

Calculabilité - Décidabilité (ICC)

Cours n°2

Stef Graillat

Sorbonne Université



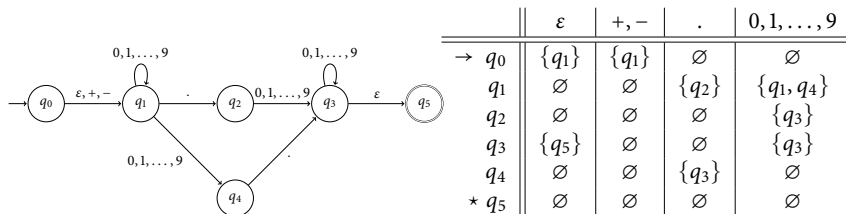
Résumé du cours précédent

- *Automate fini déterministe* : un AFD a un ensemble fini d'**états** et un ensemble fini de **symboles d'entrées**. Un état est désigné comme l'**état initial** et zéro ou plusieurs états sont désignés comme **états finaux**. Une **fonction de transition** détermine comment on passe d'un état à un autre lorsqu'on lit un symbole.
- *Diagramme de transition* : permet de représenter un automate par un graphe
- *Langage associé à un automate* : un mot est accepté si en partant de l'état initial, on arrive à un état final en lisant un par un les symboles du mot
- *Automate fini non-déterministe* : un AFN diffère d'un AFD en ce qu'un AFN peut avoir un nombre quelconque de transition en partant d'un état donné et en lisant un même symbole donné
- **ϵ -transitions** : elles permettent d'étendre un AFN en autorisant un changement d'état en lisant une entrée vide (c'est-à-dire en ne lisant aucun symbole).

Notations pour un ε -AFN

Un ε -AFN est un quintuplet $(Q, \Sigma, \delta, q_0, F)$ où $\delta : Q \times (\Sigma \cup \{\varepsilon\}) \rightarrow \mathcal{P}(Q)$

Exemple : $E = (\{q_0, q_1, \dots, q_5\}, \{\cdot, +, -, 0, 1, \dots, 9\}, \delta, q_0, \{q_5\})$



Définir des fonctions de transition étendues pour définir le langage reconnu.

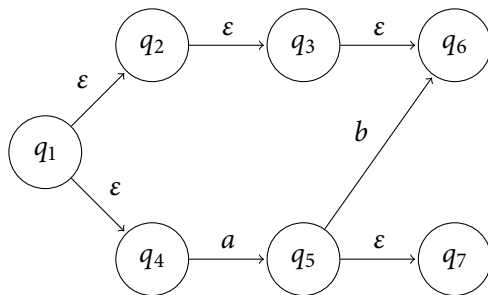
Nécessité d'une notion de fermeture d'un état par ε .

Idée : Suivre l'évolution de l'automate (d'un état) par les ε -transitions. On « ferme » un état en lui ajoutant tous les états atteignables par les mots $\varepsilon\varepsilon\cdots\varepsilon$.

Définition récursive de la ε -fermeture $ECLOSE(q)$ pour $q \in Q$

- q est dans $ECLOSE(q)$
- Si p est dans $ECLOSE(q)$ et qu'il existe une transition de p à r par ε ($r \in \delta(p, \varepsilon)$) alors r est dans $ECLOSE(q)$.

Exemple



$$\text{ECLOSE}(q_1) = \{q_1, q_2, q_3, q_4, q_6\}$$

Transitions étendues et langages

Soit $E = (Q, \Sigma, \delta, q_0, F)$ un ε -AFN, $q \in Q$ et $w \in \Sigma^*$.

On étend la fonction de transition en suivant les chemins dont la concaténation donne w . Importance de la fermeture!

Transition étendue définie par récurrence :

- $\widehat{\delta}(q, \varepsilon) = \text{ECLOSE}(q)$
- Si $w = xa$ avec $a \in \Sigma$ et $\widehat{\delta}(q, x) = \{p_1, \dots, p_k\}$. Soit

$$\bigcup_{i=1}^k \delta(p_i, a) = \{r_1, \dots, r_m\}$$

$$\text{Alors } \widehat{\delta}(q, w) = \bigcup_{i=1}^m \text{ECLOSE}(r_i)$$

Le langage associé à E est

$$L(E) = \{w \in \Sigma^* \mid \widehat{\delta}(q_0, w) \cap F \neq \emptyset\}$$

