Université d'Alger 1

Benyouçef benkhedda

Compilation

Analyse Lexicale

Licence 3 Informatique : SI

Présenté par : Mr. Chaouki BOUFENAR

Année universitaire

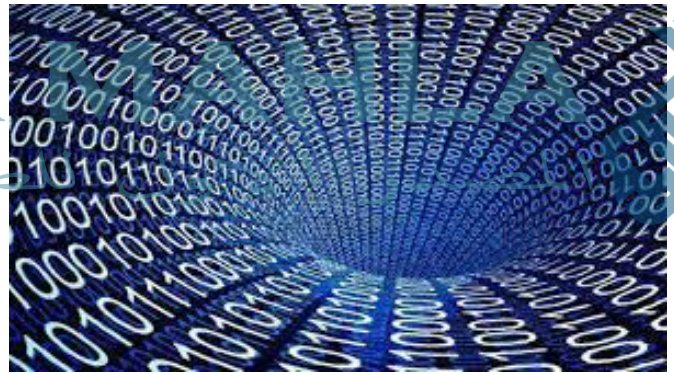
2017/2018

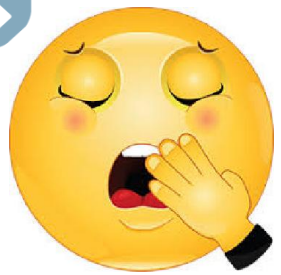
« ***Quand j'étais enfant, on m'avait dit que le Père Noël descendait par la cheminée, et que les ordinateurs se programmaient en binaire. J'ai appris depuis que la programmation se faisait de préférence dans des langages de haut niveau, plus abstraits et plus expressifs.*** »

Introduction de la thèse de X. Leroy

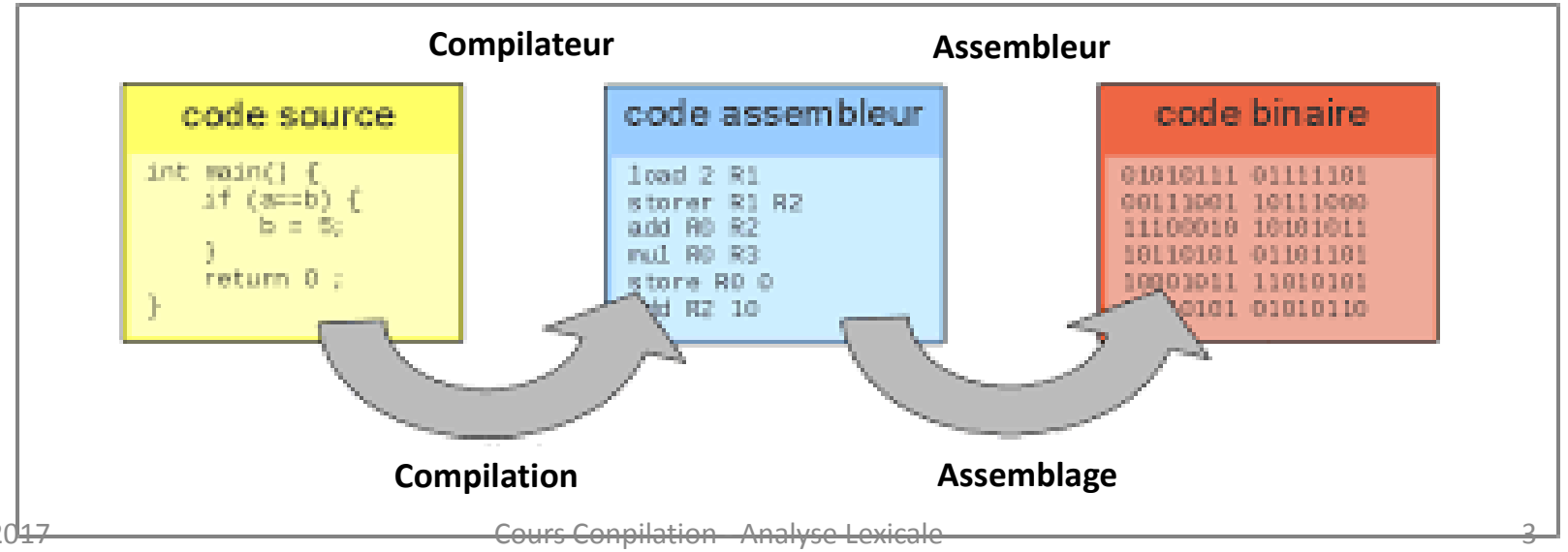
Qu’est-ce que la compilation ?

Au début de l’informatique ….





Comment on devra faire ?



