'UE Compilation

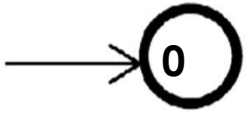
II-Analyse syntaxique par automate fini

II.1 – Reconnaissance des données licites

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

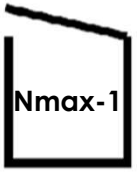
- Par convention, les états sont :
  - Numérotés consécutivement, de 0 à Nmax
  - 0 = état initial
  - Nmax = état final
  - Nmax-1 = état erreur

état initial



état final

état erreur



II-Analyse syntaxique par automate fini

II.1 – Reconnaissance des données licites

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*)

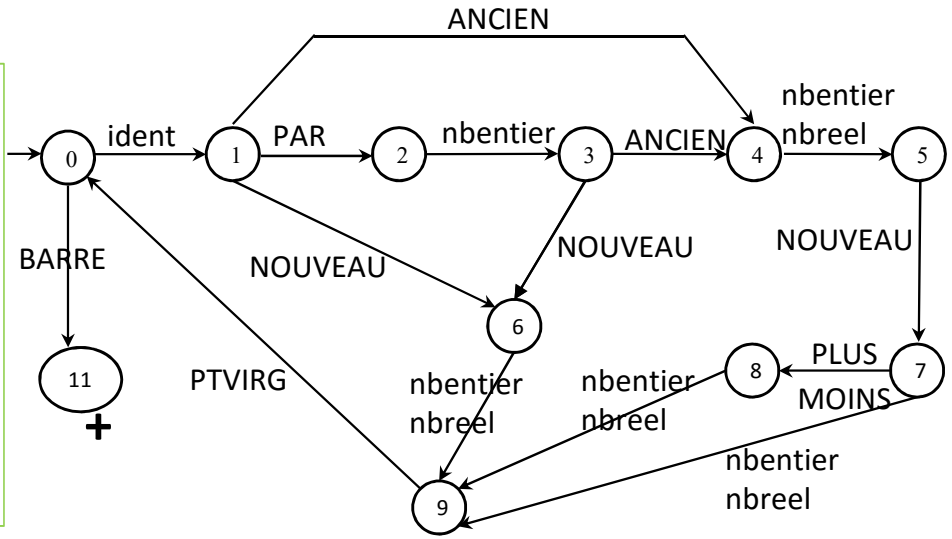
|

NOUVEAU (*nbentier* | *nbréel*)

)

PTVIRG

)* BARRE



II-Analyse syntaxique par automate fini

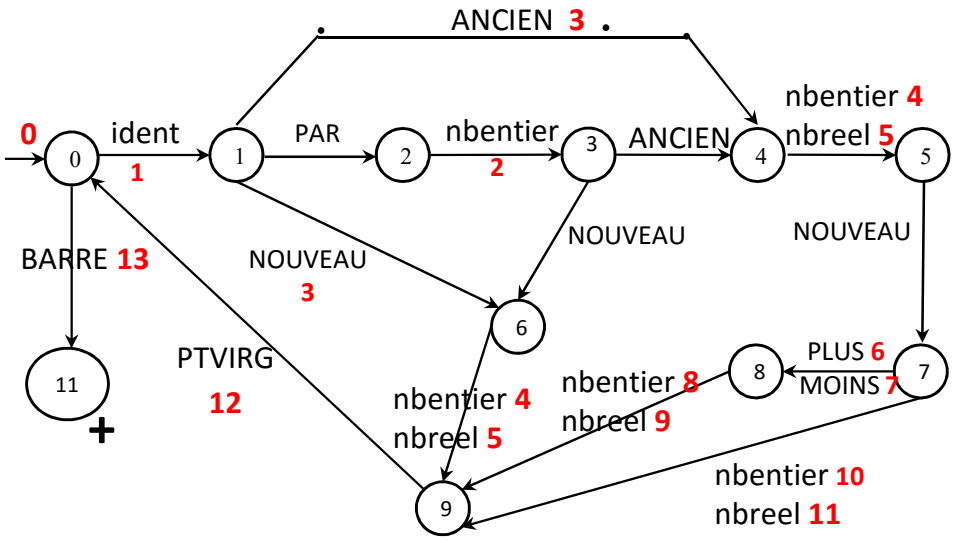
II.2 – Analyse syntaxique et actions

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 = enchainement des actions selon le parcours de l'automate
- **Mise en œuvre :**
  - Numérotation des actions conventionnelle
    - -1 = action vide (pas d'action)
    - 0 = initialisations nécessaires
    - ≥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