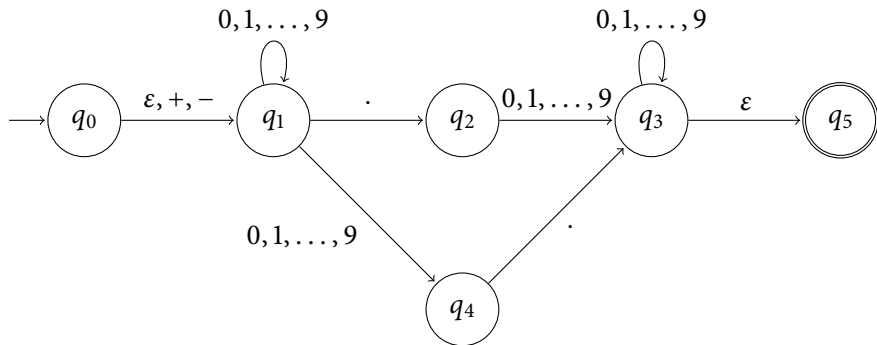
Éliminer les ε -transitions

Étant donné un ε -AFN E on peut définir un AFD D reconnaissant $L(E)$

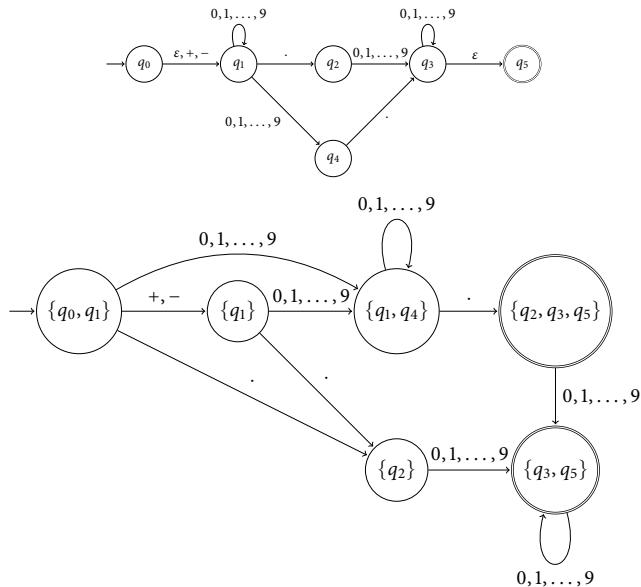
Soit $E = (Q_E, \Sigma, \delta_E, q_0, F_E)$ un ε -AFN, on définit l'AFD $D = (Q_D, \Sigma, \delta_D, q_D, F_D)$ par :

- Q_D est l'ensemble des sous-ensembles S de Q_E (vérifiant $S = \text{ECLOSE}(S)$)
- $q_D = \text{ECLOSE}(q_0)$
- $F_D = \{S \in Q_D \mid S \cap F_E \neq \emptyset\}$
- $\delta(S, a)$ est calculé comme suit :
 - Soit $S = \{p_1, \dots, p_k\}$
 - Calculer $\bigcup_{i=1}^k \delta_E(p_i, a) = \{r_1, \dots, r_m\}$
 - $\delta_D(S, a) = \bigcup_{j=1}^m \text{ECLOSE}(r_j)$

Exemple



Exemple (suite)



Théorème 1

Un langage L est accepté par un ε -AFN si et seulement s'il est accepté par un AFD.

Preuve. On utilise la construction de D et on montre par induction sur la longueur des mots que $\widehat{\delta}_E(q_0, w) = \widehat{\delta}_D(q_D, w)$

Base : $\widehat{\delta}_E(q_0, \varepsilon) = \text{ECLOSE}(q_0) = q_D = \widehat{\delta}_D(q_D, \varepsilon)$

Induction :

$$\begin{aligned}\widehat{\delta}_E(q_0, xa) &= \bigcup_{p \in \widehat{\delta}_E(q_0, x), a} \text{ECLOSE}(p) \\ &= \bigcup_{p \in \widehat{\delta}_D(q_D, x), a} \text{ECLOSE}(p) \\ &= \bigcup_{p \in \widehat{\delta}_D(q_D, xa)} \text{ECLOSE}(p) = \widehat{\delta}_D(q_D, xa)\end{aligned}$$

- AFD, AFN et ε -AFN permettent de définir des langages réguliers (langages acceptés).
→ description via le comportement d'une machine
- Expressions régulières : définies indépendamment d'une machine (nouveau type de définition d'un langage).
- Relation entre expressions régulières et AFD : les expressions régulières définissent les langages réguliers et ceux-ci seulement!
- Application : étant donnée une ER, construire un AFD reconnaissant le langage qu'il définit.
- Description déclarative du langage utilisée dans tout système gérant des chaînes de caractères (grep, Lex, Flex).