Interpréteur *versus* Compilateur

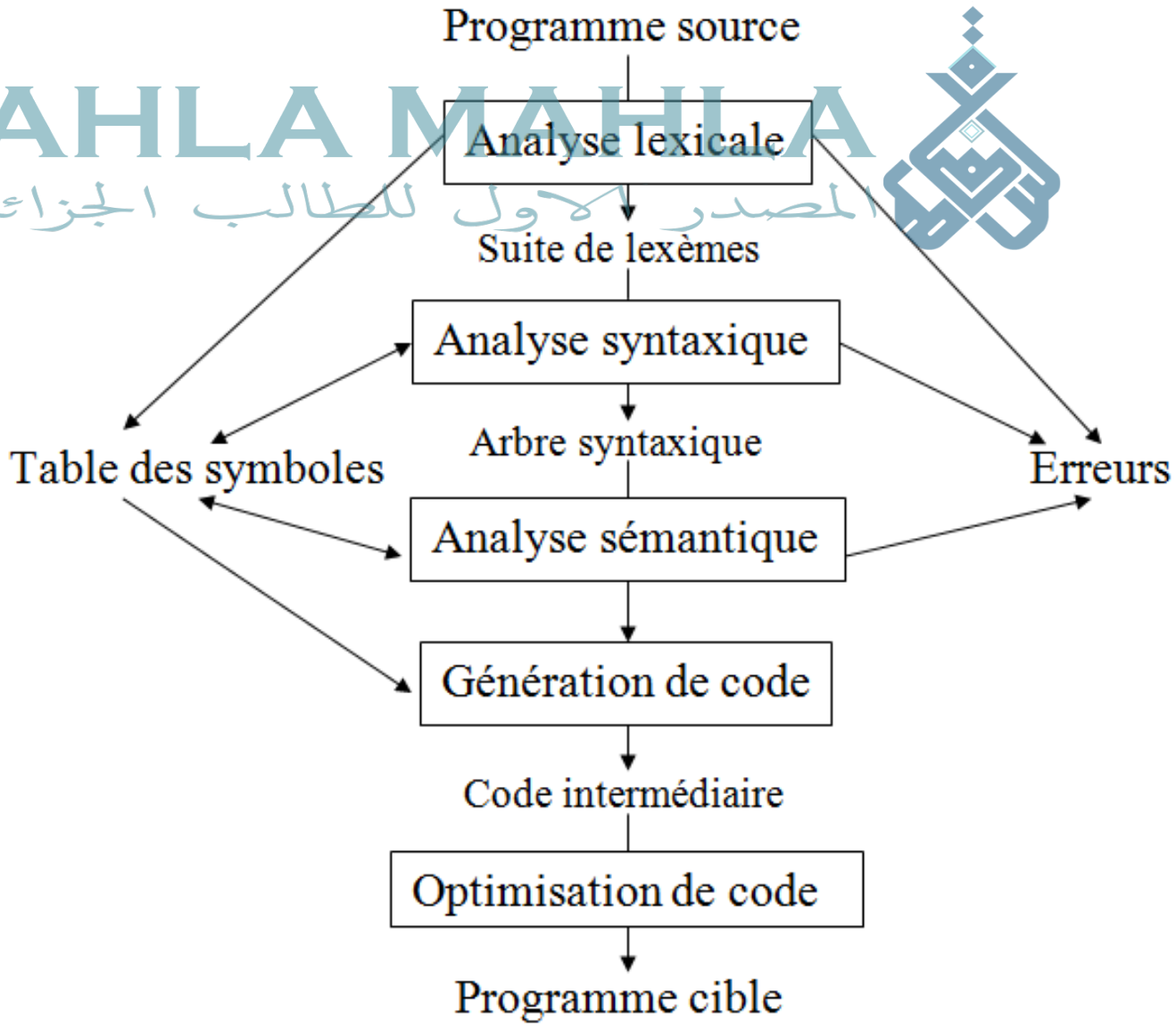
Interpréteur

- Analyse les instructions du programme source, les unes après les autres, et exécute chacune d'elles immédiatement.
- Pas de création d'un programme objet équivalent.
- ne connait pas à priori la structure du programme qu'il aura à exécuter et ne peut donc effectuer d'optimisations.
- Doit être présent sur le système à chaque fois que le programme est exécuté,
- Exécution lente (Perl, Mapple, …)

Compilateur

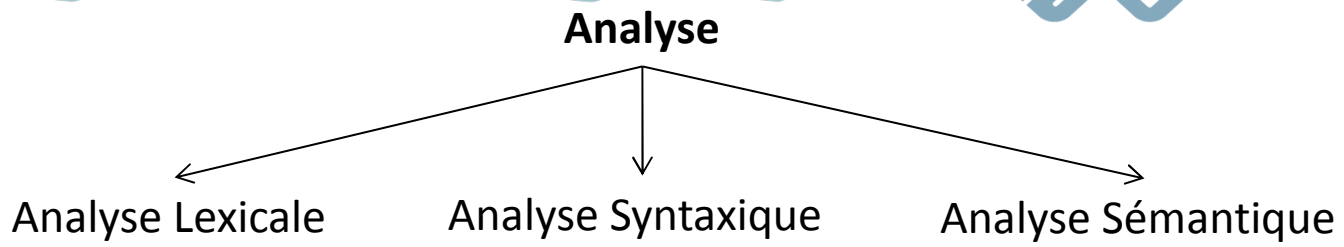
- Effectue un pré-calcul sur un programme P pour le transformer en une suite d'instructions P'.
- Création d'un code objet (fichier .obj)
- La structure du programme est connue à priori
- Ne doit pas etre présent dans le système à chaque exécution
- Exécution rapide après compilation.

