Notes de cours Informatique théorique - I
(version 1.9, 17/05/17)

S.V.P. ME SIGNALER TOUTE FAUTE DE FRAPPE, GRAMMAIRE, ORTHOGRAPHE OU LOGIQUE; LES FAUTES DE LOGIQUE VOUS APPORTENT DES POINTS SUPPLÉMENTAIRES POUR LA NOTE FINALE. TOUTE SUGGESTION D'UNE MEILLEURE TRADUCTION D'UN TERME ANGLAIS SERA ÉGALEMENT BIENVENUE.

Convention 1 *Les variables non explicites i, j, k, n, m, etc. sont des entiers non-négatifs.*

Définition 1 *Un alphabet, noté Σ, est un ensemble fini non vide de* symboles *ou* lettres.

Définition 2 *Soit Σ un alphabet. Un* mot sur Σ *est défini récursivement par*

1. *Le mot vide, noté ε, ne contenant aucun symbole, est un mot.*
2. *Pour tout $a \in \Sigma$, a est un mot.*
3. *Si $u = a_1 \dots a_n$ et $v = b_1 \dots b_m$ sont des mots, alors la* concaténation *de u et de v, notée uv et définie par $uv = a_1 \dots a_n b_1 \dots b_m$ est un mot.*

La longeur *du mot $u = a_1 \dots a_n$, notée $|u|$ est n, le nombre de symboles de u.*

Pour mieux comprendre, regardons une manière équivalente de définir les même concepts.

Définition 3 *Soit Σ un alphabet. On définit, pour $k \in \mathbb{N}$,*

- $\Sigma^k = \{a_1 \dots a_k : a_i \in \Sigma, i = 1, \dots, k\}$;
- $\Sigma^* = \cup_{k \in \mathbb{N}} \Sigma^k$.

Les éléments de Σ^* sont des mots *sur Σ. Si $w \in \Sigma^k$, on dit que w est de longueur k; on note $|w|$ la longueur de w. Le mot vide, ε, est le seul élément de Σ^0 - c'est l'unique mot de longueur* 0.

Définition 4 *Un* langage *L sur Σ est une partie de Σ^* (i.e. $L \subseteq \Sigma^*$).*

Définition 5 *Soit Σ un alphabet et soit L, L_1, L_2 des langages sur Σ. On définit*

1. *Le* complément *de L, $\overline{L} = \Sigma^* \setminus L = \{w \in \Sigma^* : w \notin L\}$;*
2. *La* concaténation *de L_1 et L_2, $L_1 \cdot L_2 = \{uv : u \in L_1,\ v \in L_2\}$;*
3. *La k-ème puissance de L, L^k récursivement par*
   (a) *$L^0 = \{\varepsilon\}$;*
   (b) *$L^k = L \cdot L^{k-1}$ pour $k > 0$;*
4. *La* fermeture (clôture)de Kleene*, $L^* = \cup_{k \in \mathbb{N}} L^k$.*

Exemple 1 *Soit un alphabet Σ et soit un langage L sur Σ. Alors*

- $L^1 = L \cdot L^0 = \{uv : u \in L, v \in L^0\} = \{uv : u \in L, v = \varepsilon\} = \{u \in \Sigma^* : u \in L\}$;
- $\emptyset \cdot L = L \cdot \emptyset = \emptyset$;
- $\{\varepsilon\} \cdot L = L \cdot \{\varepsilon\} = L$.

Définition 6 *Un automate fini déterministe, AFD,* $M = (Q, \Sigma, \delta, s, F)$ *est défini par*

- Q, *un ensemble fini d'*états*;*
- Σ, *un alphabet (ensemble fini) de* symboles *ou* lettres*;*
- $\delta : Q \times \Sigma \longrightarrow Q$, *une fonction de transition;*
- $s \in Q$, *un état* intial*;*
- $F \subseteq Q$, *un ensemble d'états* acceptants.

Souvent, q_0 est utilisé plutôt que s pour l'état initial.

Définition 7 *Soit* $M = (Q, \Sigma, \delta, s, F)$, *un AFD, et soit* $w \in \Sigma^*$, $w = a_1 a_2 \dots a_k$, $a_i \in \Sigma$ *pour* $i = 1, \dots, k$, $k \in \mathbb{N}$. *Un* calcul, *aussi appelé une* exécution, *de* M *sur* w *est une suite* $\{r_i\}_{i=0}^{k}$ *(*$k = |w|$*) d'états telle que*

1. $r_0 = s$*;*
2. $r_i = \delta(r_{i-1}, a_i)$ *pour* $i = 1, \dots, k$.

Définition 8 *Soit* $M = (Q, \Sigma, \delta, s, F)$, *un AFD. Soit* $\hat{\delta} : Q \times \Sigma^* \longrightarrow Q$ *l'extension de* δ *aux mots sur* Σ, *définie par*

1. $\hat{\delta}(q, \varepsilon) = q$ *pour tout* $q \in Q$*;*
2. $\hat{\delta}(q, ua) = \delta(\hat{\delta}(q, u), a)$, *pour* $u \in \Sigma^*$, $a \in \Sigma$.

C'est-à-dire, $\hat{\delta}(q, w)$ est l'état dans lequel se trouve M après avoir lu w, en commençant dans l'état q. On voit que $\{r_i\}_{i=0}^{k}$ est un calcul de M sur $w = a_1 \dots a_k$ si et seulement si $r_o = s$ et pour tout $i, 1 \leq i \leq k$, $\hat{\delta}(s, a_1 \dots a_i) = r_i$.

Définition 9 *Le* langage reconnu *ou, mieux,* décidé, *par* $M = (Q, \Sigma, \delta, s, F)$ *est le langage* $L(M) = \{w \in \Sigma^* : \hat{\delta}(s, w) \in F\}$.

Définition 10 *Un langage* L *sur un alphabet* Σ *est* régulier *ou* rationnel *s'il existe un AFD* M *tel que* $L = L(M)$.

On note la classe des langages réguliers $\mathcal{L}_{\mathcal{R}}$.

Définition 11 *Deux automates finis* M_1 *et* M_2 *sont* équivalents *si* $L(M_1) = L(M_2)$.

Les lemmes suivants devraient être assez clairs. On les prouve surtout pour l'exercice.

Lemme 1 *Soit* $M = (Q, \Sigma, \delta, s, F)$, *un AFD,* $q \in Q$ *et* $u, v \in \Sigma^*$. *Alors* $\hat{\delta}(q, uv) = \hat{\delta}(\hat{\delta}(q, u), v)$.

Démonstration. On fait une récurrence sur la longueur de v. Si $v = \varepsilon$, $\hat{\delta}(q, u\varepsilon) = \hat{\delta}(q, u) = \hat{\delta}(\hat{\delta}(q, u), \varepsilon)$. Si $v = xa$, $x \in \Sigma^*, a \in \Sigma$, $\hat{\delta}(q, uv) = \hat{\delta}(q, uxa) = \delta(\hat{\delta}(q, ux), a) = \delta(\hat{\delta}(\hat{\delta}(q, u), x), a) = \hat{\delta}(\hat{\delta}(q, u), xa) = \hat{\delta}(\hat{\delta}(q, u), v)$. □

Lemme 2 *Soit* $M = (Q, \Sigma, \delta, s, F)$, *un AFD, et soit* Γ, *un alphabet. Alors il existe un AFD* $M' = (Q', \Sigma \cup \Gamma, \delta', s', F')$ *équivalent.*

Démonstration. En choisissant un état $p \notin Q$, on met $Q' = Q \cup \{p\},\ s' = s,\ F' = F$ et on définit

$$\delta'(q,a) = \begin{cases} \delta(q,a) & \text{si } q \in Q, a \in \Sigma \\ p & \text{sinon} \end{cases}$$