Opérations sur les langages

Opérateurs sur les langages :

- **union** de deux langages :

$$L \cup M = \{w : w \in L \text{ ou } w \in M\}$$

- **concaténation** de deux langages :

$$L.M = \{w : w = xy, x \in L, y \in M\}$$

- **puissance** :

$$L^0 = \{\varepsilon\}, \quad L^1 = L, \quad L^{k+1} = L.L^k$$

- **clôture** d'un langage :

$$L^* = \bigcup_{i=0}^{\infty} L^i$$

Construction des expressions régulières

Une ER définit un langage à partir de constantes et d'opérateurs

Base :

- Les constantes ε et $\emptyset$ représentent les langages $\{\varepsilon\}$ et $\emptyset$
- Si $a \in \Sigma$, $\mathbf{a}$ représente $\{a\}$
- Une variable en lettres majuscules représente un langage.

Induction :

- Si E et F sont deux ER, $E + F$ est une ER telle que $L(E + F) = L(E) \cup L(F)$.
- Si E et F sont deux ER, EF est une ER telle que $L(EF) = L(E)L(F)$
- Si E est une ER, E^* est une ER telle que $L(E^*) = L(E)^*$.
- Si E est une ER, (E) est une ER telle que $L((E)) = L(E)$.

Exemple

Une ER pour $L = \{w \in \{0,1\}^* : 0 \text{ et } 1 \text{ alternent dans } w\}$ est

$$(01)^* + (10)^* + 0(10)^* + 1(01)^*$$

ou encore

$$(\varepsilon + 1)(01)^*(\varepsilon + 0)$$

Priorité des opérateurs

- L'opérateur de **clôture** (*) a la plus grande priorité : il s'applique à la plus petite suite de symboles sur sa gauche formant une expression régulière.
- L'opérateur de **concaténation** est prioritaire sur l'union.
- Finalement, on groupe avec l'**union**.

Exemples : $01^* + 1$ correspond à $(0(1)^*) + 1$

AF et expressions régulières : Des AFD aux expressions régulières et réciproquement

On a déjà montré les équivalences suivantes :

$$\text{AFD} \iff \text{AFN} \iff \varepsilon\text{-AFN}$$

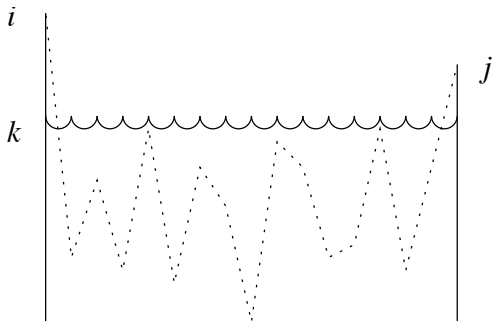
On veut montrer que les langages définis par les ER sont exactement ceux reconnus par les AFD.

- Passage d'un AFD à une ER
- Passage d'une ER à un ε -AFN.

Des AFD aux expressions régulières

$A = (Q, \Sigma, \delta, q_0, F)$ un AFD avec $Q = \{1, \dots, n\}$, $q_0 = 1$.

$R_{ij}^{(k)}$: l'expression régulière reconnaissant les mots w permettant de passer de l'état i à l'état j sans passer par un état intermédiaire de numéro strictement supérieur à k .



Des AFD aux expressions régulières (suite)

Calcul par induction

Base : $k = 0$ donc pas d'état intermédiaire.