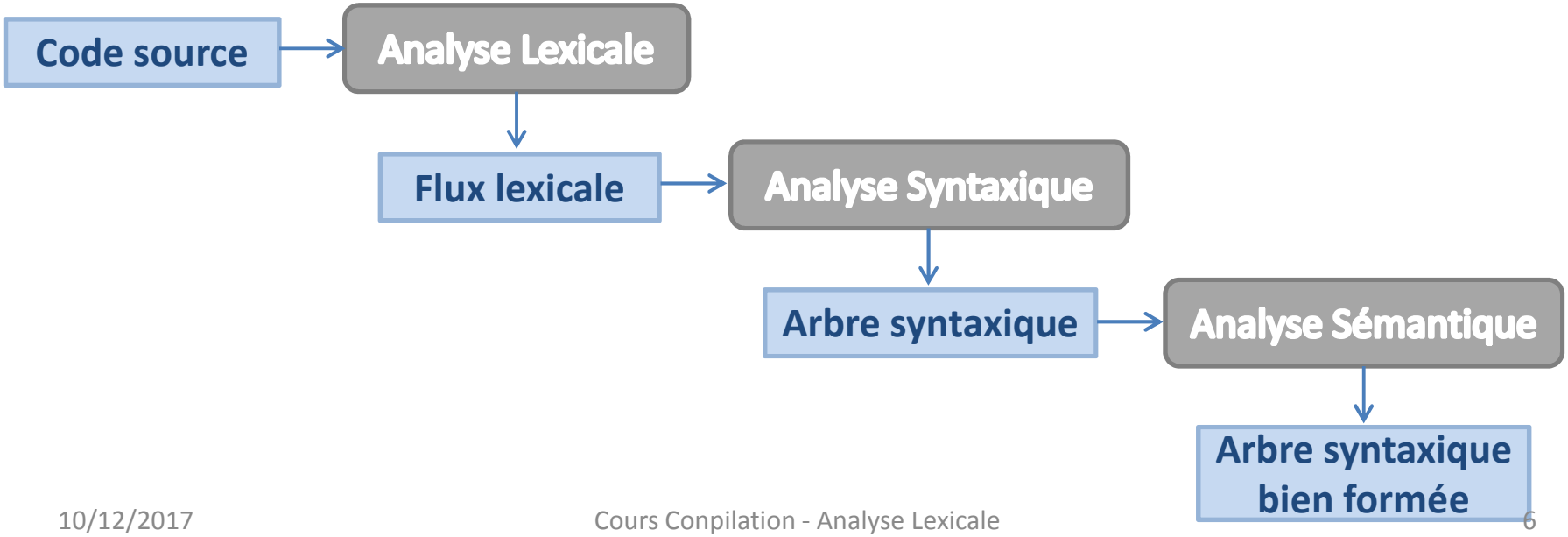
Les différentes phases de compilation



Analyse

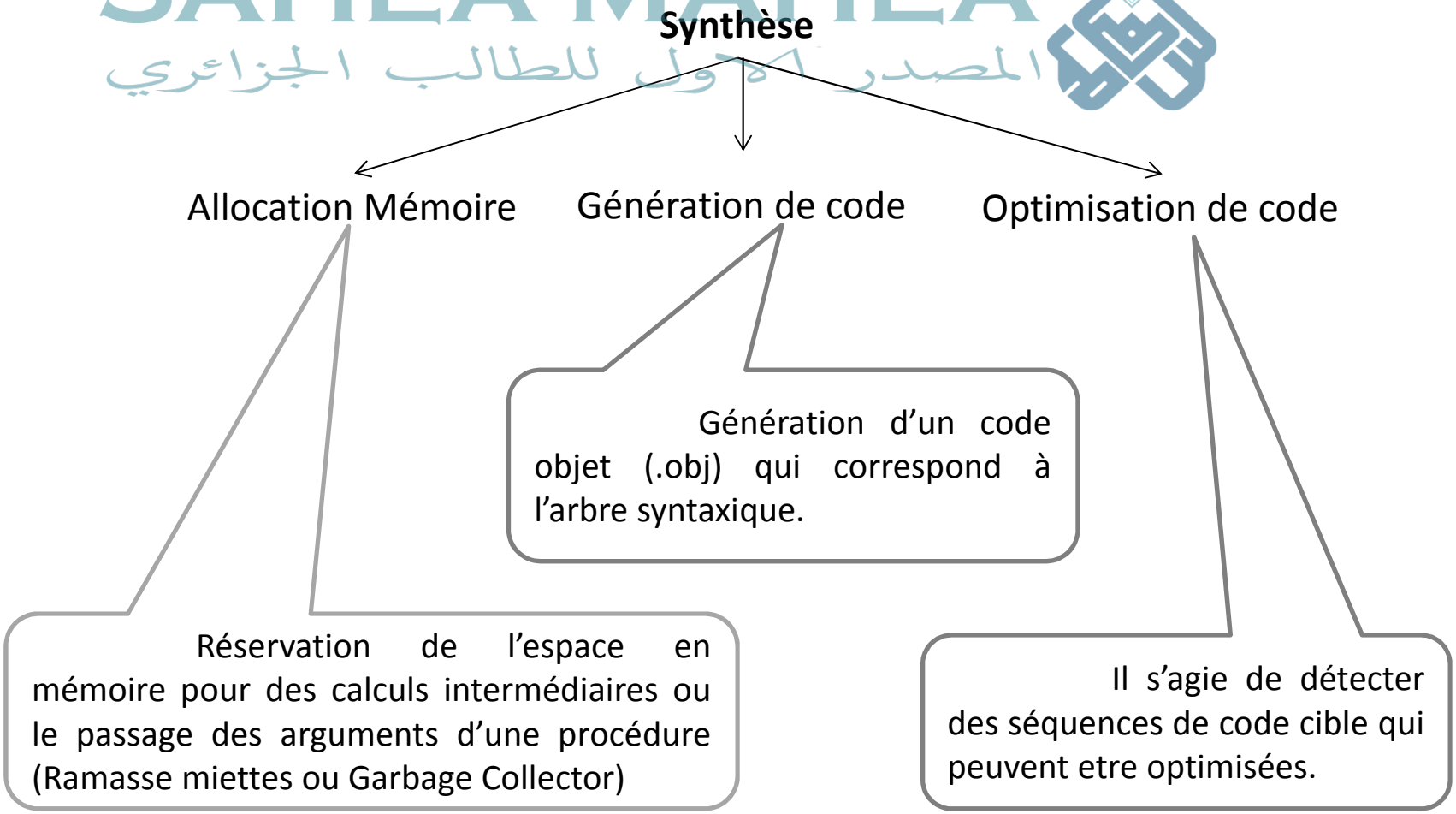
- Elle correspond à reconnaitre qu'une entrée est un programme correct du langage source.
- Doit être la plus rapide possible, c'est pourquoi on la structure en trois parties.



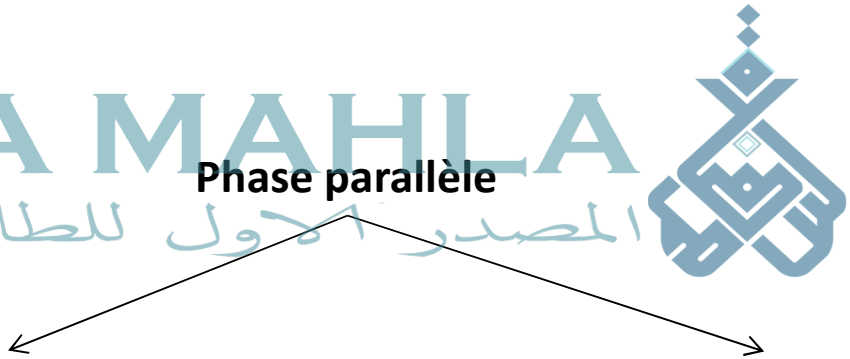


Synthèse

- Phase de reconstruction de l'expression du langage cible à partir de l'arbre syntaxique



Phase parallèle



TS associe à chaque identificateur déclaré dans le programme des divers attributs calculées au moment de l'analyse. Ces attributs fournissent des informations :

- Adresse de chaque identificateur.
- Type.
- Portée,
- Mode de passage de paramètres d'une fonction,…).

Il s'agit de les détecter et d'informer l'utilisateur le plus précisément possible

Analyse Lexicale

Quelques définitions

Une unité lexicale

Une unité lexicale est une suite de caractères qui a une signification collective.

