Formulaire Automates

Sylvain Lombardy

Définition 1 Alphabet, mot, langage

- *Un* **alphabet** *est un ensemble fini de symboles ; chacun de ces symboles est appelé* **lettre**.
- *Un* **mot** *est une suite fini de* lettres *pris dans un* alphabet *fixé. La* **longueur d'un mot** *est la longueur de cette suite de lettres. La longueur d'un mot m est notée $|m|$.*
- *Un ensemble de mots, fini ou infini, sur un alphabet donné, est appelé* **langage**. *L'ensemble de tous les mots que l'on peut former avec un alphabet A est noté A^*.*
- *On peut faire le* **produit** *de deux mots sur le même alphabet, c'est-à-dire les* **multiplier**, *en les mettant bout à bout. Le* **mot vide** *est le mot de longueur 0, il est noté ε. Si on fait le produit de n'importe quel mot m avec le mot vide, on obtient m. Il ne faut pas confondre le mot vide avec le langage vide (qui ne contient aucun mot).*
- *Un* **préfixe** *d'un mot m est un mot u que l'on peut compléter avec un mot v de sorte que $m = uv$. Le mot u est un* **préfixe propre** *de m si u est un préfixe de m et $|u| < |m|$.*
  Un **suffixe** *d'un mot m est un mot v que l'on peut compléter avec un mot u de sorte que $m = uv$. Le mot v est un* **suffixe propre** *de m si u est un suffixe de m et $|v| < |m|$.*
  Un **facteur** *d'un mot m est un mot w que l'on peut compléter avec deux mots u et v de sorte que $m = uwv$. Le mot w est un* **facteur propre** *de m si w est un facteur de m et $|w| < |m|$.*

Définition 2 Automate non déterministe

Un **automate fini non déterministe** *(NFA en anglais) $\mathcal{A}$ est défini par 5 ensembles :*

- *un ensemble fini d'***états** *(généralement noté Q) ;*
- *un* alphabet *fini (généralement noté A) ;*
- *un ensemble fini de* **transitions** *(généralement noté E) ;*
  chaque transition est caractérisée par :
  - *un* **état de départ** *p (appartenant à Q) ;*
  - *une* **étiquette** *a (appartenant à A) ;*
  - *un* **état d'arrivée** *q (appartenant à Q) ;*

  on dénote une telle transition (p, a, q).
- *un ensemble d'états* **initiaux** *(inclus dans Q) (généralement noté I) ;*
- *un ensemble d'états* **terminaux** *(inclus dans Q) (généralement noté T)*

On dénote l'automate et toutes ses composantes $\mathcal{A} = \langle Q, A, E, I, T\rangle$.

Définition 3 Chemin et langage accepté

- *Dans un automate, un* **chemin** *(fini) est une suite (finie) de transitions consécutives, c'est-a-dire que l'état d'arrivée d'une transition est le même que l'état de départ de la transition suivante.*
- *Un chemin est un* **chemin réussi** *si l'état de départ de la premièere transition est initial et l'état d'arrivée de la dernière transition est terminal.*
- *L'***étiquette d'un chemin** *est le mot obtenu en concaténant toutes les étiquettes des transitions du chemin, dans le même ordre.*
- *Un mot est* **accepté** *par un automate s'il est l'étiquette d'un chemin réussi de cet automate.*
- *Le* **langage reconnu** *par un automate est l'ensemble des étiquettes des chemins réussis de cet automate.*
- *Deux automates sont* **équivalents** *s'ils reconnaissent le même langage.*
- *Un langage est* **reconnaissable** *s'il est reconnu par un automate fini non déterministe.*
- *Dans un automate, un état p est* **accessible** *s'il existe un chemin partant d'un état initial qui arrive dans l'état p.*
  Dans un automate, un état p est **co-accessible** *s'il existe un chemin partant de p qui arrive dans un état terminal.*
- *Un état* **utile** *est un état à la fois accessible et co-accessible. Tout état d'un chemin réussi est utile.*