- Cas 1 : $i \neq j$

Si $a_1, \dots, a_p$ sont les symboles permettant de passer de i à j , alors :

$$R_{ij}^{(0)} = \mathbf{a}_1 + \dots + \mathbf{a}_p$$

- Cas 2 : $i = j$

Si $a_1, \dots, a_p$ sont les symboles permettant de passer de i à j , alors :

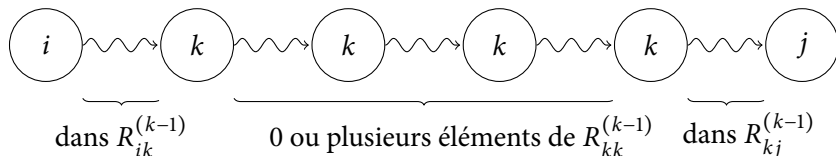
$$R_{ij}^{(0)} = \mathbf{a}_1 + \dots + \mathbf{a}_p + \varepsilon$$

Remarque : S'il n'y a pas de symbole permettant de passer de i à j alors

- Cas 1 : si $i \neq j$ alors $R_{ij}^{(0)} = \emptyset$
- Cas 2 : si $i = j$ alors $R_{ij}^{(0)} = \varepsilon$

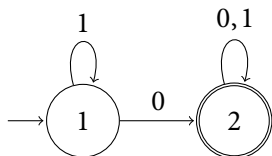
Induction : on montre que

$$R_{ij}^{(k)} = R_{ij}^{(k-1)} + R_{ik}^{(k-1)} (R_{kk}^{(k-1)})^* R_{kj}^{(k-1)}$$



ER définissant le langage reconnu par A : union des $R_{1j}^{(n)}$ pour $j \in F$.

Des AFD aux expressions régulières : un exemple



$R_{11}^{(0)}$	$\varepsilon + 1$
$R_{12}^{(0)}$	$\mathbf{0}$
$R_{21}^{(0)}$	$\emptyset$
$R_{22}^{(0)}$	$(\varepsilon + \mathbf{0} + 1)$

$$R_{ij}^{(1)} = R_{ij}^{(0)} + R_{i1}^{(0)}(R_{11}^{(0)})^*R_{1j}^{(0)}$$

	par substitution	forme simplifiée
$R_{11}^{(1)}$	$\varepsilon + 1 + (\varepsilon + 1)(\varepsilon + 1)^*(\varepsilon + 1)$	$\mathbf{1}^*$
$R_{12}^{(1)}$	$\mathbf{0} + (\varepsilon + 1)(\varepsilon + 1)^*\mathbf{0}$	$\mathbf{1}^*\mathbf{0}$
$R_{21}^{(1)}$	$\emptyset + \emptyset(\varepsilon + 1)^*(\varepsilon + 1)$	$\emptyset$
$R_{22}^{(1)}$	$\varepsilon + \mathbf{0} + 1 + \emptyset(\varepsilon + 1)^*\mathbf{0}$	$\varepsilon + \mathbf{0} + 1$

On utilise les règles de simplification suivantes :

$$(\varepsilon + R)^* = R^*, \quad R + RS^* = RS^*, \quad \emptyset R = R\emptyset = \emptyset, \quad \emptyset + R = R + \emptyset = R$$

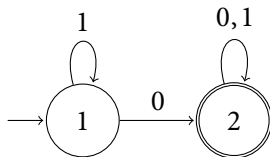
Des AFD aux expressions régulières : un exemple

	forme simplifiée
$R_{11}^{(1)}$	$\mathbf{1}^*$
$R_{12}^{(1)}$	$\mathbf{1}^* \mathbf{0}$
$R_{21}^{(1)}$	$\emptyset$
$R_{22}^{(1)}$	$\varepsilon + \mathbf{0} + \mathbf{1}$

$$R_{ij}^{(2)} = R_{ij}^{(1)} + R_{i2}^{(1)} (R_{22}^{(1)})^* R_{2j}^{(1)}$$

	par substitution	forme simplifiée
$R_{11}^{(2)}$	$\mathbf{1}^* + \mathbf{1}^* \mathbf{0} (\varepsilon + \mathbf{0} + \mathbf{1})^* \emptyset$	$\mathbf{1}^*$
$R_{12}^{(2)}$	$\mathbf{1}^* \mathbf{0} + \mathbf{1}^* \mathbf{0} (\varepsilon + \mathbf{0} + \mathbf{1})^* (\varepsilon + \mathbf{0} + \mathbf{1})$	$\mathbf{1}^* \mathbf{0} (\mathbf{0} + \mathbf{1})^*$
$R_{21}^{(2)}$	$\emptyset + (\varepsilon + \mathbf{0} + \mathbf{1}) (\varepsilon + \mathbf{0} + \mathbf{1})^* \emptyset$	$\emptyset$
$R_{22}^{(2)}$	$\varepsilon + \mathbf{0} + \mathbf{1} + (\varepsilon + \mathbf{0} + \mathbf{1}) (\varepsilon + \mathbf{0} + \mathbf{1})^* (\varepsilon + \mathbf{0} + \mathbf{1})$	$(\mathbf{0} + \mathbf{1})^*$

