Université Mohamad kheider- Biskra
Département d'informatique
Module : Compilation
Niveau : L3 **Date : 29/10/2024**

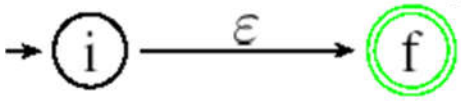
TP1 : Construction d'un AFN à partir d'une expression régulière

Il existe de nombreuses stratégies pour construire de façon automatique un automate fini à partir d'une expression régulière. Parmi ces stratégies, on va réaliser dans ce TP l'algorithme suivant :

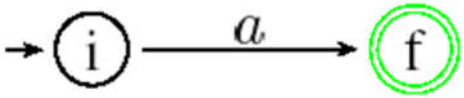
A partir d'une expression **r** sur un alphabet **A**, on cherche à construire un AFN **T(r)** qui reconnaît le langage reconnu par **r**.

1. Si l'expression est réduite à un symbole **s** alors on construit l'automate :

- Cet automate reconnaît alors le langage {ε}.

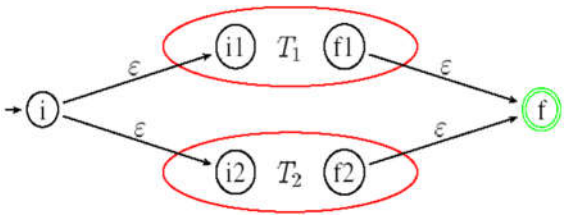


- Cet automate reconnaît alors le langage {a}.

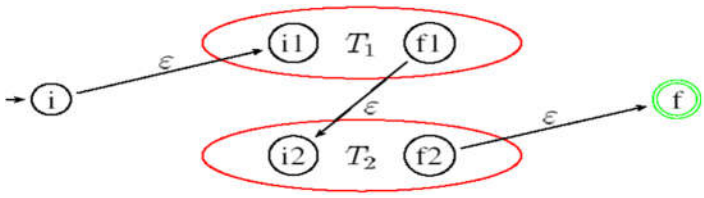


2. Si l'expression n'est pas un symbole, il faut alors la décomposer selon l'opérateur utilisé puis construire les automates **T(r_1)** et **T(r_2)** associés aux opérandes r_1 et r_2. Enfin, selon le type d'expression :

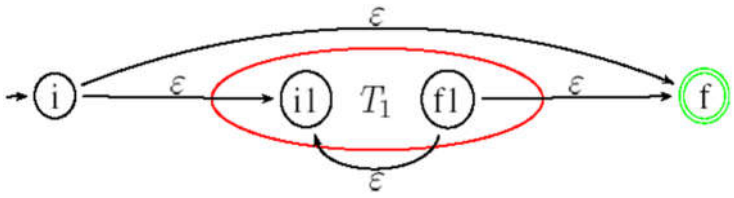
- $r_1|r_2$: on construit l'automate $T(r_1|r_2)$ avec un nouvel état de départ(i) et un nouvel état final (f). Il y a une ε-transition depuis (i) vers les états de départ de $T(r_1)$ et $T(r_2)$ et depuis les états finaux de $T(r_1)$ et $T(r_2)$ vers (f).



- r_1r_2 : on construit l'automate $T(r_1r_2)$ avec un nouvel état de départ(i) et un nouvel état final (f). Il y a une ε-transition depuis (i) vers l'état de départ de $T(r_1)$ et depuis l'état final de $T(r_1)$ vers l'état initial de $T(r_2)$ et depuis l'état final de $T(r_2)$ vers (f). L'état final de $T(r_1)$ et celui de départ de $T(r_2)$ peuvent être fusionnés.



- r_1* : on construit l'automate T(r_1*) avec un nouvel état de départ(i) et un nouvel état final (f) .On peut aller soit directement de (i) à (f) en suivant une ε-transition qui représente le fait que ε ∈ L(r_1*) soit aller de (i) à (f) en traversant T(r_1) une ou plusieurs fois.



3. Avant d'appliquer l'algorithme, il faut construire tout d'abord l'arbre associé à l'expression régulière.

Travail Demandé

1. Ecrire une fonction C : **CreateArbre** permettant de présenter une expression régulière sous forme d'un arbre correspond. Vous pouvez représenter l'arbre avec une liste où chaque maillon a quatre champs : chaine de caractères qui représente l'expression, un pointeur vers fils gauche (opérande 1), un pointeur vers fils droit (opérande 2) et caractère qui représente l'opérateur. Si l'opérateur est *, il a donc un seul opérande. L'opérateur de concaténation est représenté par un point.
2. Ecrire une fonction C : **GenererSym** qui permet de générer un automate à partir d'un seul symbole passé en paramètre.
3. Ecrire une fonction C : **GenererUnion** qui permet de générer un automate qui représente l'union de deux automates passés en paramètre.
4. Ecrire une fonction C : **GenererConc** qui permet de générer un automate qui représente la concaténation de deux automates passés en paramètre.
5. Ecrire une fonction C : **GenererEtoile** qui permet de générer un automate qui représente l'étoile de Kleene de l'automate passé en paramètre.
6. Ecrire une fonction C : **ConvertExpAut** qui permet de parcourir récursivement l'arbre d'une expression régulière passée en paramètre afin de construire l'automate qui reconnaît le langage reconnu par cette expression.
7. Ajouter la fonction **main** pour tester ces fonctions.

Bon Courage