Université Hassan II De Casablanca
Faculté des sciences AIN Chock
Département des mathématiques et informatique

Année universitaire 2024/2025
Filière SMI
Semestre S5

Module Compilation

TD 1 : Expressions régulières et automates finis

Questions de cours :

1) Citer les différentes phases de la compilation ? quel est l'entré et la sortie de chaque étape.

Phase d'analyse :
- Analyse lexicale
- Analyse syntaxique
- Analyse sémantique

Phase de synthèse :
- Génération de code intermédiaire
- Optimisation du code
- Génération du code machine

Analyse lexicale (Lexing) :
- **Entrée :** Code source (texte brut du programme)
- **Sortie :** Suite de tokens (unités lexicales)

Analyse syntaxique (Parsing) :
- **Entrée** : Suite de tokens
- **Sortie** : Arbre syntaxique abstrait (AST)

Analyse sémantique :
- **Entrée :** Arbre syntaxique abstrait (AST)
- **Sortie :** Arbre syntaxique annoté / erreurs sémantiques

Génération de code intermédiaire :
- **Entrée** : Arbre syntaxique annoté ou code source
- **Sortie** : Code intermédiaire (code simplifié, indépendant de la machine cible)

Optimisation de code (facultative) :
- **Entrée :** Code intermédiaire
- **Sortie :** Code intermédiaire optimisé

Génération de code machine :
- **Entrée :** Code intermédiaire
- **Sortie :** Code machine ou code objet

Édition de liens (Linking) :
- **Entrée :** Un ou plusieurs fichiers objets
- **Sortie :** Programme exécutable

2) Quel est le rôle de la table de symbole ?

La table des symboles est une structure de données utilisée par les compilateurs et interpréteurs pour stocker des informations sur les identificateurs (noms de variables, fonctions, classes, etc.) rencontrés lors de l'analyse d'un programme. Elle joue un rôle clé dans plusieurs étapes du processus de compilation, notamment l'analyse lexicale, l'analyse syntaxique et l'analyse sémantique.

Rôle dans les étapes de la compilation :

Analyse lexicale : Identifie les symboles du programme source et enregistre leurs premières informations dans la table.

Analyse syntaxique : Vérifie que les symboles sont utilisés correctement en fonction de la grammaire du langage.

Analyse sémantique : Vérifie que les opérations appliquées aux symboles sont cohérentes en fonction de leur type (par exemple, additionner deux entiers).

Génération de code : Utilise les informations de la table des symboles pour allouer de la mémoire et générer le code machine ou intermédiaire.

3) Quelle est la différence entre une expression régulière et une automate ?

Les expressions régulières sont des notations compactes (modèles) pour décrire le langage, tandis que **les automates** sont des modèles qui acceptent ou rejettent des chaînes en fonction de leurs états et de leurs transitions. Ils sont utilisés pour faciliter le processus de reconnaissance de mots (on exprime une expression régulière sous forme d'un automate fini pour faciliter son implémentation).

Les automates et les expressions régulières jouent un rôle similaire en ce sens qu'ils engendrent des langages, mais ils diffèrent par leur représentation et leur utilisation. Les expressions régulières sont principalement utilisées pour rechercher des motifs dans un texte, tandis que les automates servent à vérifier si un mot appartient à un langage donné.

4) Dans quelles conditions peut-on dire qu'un automate est déterministe ?

Pas d'états inaccessibles
Pas de transition multiple
Pas de transition epslion
Un seul état initial

5) Qu'est-ce qu'un état puit dans les automates ?

Un **état puits** (ou **état absorbant**) dans les automates, est un état d'un automate à partir duquel **toutes les transitions** mènent à lui-même.

6) Comment définir l'ε-fermeture d'un état ?

C'est l'ensemble construit par l'état et les états accessibles par des transitions epslion.

Exercice I :

Donner l'expression régulières des langages suivants :

1) Le langage L des mots qui commencent par a et se terminent par bc sur V= {a, b, c}.

a .X. bc => tel que peut prendre n'importe quel mot
Donc : e(L) = a.(a|b|c)*.bc

2) Langage L des mots binaires qui acceptent la division sur 2 et n'admettent pas des 0 unités au début.

V = {0,1}

1(0|1)*0 : Langage des nombres binaires divisible par 2 et qui n'accepte pas un 0 au début.

Donc : e(L) = 1(0|1)*0 + 0

3) Le langage L des nombres naturels signés.