Des AFD aux expressions régulières : un exemple



	forme simplifiée
$R_{11}^{(2)}$	1^*
$R_{12}^{(2)}$	$1^* 0(0 + 1)^*$
$R_{21}^{(2)}$	$\emptyset$
$R_{22}^{(2)}$	$(0 + 1)^*$

Le langage reconnu est

$$R_{12}^{(2)} = 1^* 0(0 + 1)^*$$

L'algorithme précédent coûte cher !

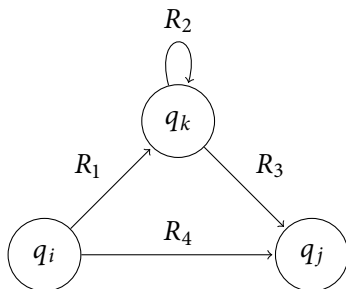
- n^3 ER à construire pour un AFD à n états et la longueur de chaque ER croît d'un facteur 4 à chaque étape.
- dans le pire cas : 4^n symboles !

Éliminer des états en autorisant les ER dans les transitions entre états.

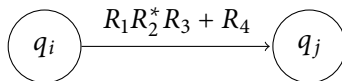
- Choisir un état final $q \in F$ et éliminer tous les autres états excepté l'état initial q_0 et q
- Appliquer le procédé de réduction pour tous les états finaux

Élimination d'état

Avant

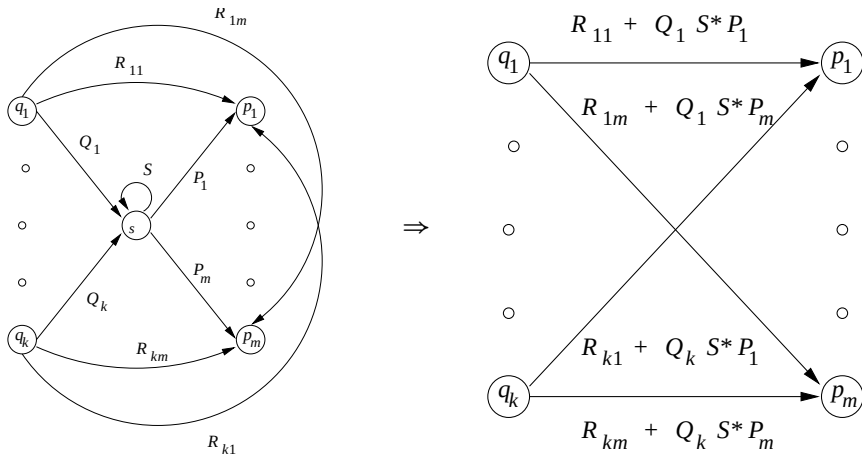


Après



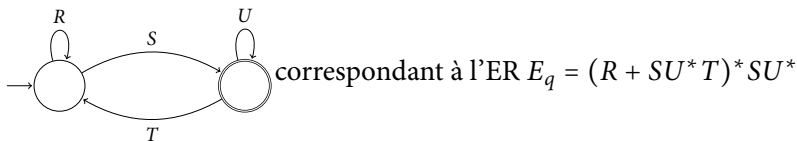
Élimination d'état (suite)

Élimination de l'état s

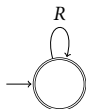


Élimination d'état : algorithme

Pour chaque état final $q \in F$, en appliquant la procédure d'élimination, on va obtenir un automate A_q de la forme



ou bien un automate A_q de la forme



correspondant à l'ER $E_q = R^*$

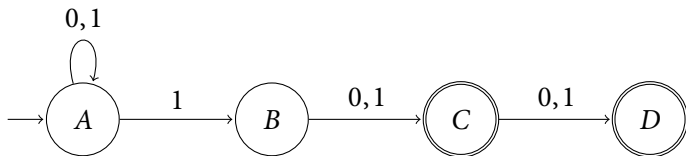
L'ER finale est

$$\bigoplus_{q \in F} E_q$$

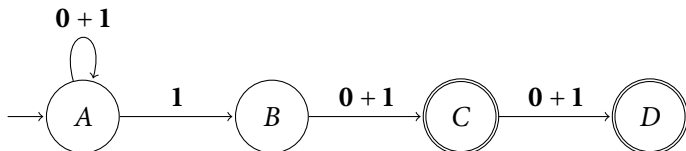
Élimination d'état : exemple

Un exemple : A tel que

$$L(A) = \{w : w = x1b \text{ ou } w = x1bc \text{ avec } x \in \{0,1\}^*, b, c \in \{0,1\}\}$$