Bien qu'il soit clair que $L(M) = L(M')$, prouvons-le, c'est-à-dire, prouvons que $\hat{\delta}(s,w) \in F$ si et seulement si $\hat{\delta}'(s,w) \in F' = F$.
Soit $w \in (\Sigma \cup \Gamma)^*$, disons $w = a_1 \dots a_n,\ a_i \in \Sigma \cup \Gamma$ pour $i = 1, \dots, n$. Soit $k = max\{i : a_1, a_2, \dots, a_i \in \Sigma\}$. On a que $\hat{\delta}'(s, a_1 \dots a_j) = \hat{\delta}(s, a_1 \dots a_j) \in Q$ pour $1 \leq j \leq k$ (par induction: si $j = 0$, $\hat{\delta}'(s,\varepsilon) = s = \hat{\delta}(s,\varepsilon) \in Q$; pour $j > 0$, en supposant - l'HI - $\hat{\delta}'(s, a_1 \dots a_{j-1}) = \hat{\delta}(s, a_1 \dots a_{j-1}) \in Q$, on obtient $\hat{\delta}'(s, a_1 \dots a_j) = \delta'(\hat{\delta}(s, a_1 \dots a_{j-1}), a_j) = \delta(\hat{\delta}(s, a_1 \dots a_{j-1}), a_j) \in Q$, par définition de δ').
Il en découle que si $k = n$, $\hat{\delta}'(s,w) = \hat{\delta}(s,w)$ et donc $\hat{\delta}(s,w) \in F$ exactement quand $\hat{\delta}'(s,w) \in F' = F$.
Si $k < n$, $a_{k+1} \in \Gamma \backslash \Sigma$. En posant $\hat{\delta}'(s, a_1 \dots a_k) = q \in Q$, on a que $\hat{\delta}'(s, a_1 \dots a_k a_{k+1}) = \delta'(q, a_{k+1}) = p$.
Mais $\hat{\delta}'(p,u) = p$ pour tout $u \in (\Sigma \cup \Gamma)^*$ (on pourrait faire une induction pour le prouver, mais on s'en passe), donc si w contient un symbole de $\Gamma \setminus \Sigma$, $\hat{\delta}'(s,w) \notin Q$, donc $\hat{\delta}'(s,w) \notin F$. Ceci termine la preuve. □

Exercice 1 *Prouvez que pour tout langage régulier il y a un nombre infini d'AFD le reconnaissant.*

Lemme 3 *Soit L_1, L_2 deux langages réguliers. Alors $L = L_1 \cup L_2$ est régulier.*

Démonstration. Soit $M_i = (Q_i, \Sigma_i, \delta_i, s_i, F_i)$ un AFD qui reconnait L_i, $i = 1, 2$. Construisons un AFD $M = (Q, \Sigma, \delta, s, F)$ qui reconnaîtra L. Sans perte de généralité (grâce au lemme 2), on peut supposer que $\Sigma = \Sigma_1 = \Sigma_2$. Pour le reste, mettons $Q = Q_1 \times Q_2,\ s = (s_1, s_2),\ F = (F_1 \times Q_2) \cup (Q_1 \times F_2)$. On définit $\delta((q_1,q_2),a) = (\delta_1(q_1,a), \delta_2(q_2,a))$. Nous vérifions que $\hat{\delta}(s,w) \in F$ si et seulement si soit $\hat{\delta}_1(s_1,w) \in F_1$, soit $\hat{\delta}_2(s_2,w) \in F_2$. En effet,

$$\hat{\delta}(s,w) = (\hat{\delta}_1(s_1,w), \hat{\delta}_2(s_2,w)) \in F = (F_1 \times Q_2) \cup (Q_1 \times F_2)$$

si et seulement si soit $\hat{\delta}_1(s_1,w) \in F_1$, soit $\hat{\delta}_2(s_2,w) \in F_2$. □

Exercice 2 *Prouver qu'avec les définitions de la preuve du lemme 3, $\hat{\delta}(s,w) = (\hat{\delta}_1(s_1,w), \hat{\delta}_2(s_2,w))$.*

Lemme 4 *Soit L_1, L_2 deux langages réguliers. Alors $L = L_1 \cap L_2$ est régulier.*

Exercice 3 *Prouver le lemme 4.*

Le lemme suivant est également utile.

Lemme 5 *Soit L_1, L_2 deux langages réguliers. Alors il existe deux AFD $M_i = (Q_i, \Sigma_i, \delta_i, s_i, F_i)$, $i = 1, 2$, tels que $L(M_i) = L_i$ et $Q_1 \cap Q_2 = \emptyset$.*

Démonstration. Puisque les deux langages sont réguliers, il y a des AFD $M_i' = (Q_i', \Sigma_i, \delta_i', s_i', F_i')$, $i = 1, 2$, tels que $L(M_i') = L_i$. On définit M_i en posant $Q_i = Q_i' \times \{i\},\ s_i = (s_i', i),\ F_i = F_i' \times \{i\}$ et $\delta_i((q,i),a) = (\delta_i'(q,a), i)$. Il est facile de voir (par récurrence) que $\hat{\delta}_i(q,w) = (\hat{\delta}_i'(q,w), i)$ pour tout $w \in \Sigma_i^*$ et donc que $\hat{\delta}_i(s_i,w) = (\hat{\delta}_i'(s_i',w), i) \in (F_i' \times \{i\}) = F_i$ si et seulement si $\hat{\delta}_i'(s_i',w) \in F_i'$. Il est évident que $Q_1 \cap Q_2 = \emptyset$. □

En ramassant Lemme 2 et Lemme 5 on obtient

Lemme 6 *Soit L_1 et L_2 deux langages réguliers. Alors il existe des AFD $M_i = (Q_i, \Sigma, \delta_i, s_i, F_i)$, $i = 1, 2$, avec le même alphabet tels que $Q_1 \cap Q_2 = \emptyset$ et $L(M_i) = L_i$.*

Exercice 4 *Prouvez ce lemme.*

Un des résultat les plus important est le théorème suivant au nom bizarre. Il permet de démontrer qu'un langage *n'est pas* régulier.

Théorème 1 *(*PUMPING LEMMA, LEMME DU POMPISTE, LEMME DE GONFLEMENT*) Soit L un langage régulier sur un alphabet Σ. Alors il existe une constante $p \in \mathbb{N}^{\geq 0}$ (longueur de pompage) telle que pour tout mot $w \in L$, $|w| \geq p$, il existe x, y, $z \in \Sigma^*$ vérifiant*

1. $w = xyz$;
2. $|xy| \leq p$;
3. $|y| > 0$;
4. $xy^iz \in L$ *pour tout* $i \geq 0$ *(*i *un entier).*

Démonstration. Soit $M = (Q, \Sigma, \delta, s, F)$ un AFD tel que $L(M) = L$. Soit p un entier, $p \geq |Q|$. Soit $w = a_1a_2 \dots a_n \in L$ ($a_i \in \Sigma$) un mot de longueur au moins p et soit $\{r_i\}_{i=0}^n$ le calcul de M sur w. Puisque $n \geq p \geq |Q|$, les états $\{r_i\}_{i=0}^n$ ne peuvent pas être tous distincts. En fait, il existe $0 \leq k < l \leq |Q| \leq p$ tels que $r_k = r_l$. Pour les fixer, prenons comme l le plus petit entier tel qu'il existe un $k < l$ avec $r_l = r_k$ (i.e. on choisit le premier état dans l'exécution de M sur w égal à un état déjà visité – ceci arrive sûrement au plus tard pour $l = |Q|$). Bien évidemment, $k < l \leq |Q| \leq p$. Posons maintenant $x = a_1 \dots a_k$, $y = a_{k+1} \dots a_l$, $z = a_{l+1} \dots a_n$. On a alors $r_l = \hat{\delta}(r_l, y) = \hat{\delta}(r_l, y^i)$, $i \in \mathbb{N}$, $\hat{\delta}(s, x) = r_k = r_l = \hat{\delta}(s, xy)$ et $\hat{\delta}(r_k, z) = r_n \in F$. On en déduit (lemme 1) que $\hat{\delta}(s, xy^iz) = \hat{\delta}(\hat{\delta}(s, x), y^iz) = \hat{\delta}(r_l, y^iz) = \hat{\delta}(\hat{\delta}(r_l, y^i), z) = \hat{\delta}(r_l, z) = r_n \in F$ pour tout entier non-négatif i. De plus, par le choix de x, y, z, $w = xyz$, $|xy| = l \leq p$ et $|y| = l - k > 0$. Donc les x, y, z vérifient les conditions du lemme. □

Soulignons encore une fois que *jamais* ce théorème ne peut démontrer qu'un langage *est* régulier.