V = {+, -, 0, 1 , 2, 3, 4, 5, 6, 7, 8, 9}

Donc : e(L) = (+|-) ?[0-9]$^{+}$

Ou pour éviter les zéros au début :

e(L) = (+|-) ?([1-9][0-9]*| 0)

4) Des entiers naturels acceptant la division sur 5.

V = {0, 1 , 2, 3, 4, 5, 6, 7, 8, 9}

e(L) = [0-9]*(0+5)

Ou pour éviter les zéros au début :

e(L) = [1-9][0-9]*(0+5) | (0+5)

5) Des identificateurs sous le langage C.

V = {A,…Z,a...b,0..9,_}

e(L) = ([A-Z] + [a-z] + _) ([A-Z] + [a-z] + [0-9] + _) *

Ou :

e(L) = ([A-Za-z _]) ([A-Za-z0-9_]) *

6) Des nombres entiers appartenant à l'intervalle [-999,1000].

L1 : les entiers entre -999 et -1

L2 : les entiers entre 1 et 999

Et on ajoute 0 et 1000.

e(L) = **(** (-) ? **(** [1-9] [0-9] [0-9] + [1-9] [0-9] + [0-9]**)** **)** | 1000

7) Le langage binaire (ne pas commencer par un 0)

e(L) = 1(0|1)*

8) Des mots qui comportent au moins deux occurrences de la sous chaines 'ab' sur V={a, b , c}

e(L) = [a-c]*ab[a-c]*ab[a-c]*

9) Reconnaître une date au format JJ/MM/AAAA

Pour les jours :

- 0[1-9] : jours de 01 à 09
- [12][0-9] : jours de 10 à 29
- 3[01] : jours 30 ou 31 (selon les mois, mais cette validation ne s'occupe pas de cela).

Pour les mois :

- 0[1-9] : mois de 01 à 09
- 1[0-2] : mois de 10 à 12.

Pour l'année :

- [0-9] : quatre chiffres pour l'année (de 0000 à 9999).

e(L) = 0[1-9]+[12][0-9]+3[01]/0[1-9]+1[0-2]/[0-9] [0-9] [0-9] [0-9]

10) Écrivez une expression régulière pour correspondre à un numéro de téléphone portable marocain.

e(L) = (+212|00212|0)(6|7)[0-9] [0-9] [0-9] [0-9] [0-9] [0-9] [0-9] [0-9]

11) Écrivez une expression régulière qui reconnaît des nombres décimaux positifs ou négatifs, avec ou sans partie fractionnelle.

e(L) = (-|+) ?[0-9]$^{+}$(.[0-9]$^{+}$) ?

Sans zéros non significatifs : (+/-)?([0-9]+[1-9][0-9]*)(.([0-9]+[0-9]*[1-9]))?

12) Écrivez une expression régulière qui génère tous les mots sur l'alphabet {a, b} qui commencent et finissent par la même lettre.

e(L) = (a(a|b)*a)|(b(a|b)*b)

13) Écrivez une expression régulière qui génère tous les mots sur l'alphabet {a, b} contenant un nombre pair de a.

e(L) = (b*a b*a b*)*|b*

ou

e(L) = (b*(ab*ab*)*)

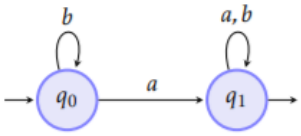
14) Écrivez une expression régulière pour valider une adresse IPv4. Une adresse IP est composée de quatre nombres séparés par des points, chaque nombre doit être compris entre 0 et 255.

- ([1-9]?[0-9]) correspond aux nombres de 0 à 99.
- (1[0-9][0-9]) correspond aux nombres de 100 à 199.
- (2[0-4][0-9]) correspond aux nombres de 200 à 249.
- (25[0-5]) correspond aux nombres de 250 à 255.

e(L) = ([1-9]?[0-9])|(1[0-9][0-9])|(2[0-4][0-9])|(25[0-5]). ([1-9]?[0-9])|(1[0-9][0-9])|(2[0-4][0-9])|(25[0-5]). ([1-9]?[0-9])|(1[0-9][0-9])|(2[0-4][0-9])|(25[0-5]). ([1-9]?[0-9])|(1[0-9][0-9])|(2[0-4][0-9])|(25[0-5])

Exercice II

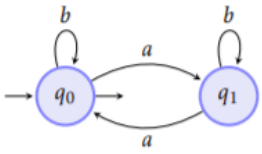
Retrouver les expressions régulières des langages à partir des automates suivants :



1)