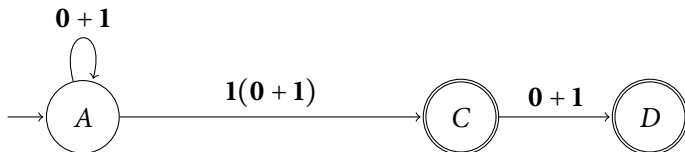


On étiquette les transitions par des ER :



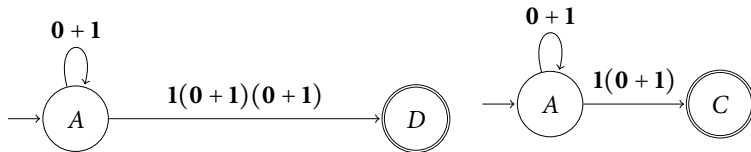
Élimination d'état : exemple

On élimine l'état B :



On supprime l'état C

On supprime l'état D



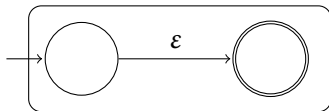
L'ER est donc $(0+1)^*1(0+1)(0+1) + (0+1)^*1(0+1)$

Des expressions régulières aux ε -AFN

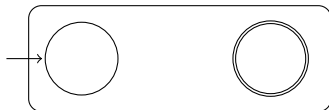
Preuve par récurrence structurale sur R : on suit la définition récursive des ER.

Base : variables et constantes

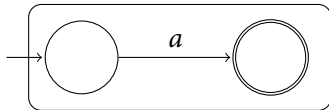
- Un ε -AFN N tel que $L(N) = \{\varepsilon\}$



- Un ε -AFN N tel que $L(N) = \emptyset$



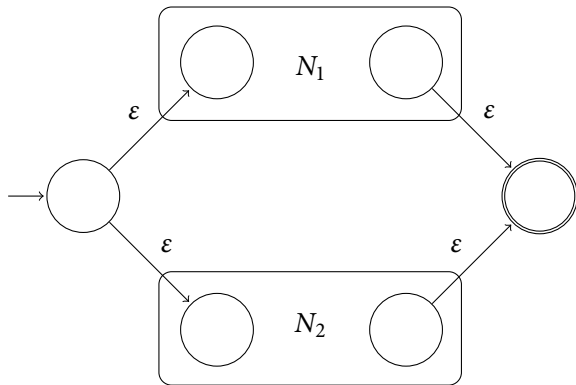
- Un ε -AFN N tel que $L(N) = \{a\}$ pour $a \in \Sigma$



Des expressions régulières aux ε -AFN

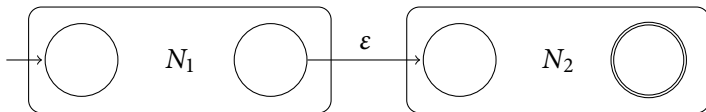
Induction :

- Étant donné N_1 et N_2 deux ε -AFN, définir un ε -AFN N reconnaissant $L(N_1) + L(N_2)$

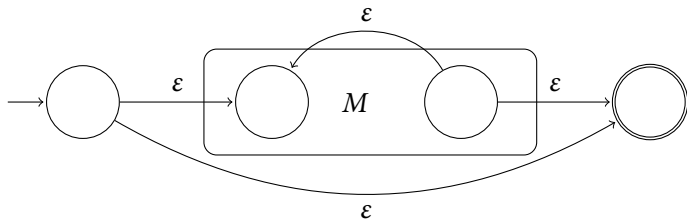


Des expressions régulières aux ε -AFN

- Étant donné N_1 et N_2 deux ε -AFN, définir un ε -AFN N reconnaissant $L(N_1).L(N_2)$



- Étant donné M un ε -AFN, définir un ε -AFN N reconnaissant $L(M)^*$



Exemple

Conversion de $(0 + 1)^*1(0 + 1)$

$(0 + 1)^*1(0 + 1)$