Vu les restrictions imposées par le résultat précédent, on peut bien se demander comment élargir la classe de langages décidables par une de nos machines. Un premier essai introduit la notion du *non déterminisme.*

Définition 12 *Un* automate fini non-déterministe*, AFN, $M = (Q, \Sigma, \delta, s, F)$ est défini par*

- Q*, un ensemble fini d'*états*;*
- Σ*, un alphabet (ensemble fini) de* symboles *ou* lettres*;*
- $\delta : Q \times (\Sigma \cup \{\varepsilon\}) \longrightarrow 2^Q$*, une fonction de transition (rappel :* 2^Q *est l'ensemble des parties de* Q*);*
- $s \in Q$*, un état* intial*;*
- $F \subseteq Q$*, un ensemble d'états* acceptants*.*

Soit $M = (Q, \Sigma, \delta, s, F)$, un AFN. On défini, pour tout $q \in Q$ et tout $k \in \mathbb{N}$, l'ensemble $E_k(q)$ d'états accessibles à partir de q par des transitions spontannées en k étapes et l'ensemble $E(q)$ d'états accessible par des transitions spontannées à partir de q :

1. $E_0(q) = \{q\}$;
2. $E_{k+1}(q) = E_k(q) \cup (\cup_{p \in E_k(q)} \delta(p, \varepsilon))$ pour $k \in \mathbb{N}$;

3. $E(q) = \cup_{k=0}^{|Q|-1} E_k(q) = \cup_{k=0}^{\infty} E_k(q) = \cup_{k\in\mathbb{N}} E_k(q) = \{p \in Q : \exists k \in \mathbb{N},\ \exists p_0, \dots p_k \in Q \text{ tels que } p_0 = q,\ p_k = p,\ \text{et } p_{i+1} \in \delta(p_i, \varepsilon) \text{ pour } i = 0, \dots, k-1\}$.

Les bornes viennent du fait que si aucun nouvel état n'est ajouté à $E_k(q)$, $E_k(q) = E_l(q)$ pour tout $l \geq k$, et que si on ajoute les états un à un, cela prends $|Q| - 1$ étapes pour épuiser Q.
Avec ceci on peut définir, pour $X \subseteq Q$, l'ensemble d'états accessible par des transitions spontanées à partir des états de l'ensemble X : $E(X) = \cup_{q\in X} E(q)$. Notons que pour tout $X \subseteq Q$, $E(E(X)) = E(X)$.

Définition 13 *Soit $M = (Q, \Sigma, \delta, s, F)$, un AFN. Soit $\hat{\delta} : Q \times \Sigma^* \longrightarrow 2^Q$ l'extension de δ aux mots sur Σ, définie par*

1. $\hat{\delta}(q, \varepsilon) = E(q)$ *pour tout* $q \in Q$;
2. $\hat{\delta}(q, ua) = \cup_{p\in\hat{\delta}(q,u)} \cup_{r\in\delta(p,a)} \hat{\delta}(r, \varepsilon) = \cup_{p\in\hat{\delta}(q,u)} \cup_{r\in\delta(p,a)} E(r)$, *pour* $u \in \Sigma^*,\ a \in \Sigma$.

C'est-à-dire, $\hat{\delta}(q, w)$ est l'ensemble de tous les états dans lesquels peut se trouver M après avoir lu w, en commençant dans l'état q.

Exemple 2 $\hat{\delta}(q, a) = \cup_{p\in\hat{\delta}(q,\varepsilon)} \cup_{r\in\delta(p,a)} E(r) = \cup_{p\in E(q)} \cup_{r\in\delta(p,a)} E(r)$

Maintenant on peut définir le langage reconnu par M comme $L(M) = \{w \in \Sigma^* : \hat{\delta}(s, w) \cap F \neq \emptyset\}$. C'est-à-dire, $a_1 \dots a_k = w \in L(M)$ $(a_i \in \Sigma)$ si et seulement si il existe un $k \in \mathbb{N}$ et une suite $\{r_i\}_{i=0}^k,\ r_i \in Q$, tels que $r_0 = s,\ r_k \in F$ et $r_{i+1} \in \hat{\delta}(r_i, a_{i+1}),\ i = 0, \dots, k-1$.
On peut aussi définir une exécution de M sur $w = a_1 \dots a_k$ - c'est une suite $\{r_i\}_{i=0}^l,\ l \geq k,\ r_i \in Q$, et une suite $\{i_j\}_{j=0}^k,\ 0 \leq i_j \leq l$ telles que $i_0 = 0$, $r_{i_j+1} \in \delta(r_{i_j}, a_{j+1})$ et $r_{i+1} \in \delta(r_i, \varepsilon)$ quand $i \notin \{i_j\}_{j=0}^k$.

Soient $\mathcal{L}_{AFD}$ et $\mathcal{L}_{\mathcal{AFN}}$ les classes de langages reconnus, respectivemment, par des automates finis déterministes et nondéterministes. On a que $\mathcal{L}_{\mathcal{R}} = \mathcal{L}_{\mathcal{AFD}}$, par définition de *régulier*. Prouvons que $\mathcal{L}_{\mathcal{AFD}} = \mathcal{L}_{\mathcal{AFN}}$. Pour cela, il faut deux lemmes.

Lemme 7 *Pour tout AFD $M = (Q, \Sigma, \delta, s, F)$ il existe un AFN $M' = (Q', \Sigma', \delta', s', F')$ équivalent.*

Démonstration. On définit M' en mettant $Q' = Q,\ \Sigma' = \Sigma,\ s' = s,\ F' = F,\ \delta'(q, a) = \{\delta(q, a)\}$. On a alors que $\hat{\delta}'(q, \varepsilon) = \{q\} = E(q)$ pour tout $q \in Q$, car il n'y a pas de transitions spontanées. Ceci permet de montrer (par récurrence) que $\hat{\delta}'(q, w) = \{\hat{\delta}(q, w)\}$. En effet, pour $w = ua$, on a $\hat{\delta}'(q, ua) = \cup_{p\in\hat{\delta}'(q,u)} \cup_{r\in\delta'(p,a)} E(r) = \cup_{p\in\hat{\delta}'(q,u)} \cup_{r\in\delta'(p,a)} \{r\} = \cup_{p\in\hat{\delta}'(q,u)} \{\delta(p, a)\} = \cup_{p\in\{\hat{\delta}(q,u)\}} \{\delta(p, a)\} = \{\delta(\hat{\delta}(q, u), a)\}$ (le passage de $p \in \hat{\delta}'(q, u)$ à $p \in \{\hat{\delta}(q, u)\}$ est par l'hypthèse de récurrence). On conclut que $\hat{\delta}(s, w) \in F$ si et seulement si $\hat{\delta}'(s, w) \cap F' \neq \emptyset$ et donc $w \in L(M)$ si et seulement si $w \in L(M')$. □

On voit, encore une fois, que prouver des "évidences" demande du travail...

Lemme 8 *Pour tout AFN $M = (Q, \Sigma, \delta, s, F)$ il existe un AFD $M' = (Q', \Sigma', \delta', s', F')$ équivalent.*

Démonstration.

On définit M' par : $Q' = \{E(X) : X \in 2^Q\}$ (rappel : 2^Q est l'ensemble des parties de Q), $\Sigma' = \Sigma, s' = E(\{s\}), F' = \{X \in Q' : X \cap F \neq \emptyset\}$ et $(\delta' : Q' \times \Sigma \longrightarrow Q')$

$$\delta'(X, a) = \cup_{q\in X} \hat{\delta}(q, a).$$

Avec ceci on a, par définition, $\hat{\delta}'(X, \varepsilon) = X = E(X)$ et $\hat{\delta}'(X, ua) = \delta'(\hat{\delta}'(X, u), a)$ et on prétend que $\hat{\delta}'(X, w) = \cup_{q\in X} \hat{\delta}(q, w)$. On le prouve, comme d'habitude, par récurrence.