– *Un automate est* **émondé** *si tous ses états sont utiles.*

Proposition 1 *Tout NFA est équivalent à un NFA émondé.*

Partie-Accessible($\mathcal{A}$)
$\mathcal{B}$: automate vide
N :=états initiaux de $\mathcal{A}$
pour tout état initial p de $\mathcal{A}$
créer un état initial p dans $\mathcal{B}$
tant que N non vide
$p := pop(N)$
si p est terminal dans $\mathcal{A}$
rendre p terminal dans $\mathcal{B}$
pour toute transition (p, a, q) de $\mathcal{A}$
si q n'est pas un état de $\mathcal{B}$
$N \leftarrow q$
créer l'état q dans $\mathcal{B}$
créer une transition (p, a, q) dans $\mathcal{B}$
retourner $\mathcal{B}$

Partie-Co-Accessible($\mathcal{A}$)
$\mathcal{B}$: automate vide
N :=états terminaux de $\mathcal{A}$
pour tout état terminal p de $\mathcal{A}$
créer un état terminal p dans $\mathcal{B}$
tant que N non vide
$q := pop(N)$
si q est initial dans $\mathcal{A}$
rendre q initial dans $\mathcal{B}$
pour toute transition (p, a, q) de $\mathcal{A}$
si p n'est pas un état de $\mathcal{B}$
$N \leftarrow p$
créer l'état p dans $\mathcal{B}$
créer une transition (p, a, q) dans $\mathcal{B}$
retourner $\mathcal{B}$

Partie-Emonde($\mathcal{A}$)
retourner Parite-Accessible(Partie-Co-accessible($\mathcal{A}$))

Proposition 2 *On peut décider*
1) si le langage accepté par un automate est vide ;
2) si un mot donné est accepté par un automate.

Langage-Vide($\mathcal{A}$)
S :=états initiaux
N :=états initiaux
tant que N non vide
$p := pop(N)$
si p est terminal
retourner faux
pour toute transition (p, a, q)
si q n'est pas dans S
$N \leftarrow q$
$S \leftarrow q$
retourner vrai

Etats-atteints($\mathcal{A}, w$)
S_0 := états initiaux
pour i de 1 à $|w|$
$S_i := \emptyset$
pour tout p dans S_{i-1}
pour toute transition (p, w_i, q)
$S_i \leftarrow p$
retourner $S_{|w|}$

Accepte($\mathcal{A}, w$)
S :=Etats-atteints($\mathcal{A}, w$)
pour tout p dans S
si p est terminal
retourner vrai
retourner faux

Définition 4 Automate déterministe

- *Un* **automate fini déterministe** *(DFA en anglais) est un automate fini non déterministe tel que :*
  - *le nombre d'états initiaux est inférieur ou égal à 1 ;*
  - *pour tout état p, pour toute lettre a, il existe au plus une transition dont l'état de départ est p et l'étiquette est a.*
- *Un automate déterministe est* **complet** *si :*
  - *il y a exactement un état initial ;*
  - *pour tout état p, pour toute lettre a, il existe exactement une transition dont l'état de départ est p et l'étiquette est a.*
- *On ne peut pas toujours avoir un automate complet et émondé.*

Déterministe-Accepte$(\mathcal{A}, w)$
    p_0 :=état initial
    pour i de 1 à $|w|$
        p_i := successeur de p_{i-1} par w_i
    si p est terminal
        retourner vrai
    sinon
        retourner faux

Proposition 3 *Tout NFA est équivalent à un DFA complet.*

Compléter$(\mathcal{A})$
    créer état *puits*
    pour toute lettre a
        créer une transition $(puits, a, puits)$
    pour tout état p
        pour toute lettre a
            s'il n'existe pas de transition (p, a, q)
                créer une transition $(p, a, puits)$

Déterminisation$(\mathcal{A})$
    $\mathcal{D}$ automate vide
    créer dans $\mathcal{D}$ un état I initial correspondant à l'ensemble des état initiaux de $\mathcal{A}$
    $N := \{I\}$
    tant que N non vide
        $P := pop(N)$
        pour toute lettre a
            $X := \emptyset$
            pour tout p dans P
                pour toute transition (p, a, q) dans $\mathcal{A}$
                    $X \leftarrow q$
            si X n'est pas un état de $\mathcal{D}$
                créer l'état X dans $\mathcal{D}$
                $N \leftarrow X$
            créer une transition (P, a, X) dans $\mathcal{D}$
        si P contient un état terminal de $\mathcal{A}$
            rendre P terminal dans $\mathcal{D}$
    retourner $\mathcal{D}$