Exemple :

OPREL des opérateurs relationnel : <, >,=

IDENT des identificateurs.

Un modèle

Un modèle est une règle associe à une unité lexicale qui décrit l'ensemble des chaines du programme qui peuvent correspondre à cette unité lexicale

Exemple : L'unité lexicale IDENT a pour modèle toute suite non vide de caractère composée de chiffre, lettre, ou des symboles et qui commence par une lettre.

Un lexème (Token en anglais)

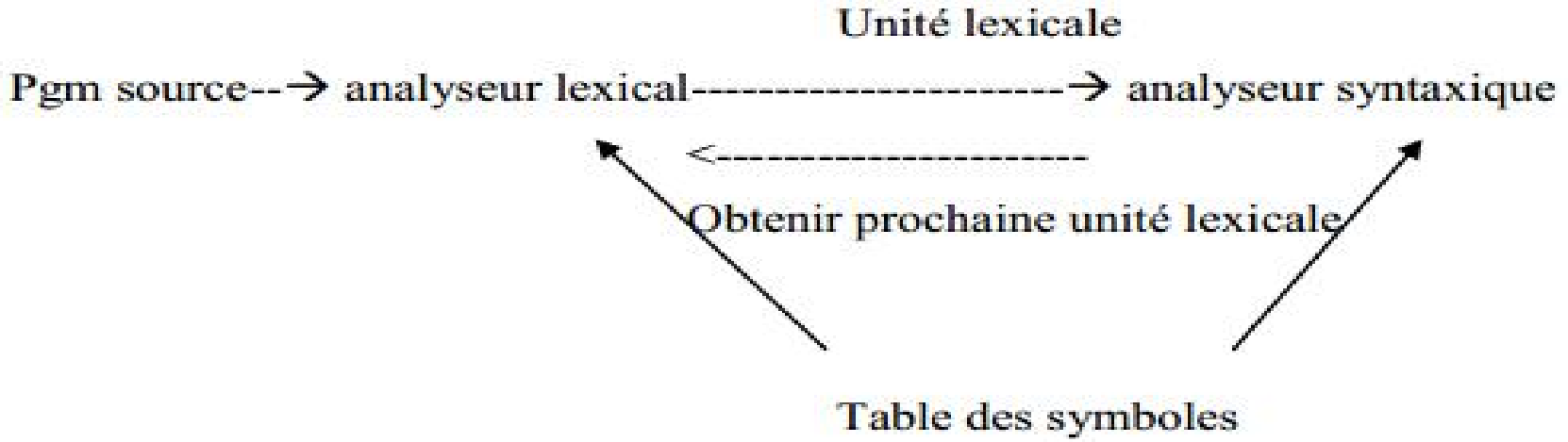
On appelle lexème toute suite de caractère du programme source qui concorde avec le modèle d'une unité lexicale.

Exemple de lexèmes pour unité lexicale IDENT : a ; b ; tot1 ; x_1

Analyse Lexicale

Introduction

- La première phase de compilation ;
- Segmenter le teste du programme source en un ensembles de lexèmes que l'analyseur syntaxique va utiliser.
- Cette interaction est implantée en faisant de l'analyseur lexical un sous programme de l'analyseur syntaxique, à la réception d'une commande « prochaine unité lexicale » émanant de l'analyseur syntaxique, l'analyseur lexical lit les caractères d'entrées jusqu'à ce qu'il puisse identifier la prochaine unité lexicale.



- Débarrasser le programme des commentaires, espacements et tabulations.
- Tenir à jour les coordonnées (ligne et colonne) dans le programme pour fin de production des éventuels messages d'erreur

Analyse Lexicale

Rappels

Alphabet

Un ensemble fini dont les éléments seront appelés lettres.

Exemple : 0,1 sont les lettres de l'alphabet binaire.

Mot sur un alphabet A

Une suite finie d'éléments de A.

Longueur d'un mot

Un mot de longueur n composé des lettres a_1, a_2……., a_n sera noté a_1a_2 …a_n,

Mot vide

Le mot vide est noté ε

Concaténation

La concaténation de deux mots m1 et m2 est notée m1m2

Ensemble des mots sur alphabet

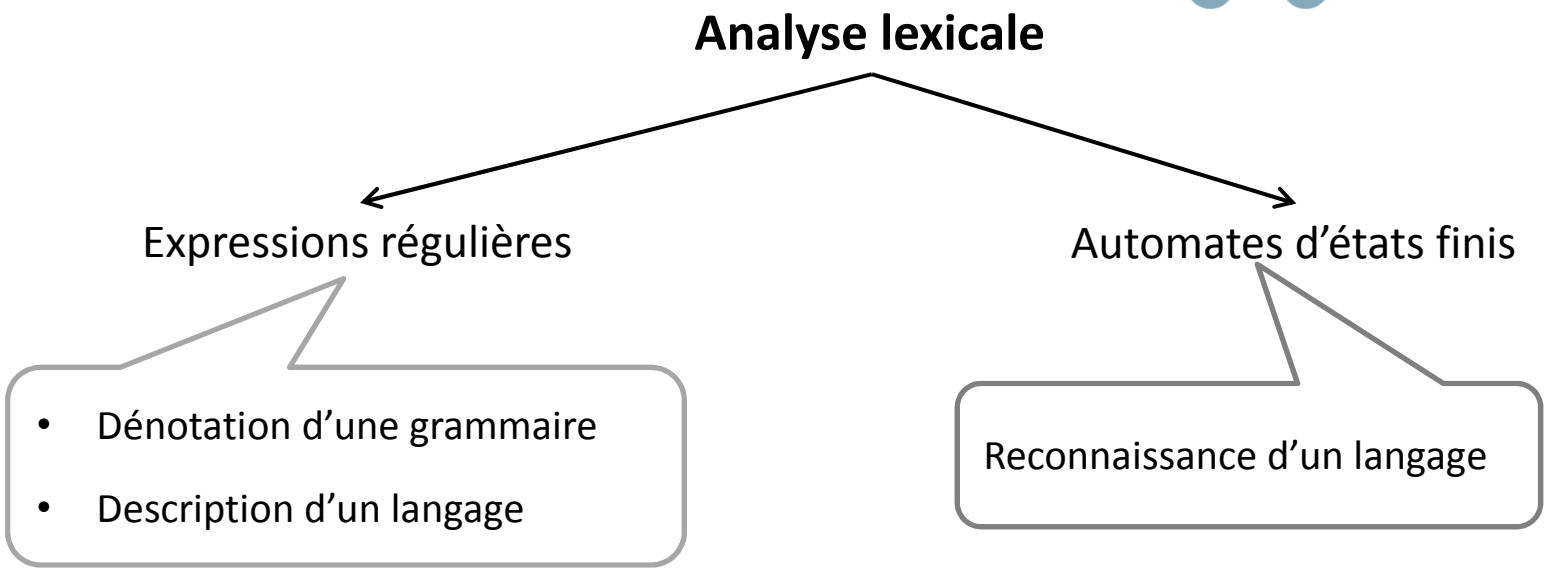
Ensemble des mots sur un alphabet A est noté A*