- Si $w = \varepsilon$, $\hat{\delta}'(X, \varepsilon) = E(X) = \cup_{q \in X} E(q) = \cup_{q \in X} \hat{\delta}(q, \varepsilon)$;
- Si $w = ua,\ u \in \Sigma^*,\ a \in \Sigma$ et $\hat{\delta}'(X, u) = \cup_{q \in X} \hat{\delta}(q, u)$, alors

$$\hat{\delta}'(X, ua) = \delta'(\hat{\delta}'(X, u), a) = \delta'(\cup_{q \in X} \hat{\delta}(q, u), a) = \cup_{q \in X} \cup_{p \in \hat{\delta}(q,u)} \hat{\delta}(p, a) =$$

$$= \cup_{q \in X} \cup_{p \in \hat{\delta}(q,u)} \cup_{r \in \delta(p,a)} E(r) = \cup_{q \in X} \hat{\delta}(q, ua) = \cup_{q \in X} \hat{\delta}(q, w).$$

En particulier, $\hat{\delta}'(s', w) = \hat{\delta}(s, w)$ et donc $\hat{\delta}'(s', w) \in F'$ si et seulement si $\hat{\delta}(s, w) \cap F \neq \emptyset$, par définition de F'. □

Remarque 1 *La preuve ci-haut donne les détails de la partie* the construction of M obviously works correctly *du livre de Sipser (p.56). Notons que la raison pour utiliser $Q' = \{E(X) : X \in 2^Q\}$ plutôt que $Q' = 2^Q$ comme dans le livre est technique : on a besoin de définir $\hat{\delta}$ de manière cohérente, i.e., on a besoin d'avoir $\hat{\delta}(X, \varepsilon) = X$ dans l'automate déterministe. Ceci ne serait pas le cas si on prenait $Q' = 2^Q$. Le livre évite les détails et, en particulier, la fonction $\hat{\delta}$, donc il peut faire la preuve "évidente". Ceci dit, la preuve du livre est ce que l'on appelerait plutôt* une esquisse d'une preuve.

Théorème 2 $\mathcal{L}_{\mathcal{R}} = \mathcal{L}_{\mathcal{AFD}} = \mathcal{L}_{\mathcal{AFN}}$

Démonstration. Ceci est une conséquence directe des deux lemmes précédents. □

Les AFN nous permettent de prouver que $L_1 \cdot L_2 \in \mathcal{L}_{\mathcal{R}}$ quand L_1, L_2 sont réguliers.

Lemme 9 *Soit L_1 et L_2 deux langages réguliers. Alors $L_1 \cdot L_2 \in \mathcal{L}_{\mathcal{R}}$.*

Démonstration. Pour $i = 1, 2$, L_i est régulier, donc il existe un AFD $M_i = (Q_i, \Sigma_i, \delta_i, s_i, F_i)$ tel que $L_i = L(M_i)$. Par le lemme 6 on peut supposer que $\Sigma_1 = \Sigma_2 = \Sigma$ et $Q_1 \cap Q_2 = \emptyset$. Soit $M = (Q_1 \cup Q_2, \Sigma, \delta, s_1, F_2)$, avec δ définie par

$$\delta(q, a) = \begin{cases} \{\delta_1(q, a)\} & \text{si } q \in Q_1, a \in \Sigma \\ \{\delta_2(q, a)\} & \text{si } q \in Q_2, a \in \Sigma \\ \{s_2\} & \text{si } q \in F_1, a = \varepsilon \end{cases}$$

Il est facile de voir (prouvez le!) que M se comporte comme M_1 tant qu'il n'est pas dans un état acceptant de M_1, auqel cas il peut choisir de transférer le contrôle à M_2. Donc M accepte exactement les mots sur Σ qui sont des concaténations d'un mot de L_1 et un mot de L_2. □

Exercice 5 *Reprouvez le lemme avec des AFN plutôt que AFD.*

Exercice 6 *Prouvez que si L est un langage régulier sur Σ, alors L^* l'est aussi.*

Il y a une autre manière de définir la même classe de langages qui a l'agréable propriété de rendre certaines preuves beacoup plus faciles (et d'autres presque impossible!).

Définition 14 *Soit Σ un alphabet. Une* expression régulière sur Σ *est définie récursivement :*

1. *$\emptyset$ est une expression régulière;*
2. *ε est une expression régulière;*
3. *a est une expression régulière pour tout $a \in \Sigma$;*
4. *si R_1 et R_2 sont des expressions régulières alors $(R_1 \cdot R_2)$ est une expression régulière;*

5. *si R_1 et R_2 sont des expressions régulières alors $(R_1 + R_2)$ est une expression régulière;*

6. *si R est une expression régulière alors (R^*) est une expression régulière.*

Exemple 3 *Soit $\Sigma = \{ab, b, c, d\}$. Alors ε, $\emptyset$, $\varepsilon + \varepsilon$, $\varepsilon \cdot \emptyset$, a, $(((a + b) \cdot (c \cdot a))^*)$ sont des expressions régulières.*

Convention 2 *Soit Σ un alphabet. On adopte des conventions (abrégés) suivants:*

- *On omet les parenthèses et le symbole de concaténation quand l'expression n'est pas ambigue. Par exemple, on écrit $(abc)+(a+c)c$ plutôt que $(((a\cdot b)\cdot c)+((a+c)\cdot c))$ ou $(((a\cdot(b\cdot c))+((a+c)\cdot c)))$;*
- *La concaténation a précedence sur la somme. Par exemple, on écrit $abc + (a + c)c$ pour $(abc) + (a + c)c$;*
- *Pour $\Sigma = \{a_1, \ldots, a_n\}$ on écrit Σ pour $a_1 + a_2 + \ldots a_n$;*
- *Pour $n \in \mathbb{N}$ et $u \in \Sigma^*$ on écrit u^n pour $uu \ldots u$ (n fois). Par extension, si R est une expression régulière, on peut se permettre des abréviations comme $(L(R))^*$.*

Comme avec les automates finis, on définit les langages associés aux expressions régulières. Avec la définition récursive d'une expression régulière, ceci est bien plus simple que les définitions correspondantes pour les automates finis.

Définition 15 *Soit Σ un alphabet et soient R, R_1, R_2 des expressions régulières sur Σ. L'*ensemble décrit par *R où le* langage représenté par *R $L(R)$ est défini récursivement par :*

1. *Si $R = \emptyset$, alors $L(R) = \emptyset$;*
2. *Si $R = \varepsilon$, alors $L(R) = \{\varepsilon\}$;*
3. *Si $R = a$, $a \in \Sigma$, alors $L(R) = \{a\}$;*
4. *Si $R = (R_1 + R_2)$, alors $L(R) = L(R_1) \cup L(R_2)$;*
5. *Si $R = (R_1 \cdot R_2)$, alors $L(R) = L(R_1) \cdot L(R_2)$;*
6. *Si $R = (R_1)^*$, alors $L(R) = (L(R_1))^*$;*

On va prouver qu'un langage est régulier si et seulement si il est décrit par une expression régulière. Il nous faut deux lemmes.

Lemme 10 *Pour tout une expression régulière R sur un alphabet Σ, $L(R)$ est un langage régulier.*

Démonstration. On procède par récurrence sur R. L'alphabet sera Σ dans chacun des cas. Dans chaque cas on dit comment construire un AFN M_R tel que $L(M_R) = L(R)$

1. Si $R = \emptyset$, alors $L(R) = \emptyset$ et l'automate $M_R = (Q, \Sigma, \delta, s, F)$ est défini par $Q = \{s\}$, $F = \emptyset$, et $\delta(s, a) = s$ pour tout $a \in \Sigma$;
2. Si $R = \varepsilon$, alors $L(R) = \{\varepsilon\}$ et l'automate $M_R = (Q, \Sigma, \delta, s, F)$ est défini par $Q = \{s, p\}$, $F = \{s\}$, et $\delta(s, a) = \delta(p, a) = p$ pour tout $a \in \Sigma$;
3. Si $R = a$, $a \in \Sigma$, alors $L(R) = \{a\}$ et l'automate $M_R = (Q, \Sigma, \delta, s, F)$ est défini par $Q = \{s, p, q\}$, $F = \{q\}$, et $\delta(s, a) = q$, $\delta(s, x) = \delta(q, y) = \delta(p, y) = p$ pour tout $x, y \in \Sigma, x \neq a$;

4. Si $R = (R_1 + R_2)$, alors $L(R) = L(R_1) \cup L(R_2)$ et, par l'hypothèse de récurrence, les deux langages sont réguliers. Par le lemme 3, $L(R_1) \cup L(R_2)$ est un langage régulier;

5. Si $R = (R_1 \cdot R_2)$, alors $L(R) = L(R_1) \cdot L(R_2)$. Ce dernier langage est régulier par le lemme 9 car L_1 et L_2 le sont par l'hypothèse.