Définition 5 Automates avec ε-transitions

- *Un* **automate avec ε-transitions** *est un automate fini non déterministe qui, en plus des transitions normales contient des* **ε-transitions**, *c'est-à-dire des transitions étiquetées par le mot vide.*
- *Pour tout état d'un automate avec ε-transitions, l'ensemble des* **ε-successeurs** *est l'ensemble des états que l'on peut atteindre en suivant un chemin d'ε-transitions.*

Proposition 4 *Tout NFA avec ε-transitions est équivalent à un NFA sans ε-transitions.*

```
Epsilon-Successeurs(A)
    pour tout p dans Q
        S_p := ∅
    pour tout état p
        pour tout q tel qu'il existe une transition (p, ε, q)
            S_p ← q
    faire
        boucle : = faux
        pour tout état p
            N_p := ∅
            pour tout q dans S_p
                pour tout r dans S_q
                    si r n'est pas dans S_p
                        N_p ← r
                    S_p ← r
            si N_p n'est pas vide
                boucle := vrai
    tant que boucle est vrai
    retourner S
```

```
Automate-sans-Epsilon(A)
    S :=Epsilon-Successeurs(A)
    B :automate vide
    pour tout état p de A
        créer un état p de B
        si p est initial dans A
            rendre p initial dans B
        si p est terminal dans A
            rendre p initial dans B
    pour toute transition (p, a, q) de A (a ≠ ε)
        créer une transition (p, a, q) dans B
    pour tout état p
        pour tout q dans S_p
            si q est terminal dans A
                rendre p terminal dans B
    pour toute transition (q, a, r) dans A
            créer une transition (p, a, r) dans B
    retourner B
```

Définition 6 Automate standard
Un **automate standard** *est un automate fini non déterministe tel que :*
– il y a un seul état initial ;
– l'état initial n'est l'état d'arrivée d'aucune transition.

Proposition 5 *Tout NFA est équivalent à un automate standard.*

```
Standard(A)
    S : automate vide
    pour tout état p dans A
        créer un état p dans S
        si p est terminal dans A
            rendre p terminal dans S
    pour toute transition (p, a, q) dans A
        créer une transition (p, a, q) dans S
    créer un état initial i dans S
    pour tout état initial p dans A
        pour toute transition (p, a, q)
            créer une transition (i, a, q) dans S
        si p est terminal
            rendre i terminal
    retourner S
```

Définition 7 Opérations sur les langages
- *L'***union** *de deux langages L et K est le langage qui contient les mots qui appartiennent à au moins un des deux langages ; on le note $L \cup K$;*
- *l'***intersection** *de deux langages L et K est le langage qui contient les mots qui appartiennent aux deux langages ; on le note $L \cap K$;*
- *le* **complémentaire** *d'un langage L est le langage (sur le même alphabet) des mots qui ne sont pas dans L, on le note $\bar{L}$;*
- *la* **concaténation** *de deux langages L et K est le langage des mots de la forme uv, où u est un mot de L et v un mot de K ; on le note LK ;*
- *la* **puissance** *n-ième d'un langage est le produit de n fois le langage par lui-même, c'est-à-dire l'ensemble des mots de la forme $u_1u_2...u_n$, où chaque u_i appartient à L ; on le note L^n ; par convention $L^0 = \{\varepsilon\}$;*
- *l'***étoile** *d'un langage est l'union de toutes les puissances du langages ; on la note L^* ; l'étoile d'un langage contient toujours le mot vide.*

Proposition 6 *L'union, l'intersection, le complémentaire, la concaténation, la puissance et l'étoile de langages reconnaissables donnent des langages reconnaissables.*

```
Union(A,B)
    C : automate vide
    pour tout état p dans A ou dans B
        créer un état p dans C
        si p est initial dans A ou dans B
            rendre p initial dans C
        si p est terminal dans dans A ou B
            rendre p terminal dans C
    pour toute transition (p, a, q) dans A ou dans B
        créer une transition (p, a, q) dans C
    retourner C
```