Langage

Un langage sur un alphabet A est un ensemble de mots de A*

Analyse Lexicale

Réalisation



Analyse Lexicale

Description (1)

Définition

- Une expression régulière est une formule close permettant de désigner un ensemble de chaines de caractères (***Langage***) construites à partir d'un alphabet Σ .
- Une expression régulière est une notation pour décrire un langage régulier.

Soit Σ un alphabet (un ensemble de lettres) :

1. Les éléments de Σ, ε (mot vide) et Ø (langage vide) sont des expressions régulières.
2. **Si** α et β sont des expressions régulières **alors**
   - (α|β), (αβ) et α* sont des expressions régulières.
   - (α|β) représente l'union,
   - (αβ) la concaténation
   - α * la répétition (Concaténations de α avec lui-même ou la chaîne vide).
   - ε est l'élément neutre par rapport à la concaténation
   - Ø (ensemble vide de caractère) neutre par rapport à l'union.

Exemple : L'expressions régulière **a|b*c** dénote l'ensemble des chaînes qui ont soit un seul **a** ou un nombre quelconque; ou nul ; de b suivi par un c.

Analyse Lexicale

Description (2)

Propriétés algébriques

Soit **r,s** et **t** des expressions régulières.

- **r|s = s|r :** l'opérateur **|** (ou) est commutatif.
- **r|(s|t) = (r|s)|t :** l'opérateur **|** est associatif.
- **(rs)t = r(st) :** la concaténation est associative.
- **r(s|t)=rs|rt :**la concaténation est distributive par rapport au |
- e **r = r** e **= r :** e est l'élément neutre de la concaténation.
- **r* =(r|**e**)*** : e est inclus dans une fermeture.
- **r** **= r* :** * est idempotent

Analyse Lexicale

Description (3)

Définition régulière

Une définition régulière est une suite de définitions de la forme:

d1 → r1
d2 → r2
. .
. .
dn → rn

Chaque **di** est un nom distinct et chaque **ri** est une expression régulière sur l'alphabet :

Exemple

La définition régulière d'un identificateur :

Lettre → A|B|...|Z|a|b|...|z
Chiffre → 0|1|2...|9
Id → Lettre(Lettre|Chiffre)*

Analyse Lexicale

Exemple

A partir de l'énoncé suivant : **for i :=1 to vmax do a :=a+i;**

Suite de lexèmes

for : mot clé	**to** : mot clé	**:=** : affectation	**;** : séparateur
i : identificateur	**vmax** : identificateur	**a** : identificateur	
:= : affectation	**do** : mot clé	**+** : opérateur arithmétique	
1 : entier	**a** : identificateur	**i** : identificateur	

Table des symboles

Numéro de Symbole	Token	Type de Token
10	for	Mot clé
11	to	Mot clé
12	do	Mot clé
13	;	Séparateur
…	…	
100	:=	Affectation
101	+	Opération
	…	
1000	i	identificateur
1001	a	identificateur
1002	vmax	identificateur
…		
5001	1	entier

Analyse Lexicale

Reconnaissance (1)

Automate d'état finis

- Un reconnait un langage par un programme qui prend en entrée une chaine **x** et répond oui si **x** est un mot du langage et non autrement.
- On compile une expression régulière en un reconnaisseur en construisant un diagramme de transition généralisé appelé ***automate d'état fini.***

> ***Théorème***
>
> Un langage régulier est reconnu par un automate fini.

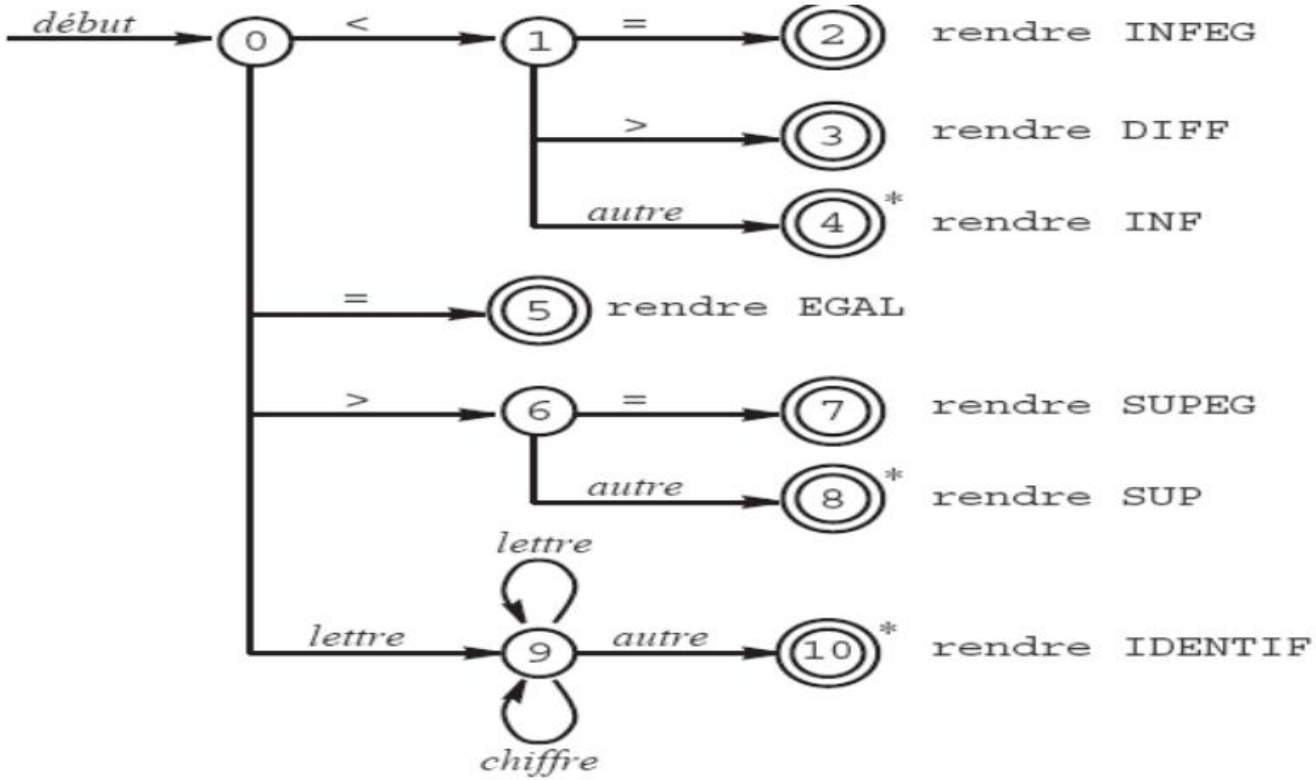
- Un bon langage régulier (et donc une ER) peut être reconnu par un automate.
- Donc, pour écrire un analyseur lexical de notre programme source, il suffit de décrire un programme simulant l'automate.
- Lorsqu'une unité lexicale est reconnue, elle est envoyée à l'analyseur syntaxique qui la traite puis repasse la main à l'analyseur lexical qui lit l'unité lexicale suivante dans le programme source.
- Et ainsi de suite, jusqu'à tomber sur une erreur ou jusqu'à ce que le programme source soit traité en entier (et alors on est content).