6. Si $R = (R_1)^*$, alors $L(R) = (L(R_1)^*$; ce langage est régulier par l'exercice 6 et l'hypthèse de récurrence.

Lemme 11 *Soit L un langage régulier sur Σ. Alors il existe une expression régulière R sur Σ telle que $L(R) = L$.*

Démonstration. Soit $M = (Q, \Sigma, \delta, s, F)$ que l'on sait existe. Soit $Q = \{q_0, q_1, \ldots, q_n\}$ avec $s = q_0$. On s'en servira pour construire une expression régulière qui décrit $L(M) = L$ en $|Q|+1$ étapes. A la k-ème étape on produira, pour chaque paire d'états (q_i, q_j) (une *paire* est ordonnée et les deux états peuvent être égaux), une expression régulière R_{ij}^k qui décrit l'ensemble de mots permettant de passer de q_i à q_j **sans passer par** un état numéroté plus que k. Plus formellement, R_{ij}^k est l'expression régulière telle que $L(R_{ij}^k) = \{w = a_1 \ldots a_m \in \Sigma^* : \hat{\delta}(q_i, w) = q_j$ et pour $1 < t \leq m,\ \hat{\delta}(q_i, a_1 \ldots a_{t-1}) \in \{q_0, \ldots, q_k\}\}$. Notons que la description formelle montre le sens de "passer par": il faut entrer et sortir.

On commence par $k = -1$. On a que $L(R_{ij}^{-1})$ est l'ensemble de mots permettant d'aller de q_i à q_j en ne passant que par des états dans $\{q_0, \ldots, q_{-1}\} = \emptyset$, c'est-à-dire, directement. Donc $L(R_{ij}^{-1}) = \{a \in \Sigma : \delta(q_i, a) = q_j\}$ et $R_{ij}^{-1} = +_{a \in \Sigma : \delta(q_i,a) = q_j} a = +\{a \in \Sigma : \delta(q_i, a) = q_j\}$ (voir la remarque après la preuve).

Supposons que nous avons déjà les R_{ij}^l pour $l = -1, \ldots, k$. On a alors que

$$R_{ij}^{k+1} = R_{ij}^k + R_{i(k+1)}^k \cdot (R_{(k+1)(k+1)}^k)^* \cdot R_{(k+1)j}^k$$

car pour aller de q_i à q_j en passant uniquement par les états dans $\{q_0, \ldots, q_{k+1}\}$, on peut soit ne pas passer par l'état q_{k+1} (i.e. on lit des mots dans $L(R_{ij}^k)$), soit passer par q_{k+1}, mais alors on peut décomposer tout w sur lequel l'exécution de l'automate sur passe par q_{k+1} en partie délimitées par les passages par q_{k+1}. Ceci donne (1) les mots qui mènent de q_i vers q_{k+1} pour la première fois (i.e. des mots dans $L(R_{i(k+1)}^k)$), (2) les mots menant de q_{k+1} à q_{k+1} sans passer par $\{q_{k+1}, \ldots, q_n\}$(i.e. les mots dans $L((R_{(k+1)(k+1)}^k)^*)$, car on peut y passer autant de fois que l'on veut une fois un passage existe), et (3) les mots qui permettent d'aller de q_{k+1} à q_j en ne passant que par les états dans $\{q_0, \ldots, q_k\}$ (i.e. les mots dans $L(R_{(k+1)j}^k)$. En mettant le tout ensemble, on obtient l'expression voulue. Le fait que $L(R_{ij}^{k+1}) = \{w = a_1 \ldots a_m \in \Sigma^* : \hat{\delta}(q_i, w) = q_j$ et pour $1 < t \leq m,\ \hat{\delta}(q_i, a_1 \ldots a_{t-1}) \in \{q_0, \ldots, q_{k+1}\}\}\}$ suit par récurrence. Il est (ou doit être) clair que l'exécution de M sur $w \in L(R_{ij}^{k+1})$ ne passe par aucun état numéroté plus que $k+1$, par construction. Le reste suit car $\hat{\delta}(q, uv) = \hat{\delta}(\hat{\delta}(q, u), v)$, comme on a vu déjà.

On construit donc les expressions régulières R_{ij}^k pour $i, j \in \{0, \ldots, |Q|\}$ et $k = \{-1, 0, 1, 2, \ldots, |Q|\}$ et ceci de manière à ce que $w \in L(R_{ij}^k)$ si et seulement si $\hat{\delta}(q_i, w) = q_j$ et aucun état intermédiare de l'exécution de M sur w n'est numéroté plus que k. On conclut que $L(R_{ij}^n) = \{w \in \Sigma^* : \hat{\delta}(q_i, w) = q_j\}$. En particulier, $w \in L$ si est seulement si $\hat{\delta}(q_0, w) \in F$, c'est-à-dire, il existe un j tel que $q_j \in F$ et $w \in L(R_{0j}^n)$. Mais alors

$$L = L(+_{j : q_j \in F} R_{0j}^n)$$

et l'expression régulière recherchée est

$$+_{j : q_j \in F} R_{0j}^n$$

ce qui termine la preuve. □

Remarque 2 *Dans la preuve nous avons utilisé la notation* $R = +_{j:q_j \in F} R^n_{0j}$. *Ceci est une notation utile car elle remplace*

Soit $F = \{q_{i_1}, q_{i_2}, \ldots, q_{i_r}\}$. *Alors* $R = R^n_{0i_1} + R^n_{0i_2} \ldots + R^n_{0i_r}$.

L'économie d'expression est encore plus grande dans la formulation $+_{a \in \Sigma : \delta(q_i, a) = q_j} a$ *plutôt que*

Soit $\{a_1, \ldots, a_r\} = \{a \in \Sigma : \delta(q_i, a) = q_j\}$. *Alors* $R^{-1}_{ij} = a_1 + a_2 + \ldots + a_r$.

où, strictement, on devrait indexer les a en plus de numéro par les paires des états!

Chaque fois que l'on construit un automate fini déterministe, on peut - devrait - se poser la question *est-ce qu'il y a un AFD équivalent avec moins d'états?*. On va donner une méthode pour répondre à cette question. On se rappelle les notions suivantes.

Définition 16 *Soit* X *un ensemble.*

- *Une* relation binaire R *sur* X *est une partie de* $X \times X$. *On écrit souvent* xRy *(parfois* $x \sim_R y$*) pour* $(x, y) \in R$.
- *Une relation binaire* R *sur* X *est une* relation d'équivalence *si elle est*
  - réflexive *:* xRx *pour tout* $x \in X$*;*
  - symétrique *: si* xRy *alors* yRx*;*
  - transitive *: si* xRy *et* yRz *alors* xRz.
- *Soient* $x \in X$ *et* R *une relation d'équivalence sur* X. *La* classe d'équivalence *de* x*, noté* $[x]_R$ *est l'ensemble* $\{y \in X : xRy\}$.
- *Une* partition *de* X *est une famille* $\Pi = \{X_i\}_{i \in I}$ *de parties de* X*,* I *un ensemble d'indices, telle que* $\cup_{i \in I} X_i = X$ *et* $X_i \cap X_j = \emptyset$ *pour* $i, j \in I,\ i \neq j$.

Exercice 7 *Soit* X *un ensemble et soit* R *une relation d'équivalence sur* X. *Prouvez que* $[x]_R = [y]_R$ *si et seulement si* xRy.

Exercice 8 *Soit* X *un ensemble et soit* Π *une partition de* X. *Prouvez que pour tout* $X, Y \in \Pi$ *soit* $X = Y$ *soit* $X \cap Y = \emptyset$.

Lemme 12 *Soit* X *un ensemble.*

1. *Soit* R *une relation d'équivalence sur* X. *Alors les classes d'équivalence de* R *forment une partition de* X.
2. *Soit* $\{X_i\}_{i \in I}$ *une partition de* X. *Alors* R*, définie par* xRy *si et seulement si il existe un* $i \in I$ *tel que* $x, y \in X_i$*, est une relation d'équivalence et ses classes d'équivalence sont exactement les ensembles* X_i*,* $i \in I$.

Démonstration.