Complémentaire($\mathcal{A}$)
- $\mathcal{C}$:=Déterminisation($\mathcal{A}$) (automate déterministe complet)
- pour tout état p de $\mathcal{C}$
  - si p est terminal
    - rendre p non terminal
  - sinon
    - rendre p terminal
- retourner $\mathcal{C}$

Intersection($\mathcal{A}, \mathcal{B}$) (sans ε)
- $\mathcal{P}$: automate vide
- $N := \emptyset$
- pour tout état initial p de $\mathcal{A}$
  - pour tout état initial q de $\mathcal{B}$
    - créer un état initial (p, q) dans$\mathcal{P}$
    - $N \leftarrow (p, q)$
- tant que N est non vide
  - (p,q) :=pop(N)
  - si p est terminal dans $\mathcal{A}$ et q est terminal dans $\mathcal{B}$
    - (p, q) est terminal dans $\mathcal{P}$
  - pour toute transition (p, a, p') de $\mathcal{A}$
    - pour toute transition (q, a, q') de $\mathcal{B}$
      - si (p', q') n'est pas un état de $\mathcal{P}$
        - créer l'état (p', q') dans $\mathcal{P}$
        - $N \leftarrow (p', q')$
      - créer la transition $((p, q), a, (p', q'))$ dans $\mathcal{P}$
- retourner $\mathcal{P}$

Concaténation($\mathcal{A}$,$\mathcal{B}$)
- $\mathcal{C}$: automate vide
- pour tout état p dans $\mathcal{A}$ ou dans $\mathcal{B}$
  - créer un état p dans $\mathcal{C}$
  - si p est initial dans $\mathcal{A}$
    - rendre p initial dans $\mathcal{C}$
  - si p est terminal dans $\mathcal{B}$
    - rendre p terminal dans $\mathcal{C}$
- pour toute transition (p, a, q) dans $\mathcal{A}$ ou dans $\mathcal{B}$
  - créer une transition (p, a, q) dans $\mathcal{C}$
- pour tout état terminal p dans $\mathcal{A}$
  - pour tout état initial q dans $\mathcal{B}$
    - pour toute transition (q, a, r) dans $\mathcal{B}$
      - créer une transition (p, a, r) dans $\mathcal{C}$
    - si q est terminal dans $\mathcal{B}$
      - rendre p terminal dans $\mathcal{C}$
- retourner $\mathcal{C}$

Etoile($\mathcal{A}$)
- $\mathcal{S}$:=Standard($\mathcal{A}$)
- i :=état initial de $\mathcal{S}$
- pour tout état terminal p de $\mathcal{S}$
  - pour toute transition (i, a, q)
    - créer une transition (p, a, q)
- rendre i terminal
- retourner $\mathcal{S}$

Définition 8 Langage rationnel

- *Un* **langage rationnel** *est obtenu à partir des lettres et du mot vide en utilisant un nombre fini de fois les opérations d'union, de produit et d'étoile. Ces trois opérations sont appelées* **opérations rationnelles**.
- *Une* **expression rationnelle** *décrit les opérations utilisées pour engendrer un langage rationnel. L'expression vide dénote le langage vide. Une expression rationnelle ne comporte donc ni intersection ni complémentaire.*
- *L'utilitaire* `grep` *ou les lexeurs comme* `lex` *ou* `flex` *utilisent des expressions étendues dans lesquels on peut utiliser*
  - *l'ensemble des puissances strictement positives ;*
  - *des intervalles de lettres ;*
  - *le complémentaire sur l'alphabet ;*
  - *etc.*

Proposition 7 *Tout langage rationnel est reconnaissable par un automate standard.*
De plus si un langage est décrit par une expression rationnelle avec n lettres, il existe un automate standard avec $n+1$ états qui reconnaît le même langage.