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) :

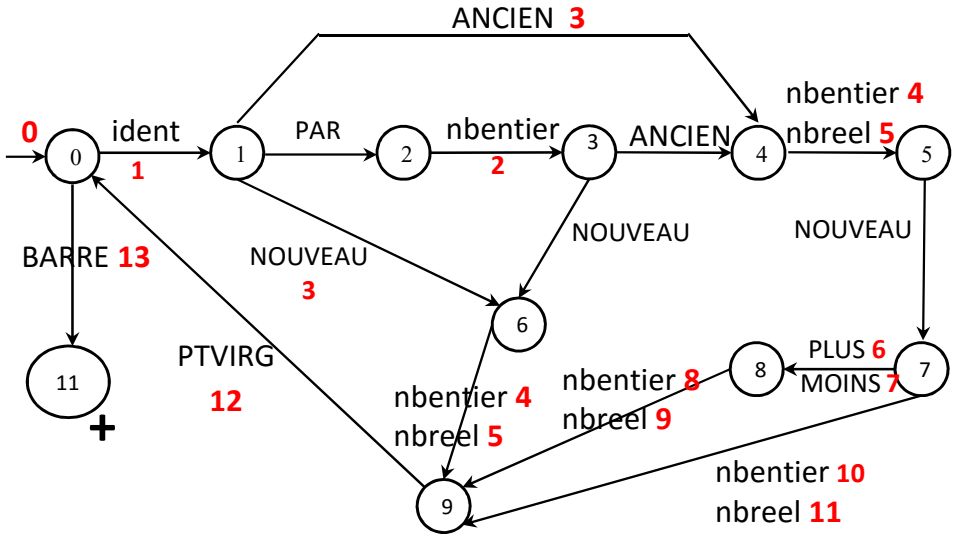
Définitions et mis à jour par Lex

- **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 chaineIdent**(int numIdent) = chaine de l'ident qui a pour code numIdent

II-Analyse syntaxique par automate fini

II.2 – Analyse syntaxique et actions

- **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

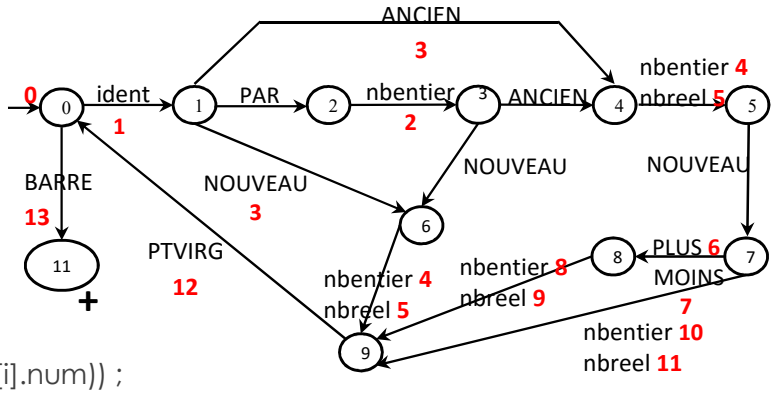
II.2 – Analyse syntaxique et actions

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.chaineIdent(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

- **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

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 = { *nbentier, PLUS, ETOILE, PTVIRG, BARRE* }

(*nbentier* (PLUS *nbentier* (ETOILE *nbentier*)$^{0/1}$)*

PTVIRG)$^{+}$

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

	Ex	Résultats des actions affichage suivant
Une suite d'expressions arithmétiques	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