1. On prend $I = X$. Il est clair que $X = \cup_{x \in X} \{x\} = \cup_{x \in X} [x]_R$ car $x \in [x]_R$ pour tout $x \in X$. Par l'exercice 7, $[x]_R = [y]_R$ si et seulement si xRy. Si $[x]_R \neq [y]_R$ alors $[x]_R \cap [y]_R = \emptyset$, car sinon, il y aurait un $z \in [x]_R \cap [y]_R$ et, par transitivité et symétrie, on aurait que xRy, donc $[x]_R = [y]_R$. Donc les classes d'équivalence de R partitionnent X.

2. D'abord, pour tout $x \in X$, x appartient un des ensembles X_i, $i \in I$. On a alors xRx (réflexivité). Si x et y appartiennent à X_i, alors xRy et yRx, par définition (symétrie). Si x et y appartiennent à X_i (xRy) et y et z appartiennent à X_j (yRz), alors $i = j$ et x et z sont dans X_i, donc xRz. Pour les classes d'équivalences, on a que $[x]_R = \{y \in X : xRy\} = \{y \in X : y \in X_i \text{ tel que } x \in X_i\} = X_i$.

□

La dernière idée dont on a besoin sur les relations d'èquivalence est celle de raffinement. Une relation d'équivalence R sur X partitionne l'ensemble X, c'est-à-dire, elle découpe le gâteau X en morceaux. Si une autre relation d'équivalence E découp X de manière à ce qu'elle ne découpe que le morceaux déjà obtenus par R, on dit que E raffine R. Formellement:

Définition 17 *Soit X un ensemble et soient R, E deux relations d'équivalence sur X. On dit que E* raffine *R si pour tout $x \in X$, $[x]_E \subseteq [x]_R$*

De manière équivalente, E raffine R si pour tout $x, y \in X$, xRy quand xEy.

Soit $M = (Q, \Sigma, \delta, s, F)$ un AFD. Soit R_M la relation d'équivalence sur Σ^* définie par $(x, y) \in R_M$ si $\hat{\delta}(s, x) = \hat{\delta}(s, y)$, c'est-à-dire, à partir de l'état initial on arrive au même état en lisant x et en lisant y. Bien évidemment, si $(x, y) \in R_M$, alors pour tout $z \in \Sigma^*$, alors $\hat{\delta}(s, xz) = \hat{\delta}(s, yz)$. On vient de prouver que R_M est une relation d'équivalence invariante à droite (voir plus bas). On peut dire plus: le nombre de classes d'équivalence de R_M est égal au nombre d'états de M. En effet, les classes d'équivalence de R_M sont tout simplement C_q, $q \in Q$, avec $C_q = \{x \in \Sigma^* : \hat{\delta}(s, x) = q\}$.

Définition 18 *Soit Σ un alphabet et R une relation d'équivalence sur Σ^*. On dit que*

1. *R est* invariante à droite *si quand xRy pour $x, y \in \Sigma^*$, alors pour tout z on a $xzRyz$;*
2. *R est* d'index fini *si le nombre de classes d'équivalence de R est fini.*

On peut définir une autre relation utile sur Σ^*. Soit $L \subseteq \Sigma^*$. Définissons R_L sur Σ^* en disant que $(x, y) \in R_L$ si pour tout $z \in \Sigma^*$, $xz \in L$ si et seulement si $yz \in L$. Celle-ci est également invariante à droite car si $(x, y) \in R_L$ et $w \in \Sigma^*$, alors pour tout $z \in \Sigma^*$ on a que $x(wz) = (xw)z \in L$ si et seulement si $y(wz) = (yw)z \in L$. Les classes d'équivalence de R_L sont donc $[x] = \{w \in \Sigma^* : (x, w) \in R_L\} = \{w \in \Sigma^* | xz \in L \text{ si et seulement si } wz \in L \text{ pour tout } z \in \Sigma^*\}$.

Exercice 9 *Prouvez que R_M et R_L sont des relations d'équivalence et vérifiez que les classes d'équivalence sont celles décrites plus haut.*

Quand L est régulier, la relation R_L nous donne un moyen de construire l'automate fini déterministe unique M_L ayant le nombre d'états minimum et tel qe $L(M) = L$. La base de la construction est le théorème suivant.

Théorème 3 *(*Myhill-Nerode*) Soit Σ un aphabet et $L \subseteq \Sigma^*$. Alors les énoncés suivants sont équivalents:*

1. *L est un langage régulier;*
2. *L est la réunion de (certaines) classes d'équivalence d'une relation d'équivalence sur Σ^* invariante à droite et d'index fini;*
3. *la relation R_L est d'index fini.*

Démonstration. On prouve (1) $\Longrightarrow$ (2) $\Longrightarrow$ (3) $\Longrightarrow$ (1).

- **(1) $\Longrightarrow$ (2)**
  On a prouvé ceci plus haut: soit M un AFD tel que $L(M) = L$. Alors la relation R_M est invariante à droite, d'index fini, et on a que $L = \cup_{x\in L}[x] = \cup_{q\in F}C_q$.

- **(2) $\Longrightarrow$ (3)**
  Nous prouvons que toute relation E vérifiant (2) raffine R_L. Puisque le nombre de classes d'équivalence (l'index) de E est fini, le nombre de clases d'équivalence de R_L l'est aussi (E a au moins autant de clases d'équivalence que R_L). Soit donc E un relation d'équivalence vérifiant (2) et soit $x, y \in \Sigma^*$ tels que xEy. Pour tout $z \in \Sigma^*$, $xzEyz$ (car E est invariante à droite), c'est-à-dire,xz et yz appartiennent à la mème classe. Donc $xz \in L$ si et seulement si $yz \in L$, c'est-à-dire, $(x, y) \in R_L$.

- **(3) $\Longrightarrow$ (1)**
  Nous allons définir un automate fini déterministe M_L tel que $L(M_L) = L$. Pour $x \in \Sigma^*$ mettons $[x] = [x]_{R_L}$. Soit $M_L = \{Q, \Sigma, \delta, s, F\}$ défini par
  $Q = \{[x] : x \in \Sigma^*\}$
  $s = [\varepsilon]$
  $F = \{[x] : x \in L\}$
  $\delta([x], a) = [xa]$

Il nous reste a prouver que la définiton est bonne et que $L(M) = L$. Pour la première partie, si $[x] = [y]$ alors $xz \in L$ si et seulement si $yz \in L$ pour tout $z \in \Sigma^*$. Donc pour tout $a \in \Sigma$, $xaz \in L$ si et seulement si $yaz \in L$, et alors $[xa] = [ya]$. Pour la deuxième, on observe (prouvez-le!) que $\hat{\delta}([x], y) = [xy]$ pour tout $x, y \in \Sigma^*$, et donc $\hat{\delta}(s, w) = \hat{\delta}([\varepsilon], w) = [w]$. On a alors que $w \in L(M)$ si et seulement si $[w] \in F$ si et seulement si $w \in L$. □

Le théorème nous permet de construire l'automate minimal pour un langage régulier. Un automate fini déterministe $M = (Q, \Sigma, \delta, s, F)$ est *minimal* si aucun autre AFD équivalent $M' = (Q', \Sigma, \delta', s', F')$ n'a moins d'états, i.e. si $|Q| \leq |Q'|$. On dit que M est *minimal pour* L si $L(M) = L$ et M est minimal.

Définition 19 *Soit $M = (Q, \Sigma, \delta, s, F)$ et $M' = (Q', \Sigma, \delta', s', F')$ deux automates. On dit que M et M' sont* isomorphes *s'il existe une bijection $\phi : Q \longrightarrow Q'$ telle que $\phi(s) = s'$, $F' = \{\phi(q) : q \in F\}$ et pour tout $a \in \Sigma$ et tout $q \in Q$ on a $\delta'(\phi(q), a) = \phi(\delta(q, a))$.*

Ceci est une manière formelle de dire que la seule chose qui change entre M et M' sont les noms des états (c'est ϕ qui change les noms). Il devrait être claire que deux automates isomorphes sont équivalents (acceptent le même langage).