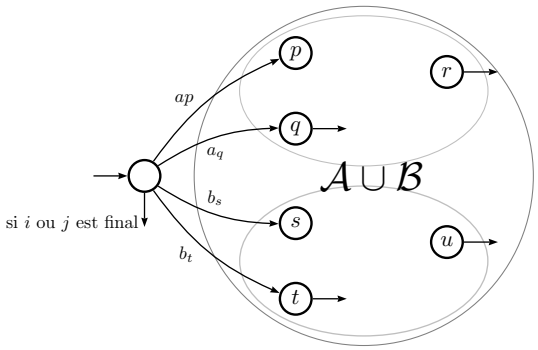
Union-Standard($\mathcal{A}$,$\mathcal{B}$)
$\mathcal{C}$: automate vide
$i_{\mathcal{A}}$:=état initial de $\mathcal{A}$
$i_{\mathcal{B}}$:=état initial de $\mathcal{B}$
pour tout état $p \neq i_{\mathcal{A}}, i_{\mathcal{B}}$ dans $\mathcal{A}$ ou dans $\mathcal{B}$
  créer un état p dans $\mathcal{C}$
  si p est terminal dans $\mathcal{A}$ ou dans $\mathcal{B}$
    rendre p terminal dans $\mathcal{C}$
créer un état i dans $\mathcal{C}$
pour toute transition (p, a, q) dans $\mathcal{A}$ ou dans $\mathcal{B}$
  si $p \neq i_{\mathcal{A}}, i_{\mathcal{B}}$ créer une transition (p, a, q) dans $\mathcal{C}$
  sinon créer une transition (i, a, q) dans $\mathcal{C}$
retourner $\mathcal{C}$

Concaténation-Standard($\mathcal{A}$,$\mathcal{B}$)
$\mathcal{C}$: automate vide
$i_{\mathcal{B}}$:=état initial de $\mathcal{B}$
pour tout état $p \neq i_{\mathcal{B}}$ dans $\mathcal{A}$ ou dans $\mathcal{B}$
  créer un état p dans $\mathcal{C}$
  si p est initial dans $\mathcal{A}$
    rendre p initial dans $\mathcal{C}$
  si p est terminal dans $\mathcal{B}$
    rendre p terminal dans $\mathcal{C}$
si $i_{\mathcal{B}}$ est terminal dans $\mathcal{B}$
  pour tout état terminal p de $\mathcal{A}$
    rendre p terminal dans $\mathcal{C}$
pour toute transition (p, a, q) dans $\mathcal{A}$
  créer une transition (p, a, q) dans $\mathcal{C}$
pour toute transition (p, a, q) dans $\mathcal{B}$
  si $p \neq i_{\mathcal{B}}$ créer une transition (p, a, q) dans $\mathcal{C}$
  sinon pour tout état terminal p de $\mathcal{A}$
    créer une transition (p, a, q) dans $\mathcal{C}$
retourner $\mathcal{C}$

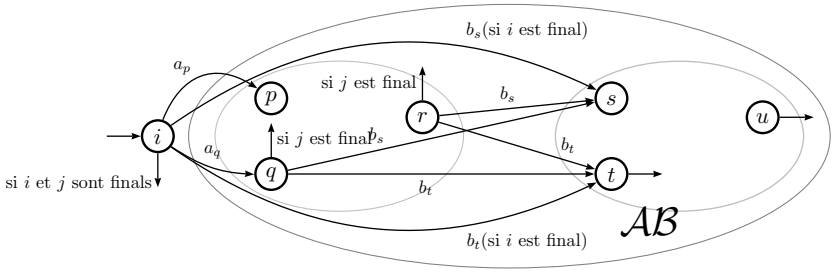
Etoile-Standard($\mathcal{A}$)
$\mathcal{C}$:=copie de $\mathcal{A}$
i :=état initial de $\mathcal{C}$
pour tout état terminal p de $\mathcal{C}$
  pour toute transition (i, a, q)
    créer une transition (p, a, q)
rendre i terminal
retourner $\mathcal{C}$

Soit $\mathcal{A} = \langle Q, A, E, \{i\}, T\rangle$ et $\mathcal{B} = \langle R, A, F, \{j\}, U\rangle$ deux automates standards.