Analyse Lexicale

Reconnaissance (2)

Exemple

Reconnaissance des unités lexicales INFEG, DIFF, INF, EGAL, SUPEG, SUP, IDENTIF, respectivement définies par les expressions régulières <=, <>, <, =, >=, > et lettre(lettre/chiffre)*.



Analyse Lexicale

Automates d'états finis (3)

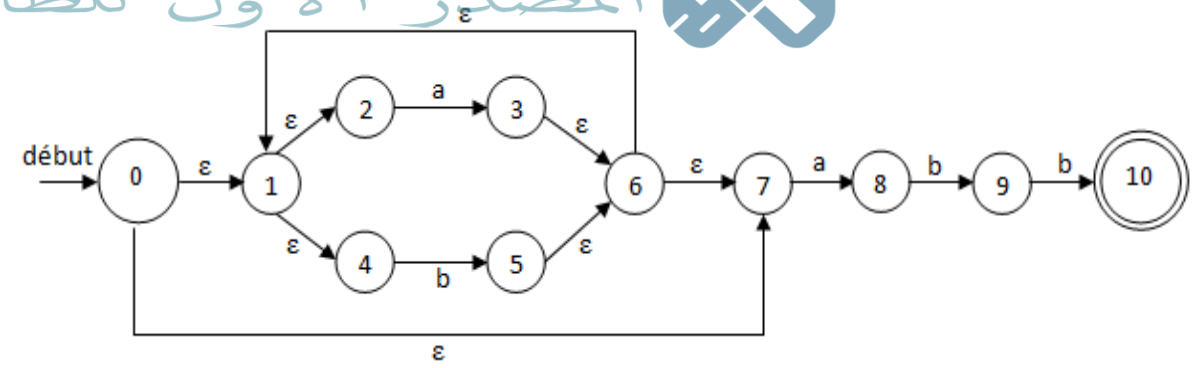
Exemple

ER= (a|b)*abb

- Plusieurs arcs reconnaissant le même caractère peuvent « sortir » du même état.
- Existence de transitions, notées ε, qui ne correspondent à aucune reconnaissance.

Automate d'état fini

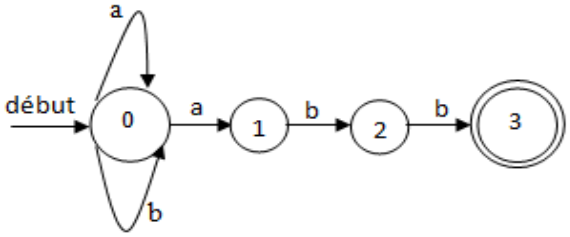
Non Déterministe (AFND)



Un AFND est un quintuple $A=(\mathbf{Q}, \sum, \delta, q_0, \mathbf{F})$, où

- $\mathbf{Q}= \{0,1,2,3,4,5,6,7,8,9,10\}$: Ensemble fini d'états
- $\sum = \{\varepsilon, a, b, c\}$: Alphabet
- $\delta : \mathbf{Q} \times \sum \cup \{\varepsilon\} \rightarrow \mathbf{Q}$: Fonction de transition
- $q_0 = \{0\}$: État initial
- $\mathbf{F}= \{0\}$: Ensemble des états terminaux ou finaux

Déterministe (AFD)



Analyse Lexicale

Démarche Générale de construction d'un analyseur lexical

En se basant sur les propositions démontrables suivantes :

- L'ensemble des unités lexicales d'un langage donné constitue un langage régulier **L**.
- ∀ **r** expression régulière, ∃ un AFND qui accepte l'ensemble régulier décrit par **r** ;
- **Si** un langage **L** est accepté par un AFND **Alors** il existe un AFD acceptant **L**.

Etape 1 : Spécification des unités lexicales

Spécifier chaque type d'unité lexicale à l'aide d'une expression régulière **r**.

Etape 2 : Conversion de chaque **r** en AFND

Etape 3 : Réunion des AFNs

Construire l'automate Union de tous les automates de l'étape 2 (on peut ajouter un nouvel état initial d'où partent un ensemble d'arcs étiquetés ε) .

Etape 4 : Détermination ou transformation de l'AFND obtenu en AFD

Rendre l'automate de l'étape 3 déterministe.

Etape 5 : Minimisation de l'AFD.

Etape 6 : Implémentation de l'AFD minimisé en le simulant à partir de sa table de transition.