Exercice 10 *1. Pourquoi faut-il insister sur $\phi(s) = s'$ et $F' = \{\phi(q) : q \in F\}$? Donner des exemples où l'un ou l'autre est absent, le reste est vérifié, et les automates ne sont pas isomorphes.*

2. Est-ce que la définition suivante est équivalente a la définition 19? Prouvez votre réponse.

Définition 20 *Soit $M = (Q, \Sigma, \delta, s, F)$ et $M' = (Q', \Sigma, \delta', s', F')$ deux automates tels que $L(M) = L(M')$. On dit que M et M' sont* isomorphes *s'il existe une bijection $\phi : Q \longrightarrow Q'$ telle que pour tout $a \in \Sigma$ et tout $q \in Q'$ on a $\delta'(\phi(q), a) = \phi(\delta(q, a))$.*

Théorème 4 *Soit L un langage régulier sur l'alphabet Σ. Alors l'automate fini déterministe M_L construit dans la preuve du théorème de Myhill et Nerode est l'AFD minimum pour L et il est unique à l'isomorhisme près.*

Démonstration. Soit $M_L = (Q_L, \Sigma, \delta_L, s_L, F_L)$ l'automate construit dans la preuve de "(3) $\Longrightarrow$ (1)" du théorème de Myhill - Nerode. On a vu dans la preuve de "(2) $\Longrightarrow$ (3)" du théorème 3 que si une relation E vérifie (2), alors elle raffine R_L. On en déduit que si $M = (Q, \Sigma, \delta, s, F)$ est un automate qui accepte L, alors $|Q_L| \leq |Q|$. Si M est un automate minimum, alors $|Q_L| \geq |Q|$, donc $|Q_L| = |Q|$, et on vois facilement que pour tout $q \in Q$ il existe un mot $w_q \in \Sigma^*$ tel que $\hat{\delta}(s, w_q) = q$ (sinon q ne peut jamais être atteint, donc il ne sert a rien, et l'automate obtenu de M en enlevant q accepterait toujours L mais avec moins d'états que M_L, ce qui n'est pas possible).
Il faut maintenant exhiber une bijection ϕ entre Q et Q_L telle que pour tout $a \in \Sigma$ et tout $q \in Q$ on a $\delta_L(\phi(q), a) = \phi(\delta(q, a))$. On définit $\phi : Q \longrightarrow Q_L$ par $\phi(q) = [w_q]$. La fonction est bien définie: si $\hat{\delta}(s, w_q) = \hat{\delta}(s, w'_q)$ alors $[w_q]_{R_L} = [w'_q]_{R_L}$. De plus, ϕ est surjective car pour $u \in \Sigma^*$, $[w_{\hat{\delta}(s,u)}] = [u]$ et $\phi(\hat{\delta}(s, u)) = [u]$. Puisque $|Q_L| = |Q|$, ϕ est une bijection[1]. Pour terminer, on observe que pour tout $a \in \Sigma$, $\delta_L(\phi(q), a) = \delta_L([w_q]_L, a) = [w_q a]_L = \phi(\hat{\delta}(s, w_q a)) = \phi(\delta(q, a))$. □

Il est instructif de réfléchir au fait qu'un isomorphisme est une relation à deux sens (en effet, c'est une relation d'équivalence sur la classe des AFD). La bijection $\phi : Q \longrightarrow Q_L$ doit avoir un inverse, disons $\phi^{-1} = \psi : Q_L \longrightarrow Q$ qui est également un isomorphisme. Mais si on fait ce qu'il faut, c'est-à-dire, si on définit $\psi([w]) = \hat{\delta}(s, w)$, prouver que la définition est bonne et que ψ est un isomorphisme demande un peu plus de travail. Ce travail vaut la peine car il explique pourquoi l'algorithme de minimisation d'un AFD marche. Faisons le.

Soit $M = (Q, \Sigma, \delta, s, F)$ un automate minimum qui décide L. Soit $u, v \in \Sigma^*$ tels que $[u]_{R_L} = [v]_{R_L}$ mais $q_u = \hat{\delta}(s, u) \neq \hat{\delta}(s, v) = q_v$. Sans perte de généralité, soit $q_v \neq s$. Alors l'automate $M' = (Q', \Sigma, \delta', s', F')$ défini par $Q' = Q \setminus \{q_v\}$, $F' = F \setminus \{q_v\}$, $s' = s$, et

$$\delta'(q, a) = \begin{cases} \delta(q, a) \text{ si } \delta(q, a) \neq q_v \\ q_u \text{ sinon.} \end{cases}$$

Ce nouvel automate décide L mais a un état de moins, ce qui contredit la minimalité de M. Pour voir que $L = L(M) = L(M')$, on se souvient que $[u] = [v]$, c'est-à-dire, $uz \in L$ si et seulement si $vz \in L$ pour tout $z \in \Sigma^*$. On a alors que $\hat{\delta}'(q_u, z) \in F'$ si et seulement si $\hat{\delta}'(q_v, z) \in F'$ pour tout $z \in \Sigma^*$. En en déduit que pour x, y tels que $\hat{\delta}(s, x) = q_u$ et $\hat{\delta}(s, y) = q_v$, $[x] = [y] = [u] = [v]$ et que $\hat{\delta}'(s', x) = q_u = \hat{\delta}(s', y)$. En effet, M' est obtenu à partir de M en identifiant q_u et q_v (et en enlevant les transition partant de q_v). Puisque M' mène à une contradiction, il ne peut pas exister. On en déduit que dans M, on a $q_u = \hat{\delta}(s, u) = \hat{\delta}(s, v) = q_v$ pour u, v tels que $[u] = [v]$. Ceci prouve tout de suite que ψ est bien défini.

Pour voir que ψ est un isomorphisme : $\psi(\delta_L([w], a) = \psi([wa]) = \hat{\delta}(s, wa) = \hat{\delta}(\delta(s, w), a) = \hat{\delta}(\psi[w], a)$. □

Ce que l'on vient de prouver peut servir a trouver un automate détermniste minimum pour chaque langage régulier. Nous avons vu comment faire si le langage L est donné : les classes d'équivalence de R_L deveinnent les états de l'unique AFD minimal M_L. Avec un peu de réflexion, on peut minimisier un AFD donné.

Soit $M = (Q, \Sigma, \delta, s, F)$ un AFD donné et soit R_M la relation d'équivalence sur Σ^* induite par M (rappel : $uR_M v$ si $\hat{\delta}(s, u) = \hat{\delta}(s, v)$). Nous avons vu dans la preuve du théorème 3, (2)$\Longrightarrow$(3), que – en particulier – R_M raffine R_L, i.e. toute classe d'équivalence de R_M (que l'on peut identifier à un état de M) est contenu dans une classe d'équivalence de R_L (état de M_L). Afin de minimiser l'automate, il faut alors identifier (dans les deux sens du mot) les états de M correspondant à une même classe de R_L. Comment faire?

La question devient : comment peut-on décider si deux classes de R_M (états de M) appartiennent à la même classe d'èquivalence de R_L? Un peu de notation d'abord. On va écrire $p \equiv q$ si on a

[1] Un lemme à prouver si vous ne le connaissez pas : *Soient X, Y deux ensembles finis avec $|X| = |Y|$. Si $f : X \longrightarrow Y$ est injective ou surjective, elle est bijective.* Notons que ceci n'est pas vrai si les ensembles sont infinis.

$\hat{\delta}(p,x) \in F$ si et seulement si $\hat{\delta}(q,x) \in F$ pour tout $x \in \Sigma^*$ (donc pour toute paire de mots u, v tels que $\hat{\delta}(s,u) = p$ et $\hat{\delta}(s,v) = q$ on aura $\hat{\delta}(s,ux) \in F$ si et seulement si $\hat{\delta}(s,vx) \in F$ pour tout $x \in \Sigma^*$, i.e. uR_Lv).