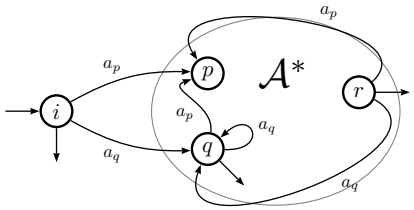
- L'automate standard $\mathcal{A} \cup \mathcal{B} = \langle Q \cup R \setminus \{j\}, A, G, \{i\}, V\rangle$ est défini par :



- L'automate standard $\mathcal{AB} = \langle Q \cup R \setminus \{j\}, A, G, \{i\}, V\rangle$ est :



- L'automate standard $\mathcal{A}^* = \langle Q, A, E_*, \{i\}, T_*$ est :



Définition 9 Position et automate des positions

- *Dans une expression rationnelle, on peut donner un numéro distinct à chaque occurence de lettre ; on appelle* **position** *le numéro de chaque lettre ; on note* $\mathsf{Pos(E)}$ *l'ensemble des positions et* $\ell(p)$ *est la lettre en position* p*.*
- *L'automate des positions est défini de la manière suivante :*
  - *l'ensemble de ses états est* $\{0\} \cup \mathsf{Pos(E)}$ *;*
  - *l'unique état initial est* $\{0\}$ *;*
  - *l'état* 0 *est terminal si* $\mathsf{Null(E)}$ *est vrai ;*
  - *un état* p *de* $\mathsf{Pos(E)}$ *est terminal si* p *est dans* $\mathsf{Last}(E)$ *;*
  - *pour tout* p *dans* $\mathsf{First}(E)$*, il y a une transition* 0 *à* p *étiquetée par* $\ell(p)$ *;*
  - *pour toute position* p*, pour tout* q *dans* $\mathsf{Follow}(E,p)$*, il y a une transition* p *à* q *étiquetée par* $\ell(q)$*.*

$$
\begin{aligned}
\mathsf{Null}(\emptyset) &= \text{False}\\
\mathsf{Null}(\varepsilon) &= \text{True}\\
\mathsf{Null}(a_p) &= \text{False}\\
\mathsf{Null}(E \cup F) &= \mathsf{Null}(E)\text{or}\mathsf{Null}(F)\\
\mathsf{Null}(EF) &= \mathsf{Null}(E)\text{and}\mathsf{Null}(F)\\
\mathsf{Null}(E^*) &= \text{True}
\end{aligned}
$$

$$
\begin{aligned}
\mathsf{First}(\emptyset) = \mathsf{First}(\varepsilon) &= \emptyset\\
\mathsf{First}(a_p) &= \{p\}\\
\mathsf{First}(E \cup F) &= \mathsf{First}(E) \cup \mathsf{First}(F)\\
\mathsf{First}(EF) &= \begin{cases} \mathsf{First}(E) \cup \mathsf{First}(F) & \text{si } \mathsf{Null}(E)\\ \mathsf{First}(E) & \text{sinon}\end{cases}\\
\mathsf{First}(E^*) &= \mathsf{First}(E)
\end{aligned}
$$

$$
\begin{aligned}
\mathsf{Last}(\emptyset) = \mathsf{Last}(\varepsilon) &= \emptyset\\
\mathsf{Last}(a_p) &= \{p\}\\
\mathsf{Last}(E \cup F) &= \mathsf{Last}(E) \cup \mathsf{Last}(F)\\
\mathsf{Last}(EF) &= \begin{cases} \mathsf{Last}(E) \cup \mathsf{Last}(F) & \text{si } \mathsf{Null}(F)\\ \mathsf{Last}(F) & \text{sinon}\end{cases}\\
\mathsf{Last}(E^*) &= \mathsf{Last}(E)
\end{aligned}
$$

$$
\begin{aligned}
\mathsf{Follow}(\emptyset, p) = \mathsf{Follow}(\varepsilon, p) &= \emptyset\\
\mathsf{Follow}(a_q, p) &= \emptyset\\
\mathsf{Follow}(E \cup F, p) &= \begin{cases} \mathsf{Follow}(E,p) & \text{si } p \text{ est une position de } E\\ \mathsf{Follow}(F,p) & \text{si } p \text{ est une position de } F\\ \emptyset & \text{sinon}\end{cases}\\
\mathsf{Follow}(EF, p) &= \begin{cases} \mathsf{Follow}(E,p) & \text{si } p \text{ est une position de } E \text{ et } p \notin \mathsf{Last}(E)\\ \mathsf{Follow}(E,p) \cup \mathsf{First}(F) & \text{si } p \in \mathsf{Last}(E)\\ \mathsf{Follow}(F,p) & \text{si } p \text{ est une position de } F\\ \emptyset & \text{sinon}\end{cases}\\
\mathsf{Follow}(E^*, p) &= \begin{cases} \mathsf{Follow}(E,p) \cup \mathsf{First}(E) & \text{si } p \text{ est une position de } E\\ \emptyset & \text{sinon}\end{cases}
\end{aligned}
$$

Proposition 8 *L'automate des positions d'une expression est le même que l'automate standard.*

Proposition 9 Lemme d'Arden
Soit A et B deux langages. On considère l'équation $X = AX \cup B$.
Le langage A^*B est la plus petite solution de cette équation.
Si A ne contient pas le mot vide, cette solution est unique.