Analyse Lexicale

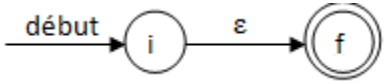
Démarche Générale de construction d'un analyseur lexical

Étape 2 : Construction d'un AFND à partir d'une expression régulière

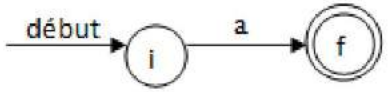
Expression Régulière **(r)** → Algorithme de **Thompson** → AFND **N(r)**

Les règles de l'algorithme de Thompson sont les suivantes :

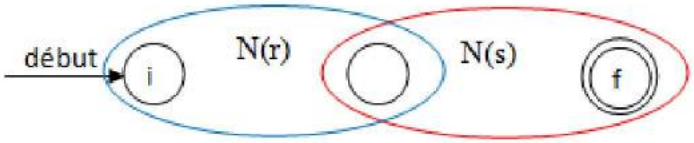
1. Pour r = ε N(ε) est représenté par :



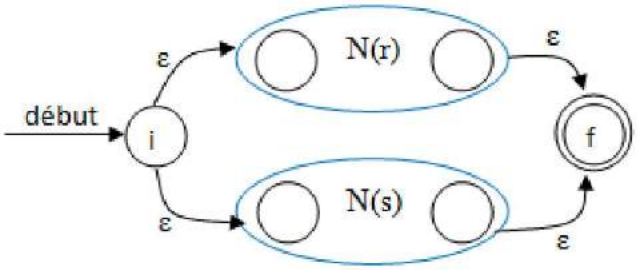
2. Pour r = a N(a) est représenté par :



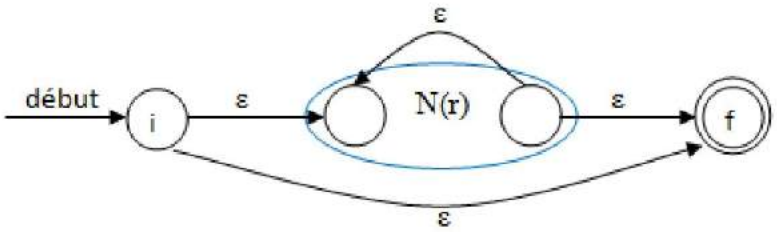
3. Pour N(rs) on a :



4. Pour N(r|s) on a :



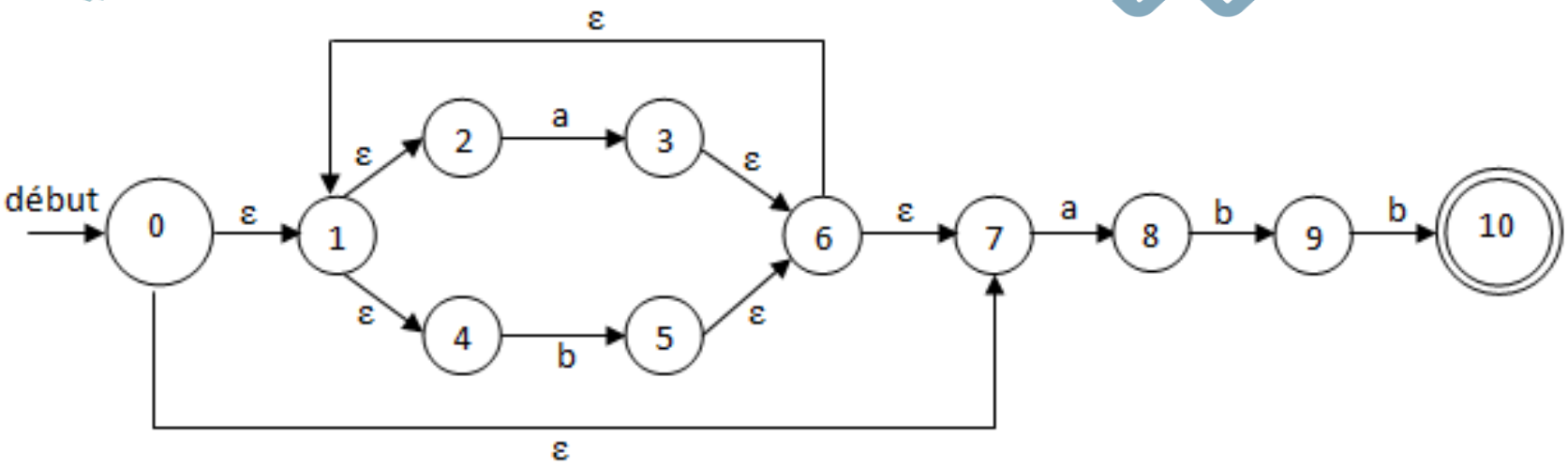
5. Pour r* , N(r*) sera donnée par la représentation suivante :



Analyse Lexicale

Démarche Générale de construction d'un analyseur lexical (2)

L'AFND qui reconnaît l'expression régulière (a|b)*abb selon la méthode de construction de Thompson est :



Analyse Lexicale

Démarche Générale de construction d'un analyseur lexical (3)

Étape 03 : :Détermination d'un AFND

La détermination consiste à transformer un AFND en un AFD équivalent. Elle peut être effectuée par différentes méthodes.

Etats	Symboles	
	a	b
0	{0,2}	{1}
1	{3}	{0,2}
2	{3,4}	{2}
3	{2}	{1}
4	/	{3}

1. **Méthode 1 :** Détermination d'un AFN ne contenant pas de ε-transitions

- Étape 01 : A Partir de l'état initial $E_0 = \{e_0\}$
- Étape 02 : Construire E_1 qui est l'ensemble des états obtenus à partir de E_0 par la transition étiquetée a, E_1 = Transiter(E_0,a).
- Étape 03 : Recommencer l'étape 2 pour toutes les transitions possibles et pour chaque nouvel ensemble d'états E_i.
- Étape 04 : Tous les ensembles d'états E_i contenant au moins un état final deviennent finaux.
- Étape 05 : Renuméroter les ensembles d'états obtenus en tant que simples états.