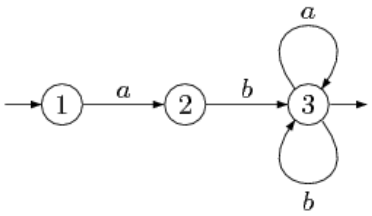
V= {a,b}
e(L) = b*a(a+b)*

2)



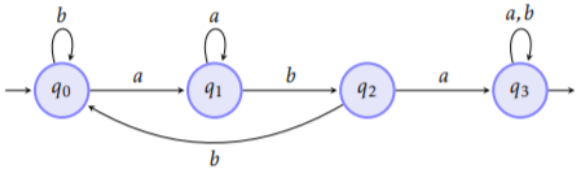
V = {a,b}
e(L) = (b*ab*a)* | b*
ou
e(L) = (b+ab*a)*

3)



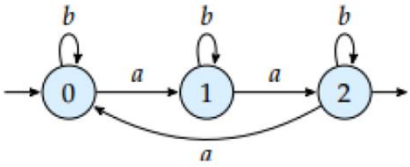
e(L) = ab(a+b)*

4)



e(L) = (b*a⁺b) .(bb*a⁺b)*. a(a|b)*

5)



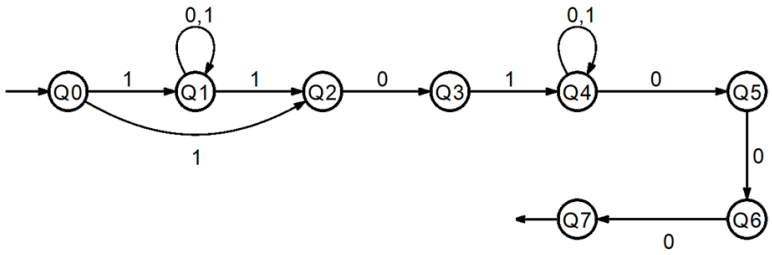
e(L) = b*ab*ab* (a b*ab*ab*)*

Exercice III

Donner l'expression régulière et l'automate fini engendré par les langages :

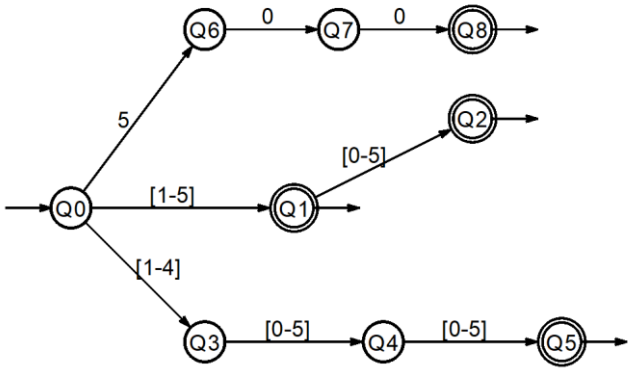
1) L1 le langage des mots binaires qui commencent par 1, incluent au moins une occurrence de la chaine 101 et acceptent la division sur 8.

V={0,1}
L11 : Les mots qui commencent par 1 : 1 X
L12 : Les mots qui contiennent 101 : X 101 X
L13 : Les mots divisibles par 8 : X 000
e(L1) = ((1(0+1)*101) | 101)(0+1)*000
Ou
e(L1) = (1(1|0)*)?101(1|0)*000



2) L2 le langage des nombres naturels appartenant à l'intervalle [1,500] sur V={0,1,2,3,4,5}.

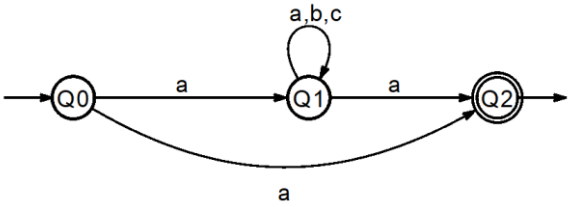
L21 : les entiers qui se compose d'un seul chiffre : [1-5]
L22 : les entiers qui se composent de deux chiffres [1-5][0-5]
L23 : de trois chiffres : [1-4][0-5][0-5] | 500
e(L2) = [1-5] + [1-5][0-5] + [1-4][0-5][0-5] + 500



3) L3 le langage des mots qui commencent par a et ne se terminent pas par b ou c sur V={a,b,c}

V = {a,b,c}
L31 : Mots qui comment par aX
L32 : Ne se termine pas par b ou c : Xa|ε

e(L3) = a | a(a+b+c)*a



4) L4 le langage des mots binaires dont chaque 1 doit être suivi de 0.

e(L4) = $(0 \mid 10)^+$