Proof.

1. **A^*B est une solution**
   $A^*B = (AA^* \cup \{\varepsilon\})B = A(A^*B) \cup B$.
2. **A^*B est la plus petite solution**
   Soit S une autre solution. Supposons par l'absurde qu'il existe un mot m dans A^*B qui n'est pas dans S et prenons le de longueur minimale. Le mot m est dans A^*B, donc soit a) m est dans B, soit b) m est de la forme uv, avec $u \neq \varepsilon$ dans A et v dans A^*B.
   Cas a) : m est dans B, comme $S = AS \cup B$, m est dans S, impossible.
   Cas b) : $m = uv$, comme $|v| < |m|$, par minimalité de m, v est dans S ; comme $S = AS \cup B$ et que u est dans A, $uv = m$ est dans S, impossible.
3. **Si A ne contient pas ε, A^*B est la plus grande solution**
   Soit S une autre solution. Supposons par l'absurde qu'il existe un mot m dans S qui n'est pas dans A^*B et prenons le de longueur minimale. Le mot m est dans $S = AS \cup B$, donc soit a) m est dans B, soit b) m est de la forme uv, avec u dans A et v dans A^*B.
   Cas a) : m est dans B, donc m est dans A^*B, impossible.
   Cas b) : $m = uv$, comme A ne contient pas le mot vide, $|u| > 0$, donc $|v| < |m|$ et par minimalité de m, v appartient à A^*B ; donc $uv = m$ est dans A^*B, impossible.

□

Définition 10 Futur d'un état
Soit $\mathcal{A}$ un automate et p un état de $\mathcal{A}$. Le futur de p dans $\mathcal{A}$ est l'ensemble des étiquettes des chemins de p à un état terminal. On note L_p le futur de p.

Proposition 10 *Le futur de p dans $\mathcal{A}$ vérifie :*

$$L_p = \begin{cases} \varepsilon \cup \bigcup_{(p,a,q)\, transition} aL_q & \text{si } p \text{ est initial,} \\ \bigcup_{(p,a,q)\, transition} aL_q & \text{sinon.} \end{cases}$$

Si I est l'ensemble des états initiaux de $\mathcal{A}$, le langage reconnu par A est :

$$L = \bigcup_{p \in I} L_p.$$

Proposition 11 *Tout langage reconnaissable est rationnel.*

En effet, grâce au lemme d'Arden, on peut résoudre le système d'équations de la proposition précédente et trouver une expression rationnelle pour le langage.

Théorème 1 Kleene
Les langages rationnels sont exactement les langages reconnaissables.

Définition 11 Automate généralisé

- *Un* **automate généralisé** *est un automate dans lequel les étiquettes des transitions sont des expressions rationnelles.*
- *Dans un automate généralisé, s'il existe deux transitions avec les mêmes états de départ et d'arrivée, on les remplace par une seule transition dont l'étiquette est l'union des deux étiquettes précédentes.*

Elimination-etats($\mathcal{A}$)
créer deux états i et t
pour tout état initial p
créer une transition (i, ε, p)
pour tout état terminal p
créer une transition (p, ε, t)
pour tout état $p \neq i, t$
s'il existe une transition (p, E, p)
$G := \mathsf{E}^*$
supprimer la transition (p, E, p)
sinon
$G := 0$
pour toute transition (q, F, p)
pour toute transition (p, H, r)
si $G = 0$
créer une transition (q, FH, r)
sinon
créer une transition (q, FGH, r)
supprimer l'état p
retourner l'étiquette de la transition entre i et t

Proposition 12 *L'expression retournée par l'algorithme* **Elimination-etats** *représente le langage reconnu par l'automate. Si on élimine les états dans le même ordre que l'on élimine les inconnues du système de la proposition 9, on obtient des expressions similaires.*