Etats	Symboles	
	a	b
{0}	{0,2}	{1}
{0,2}	{0,2,3,4}	{1,2}
{1}	{3}	{0,2}
{0,2,3,4}	{0,2,3,4}	{1,2,3}
{1,2}	{3,4}	{0,2}
{3}	{2}	{1}
{1,2,3}	{2,3,4}	{0,1,2}
{3,4}	{2}	{1,3}
{2}	{3,4}	{2}
{2,3,4}	{2,3,4}	{1,2,3}
{0,1,2}	{0,2,3,4}	{0,1,2}
{1,3}	{2,3}	{0,1,2}
{2,3}	{2,3,4}	{1,2}

Analyse Lexicale

Démarche Générale de construction d'un analyseur lexical (4)

Après renumérotation, la table de l'AFD devient :

Etats	Symboles	
	a	**b**
A	B	C
B	D	E
C	F	B
D	D	G
E	H	B
F	I	C
G	J	K
H	I	L
I	H	I
J	J	G
K	D	K
L	M	K
M	J	E

Analyse Lexicale

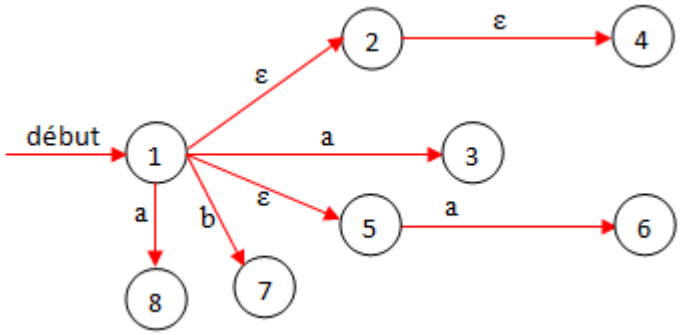
Démarche Générale de construction d'un analyseur lexical (4)

Méthode 02 : méthode de construction de sous-ensembles pour la transformation des AFND en AFD

Notation et opérations utilisées : Soit un AFND noté N et **D** son AFD équivalent

- DTrans est la table des transitions de **D**
- Détats est l'ensemble des états de **D**
- **e** est un état de **N**, $\mathbf{e_0}$ est l'état initial de **N**
- **T** est un ensemble d'états de **N**
- L'opération **ε-fermeture(e)** est l'ensemble des états de **N** accessibles à partir de l'état e par des ε-transitions uniquement.
- Un état est accessible à partir de lui-même par une ε-transition (même si l'arc n'est pas visible)
- L'opération **ε-fermeture(T)** donne l'ensemble des états de N accessible à partir des états e (**e**∈ **T**) par des ε-transitions uniquement.
- L'opération **Transiter(T,a)** donne l'ensemble des états de **N** vers lesquels il existe une transition avec le symbole d'entrée **a** à partir des états **e** ∈ **T**.

Exemple :



- ε-fermeture(1) = {1, 2, 4,5}
- ε-fermeture({2, 5}) = {2, 4, 5}
- ε-fermeture({3,6}) = {}
- Transiter({1, 5},a) = {3, 8, 6}
- Transiter({1},b) = {7}
- Transiter({2},b) = {}

Analyse Lexicale

Démarche Générale de construction d'un analyseur lexical (5)

Algorithme ConstructionSousEnsembles
Début
 ε-fermeture(e0) est l'unique état de Détats et il est no marqué ;
 Tant qu' il existe un état T non marqué dans Détats Faire
  Marque T
  Pour chaque symbole d'entrée a **Faire**
   U← ε−fermeture(Transiter(T,a))
   Si U∉Détats **Alors**
    Ajouter U comme nœud non marqué dans Détats
   Finsi
   Dtransit[T,a] ← U ;
  Finpour
 FinTantque
Fin

Analyse Lexicale

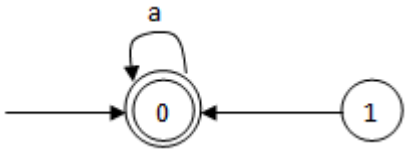
Démarche Générale de construction d'un analyseur lexical (5)

Étape 05 : Minimisation d'un AFD

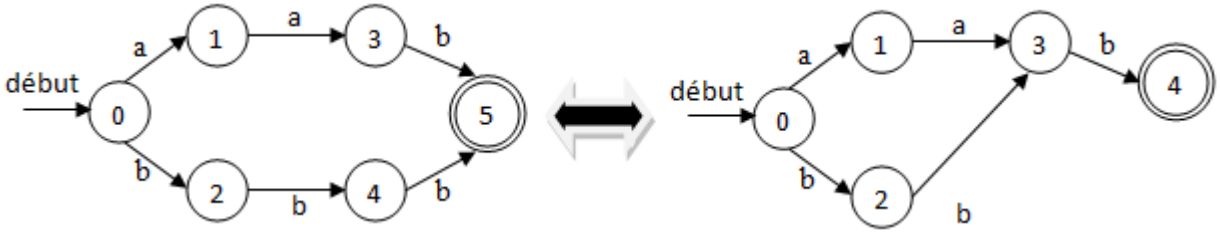
La minimisation d'un AFD consiste en deux opérations principales :

- Elimination de tous les états inaccessibles de l'AFD
- Regroupement des états équivalents en les remplaçant par des classes d'équivalence d'états sachant que :
- Un état e est accessible s'il existe une chaine menant à e à partir de l'état initial. Un état, autre que l'état initial, est inaccessible lorsqu'aucun arc n'arrive sur cet état dans un chemin provenant de l'état initial.

Exemple : dans l'automate l'état 1 est inaccessible.



On dit que deux états sont équivalents si ces états permettent d'atteindre un état final à travers la même chaine. En d'autres termes, tous les suffixes reconnus à partir de ses états sont exactement les mêmes.



Analyse Lexicale

Démarche Générale de construction d'un analyseur lexical (5)

Étape 06 : Simulation d'un AFD

Soit une chaine x représentée par un fichier de caractères se terminant par EoF qui est un caractère spécial indiquant la fin d'un fichier. Soit un AFD D avec un état initial e_0 et un ensemble d'états finaux F. soit la fonction Transiter (e,ch) qui donne l'état vers lequel il y a transition à partir de l'état e avec le caractère ch.

```
Algorithme SimulAFD
Début
    e ← e0
    carsuiv(ch);
    Tantque ch≠ EoF Faire
      Début
        e ←Transiter (e,ch);
        carsuiv (ch);
      Fin Tantque
    Si e ∉ F Alors
        accepter (x);
    Sinon
        rejeter (x);
    Fin
Fin
```

Fin de la présentation

Merci pour votre attention