Commme souvent, il est plus simple de décider si $p \not\equiv q$ car pour cela il suffit de trouver UN mot x tel que $\hat{\delta}(p,x) \in F$ tandis que $\hat{\delta}(q,x) \notin F$ (ou inversement). Soit $x = a_1 \dots a_k$. Si, sans perte de généralité, $\hat{\delta}(p,x) \in F$ et $\hat{\delta}(q,x) \notin F$, alors bien évidemment, pour tout $1 \leq i \leq k$, $\hat{\delta}(p, a_1 \dots a_i) \not\equiv \hat{\delta}(q, a_1 \dots a_k)$ (car $\hat{\delta}(\hat{\delta}(p, a_1 \dots a_i), a_{i+1} \dots a_k) \in F$ et $\hat{\delta}(\hat{\delta}(q, a_1 \dots a_i), a_{i+1} \dots a_k) \notin F$). En fait, l'implication va dans les deux sens. On conclut que $p \not\equiv q$ si et seulement si il existe un mot $x \in \Sigma^*$ tel que $\hat{\delta}(p,x) \not\equiv \hat{\delta}(q,x)$. En particulier, puisque M est fini, $p \not\equiv q$ si et seulement si $\hat{\delta}(p,x) \not\equiv \hat{\delta}(q,x)$ pour un $x \in \Sigma^*$, $|x| \leq |Q|$. L'observation clé est que si $p \not\equiv q$ parce que $x = a_1 \dots a_k$ est tel que $\hat{\delta}(p,x) \in F$ et $\hat{\delta}(q) \notin F$ (sans perte de g'enéralité), alors pour tout $1 \leq i \leq k$, $\hat{\delta}(p, a_1 \dots a_i) \not\equiv \hat{\delta}(q, a_1 \dots a_i)$ parce que $\hat{\delta}(\hat{\delta}(p, a_1 \dots a_i), a_{i+1} \dots a_k) = \hat{\delta}(p,x) \not\equiv \hat{\delta}(q,x) = \hat{\delta}(\hat{\delta}(q, a_1 \dots a_i), a_{i+1} \dots a_k)$. On renverse donc la procédure. D'abord, $p \not\equiv q$ pour $p \in F$ et $q \in Q \setminus F$. Ensuite, si $p' \neq q'$ et $\delta(p',a) = p \in F$, $\delta(q',a) = q \in Q \setminus F$ pour un $a \in \Sigma$, alors $p' \not\equiv q'$. En général, si pour deux états p', q' il existe un $a \in \Sigma$ tel que $\delta(p',a) \not\equiv \delta(q',a)$, alors $p' \not\equiv q'$.

Ceci donne un algorithme de minimisation. Supposons que l'automate $M = (Q, \Sigma, \delta, s, F)$ a $Q = \{q_0, q_1, \dots, q_{n-1}\}$ et $s = q_0$. Puisque les états inaccessibles à partir de s (i.e. les états q tel que $\hat{\delta}(s,w) \neq q$ pour tout $w \in \Sigma^*$) ne servent à rien, on peut les enlever et travailler uniquement avec les états accessible. On supppose dans la suite que pour tout $q \in Q$ il existe un $w_q \in \Sigma^*$ tel que $\hat{\delta}(s, w_q) = q$.

Exercice 11 *Comment enlever les états inaccessibles? Donnez un algorithme qui prend en entrée un AFD $M = (Q, \Sigma, \delta, q_0, F)$ et donne en sortie un AFD $M' = (Q', \Sigma, \delta', q_0, F')$ équivalent et tel que pour tout $q \in Q$ il existe un $w_q \in \Sigma^*$ tel que $\hat{\delta}(q_0, w_q) = q$. Pourquoi q_0 reste?*

Supposons toujours que $Q = \{q_0, \dots, q_{n-1}\}$ avec $s = q_0$. La version naïve de l'algorithme est toute simple. On ne prends pas en compte la symétrie de $\equiv$ et $\not\equiv$ afin de simplifier la suite.

```
Pour q_i ∈ F et q_j ∈ Q \ F, q_i ≢ q_j et q_j ≢ q_i
Pour k = 0..n − 1 faire
      Pour i = 0..n − 1 faire
            Pour j = 0..n − 1 faire
                  Pour tout a ∈ Σ faire
                        Si δq_i, a) ≢ δ(q_j, a) alors q_i ≢ q_j
```

Les paires d'états équivalents peuvent alors être identifiés pour en faire un seul. Pour faire plus pratique, on peut prendre un tableau $T[0..\ n-1, 0..n-1]$ initialisé à $\equiv$ partout et faire tourner l'algorithme pour remplacer certains $\equiv$ par $\not\equiv$.

```
Pour i = 0..n − 1 faire
   Pour j = i + 1..n − 1 faire
         Si q_i ∈ F et q_j ∉ F alors T[i, j] ← ≢
         Si q_i ∉ F et q_j ∈ F alors T[i, j] ← ≢

Pour k = 0..n − 1 faire
   Pour i = 0..n − 1 faire
         Pour j = 0..n − 1 faire
               Pour tout a ∈ Σ faire
                     Si δq_i, a) ≢ δ(q_j, a) alors q_i ≢ q_j
```

Pour obtenir l'automate minimal, il suffit de définir, pour tout $i \in \{0, \ldots, n-1\}$, $Q_i = \{q_j \in Q : q_j \equiv q_i\}$ (l'information nécessaire est dans T) et un nouvel automate $M^* = (\{Q_i : i \in \{0, \ldots n-1\}, \Sigma, \delta^*, Q_0, \{Q_i : q_i \in F\})$ où $\delta^*(Q_i, a) = Q_j$ tel que $\delta(q_i, a) = q_j$.

Exercice 12 *Prouver que δ^* est bien définie et que $L(M^*) = L(M)$*

Exemple 4 *Soit $M = (Q, \Sigma, \delta, s, F)$ l'AFD défini par $\Sigma = \{a, b, c\}$, $Q = \{q_0, \ldots, q_8\}$, $s = q_0$, $F = \{q_0, q_4, q_8\}$ et*

	a	b	c
$\mathbf{q_0}$	q_1	q_1	q_5
q_1	q_2	q_6	q_6
q_2	q_3	q_7	q_3
q_3	q_4	q_8	q_0
$\mathbf{q_4}$	q_5	q_1	q_1
q_5	q_6	q_2	q_6
q_6	q_7	q_3	q_3
q_7	q_8	q_0	q_4
$\mathbf{q_8}$	q_5	q_1	q_5

et soit $L = L(M)$.

Dans le tableau suivant on met dans la case (i, j) soit rien, si les états q_i et q_j peuvent être équivalent (i.e. si on ne peut pas les distinguer par un mot sur Σ), soit k, x si à la $k-$ème itération on a, pour un $x \in \Sigma \cup \{\varepsilon\}$, $\delta(q_i, x) = q_{i'}$, $\delta(q_j, x) = q_{j'}$ et $q_{i'} \not\equiv q_{j'}$, i.e. si la case i', j' n'est pas vide. On commence par l'initialisation, itération 0, pendant laquelle on remplit les cases (i, j) telles que $q_i \in F$ et $q_j \not\in F$ ou $q_j \in F$ et $q_i \notin F$. On fait au plus $|Q|$ itérations car si deux états peuvent être distingués par un mot $u \in \Sigma^*$, ils peuvent l'être par un mot de longueur au plus $|Q| - 1$. On peut, bien sûr, arrêter avant - si après une itération le tableau n'a pas changé, il ne changera plus jamais. Il n'est pas nécessaire de remplir le tout, les cases (i, j) avec $j > i$ sont suffisantes (les états acceptants sont en **gras***)*

8									
7									$0, \varepsilon$
6								$1, a$	$0, \varepsilon$
5							$2, a$	$1, a$	$0, \varepsilon$
4						$0, \varepsilon$	$0, \varepsilon$	$0, \varepsilon$	
3					$0, \varepsilon$	$1, a$	$1, a$		$0, \varepsilon$
2				$1, a$	$0, \varepsilon$	$2, a$		$1, a$	$0, \varepsilon$
1			$2, a$	$1, a$	$0, \varepsilon$		$2, a$	$1, a$	$0, \varepsilon$
0		$0, \varepsilon$	$0, \varepsilon$	$0, \varepsilon$		$0, \varepsilon$	$0, \varepsilon$	$0, \varepsilon$	
	0	1	2	3	**4**	5	6	7	**8**

Conclusion: $q_0 \equiv q_4 \equiv q_8$, $q_1 \equiv q_5$, $q_2 \equiv q_6$, $q_3 \equiv q_7$ *e l'automate minimal est*

	a	b	c
$q_0 \equiv q_4 \equiv q_8$	q_1	q_1	q_1
$q_1 \equiv q_5$	q_2	q_2	q_2
$‘ q_2 \equiv q_6$	q_3	q_3	q_3
$q_3 \equiv q_7$	q_0	q_0	q_0

Il est instructif d'observer de près ce que l'on vient de faire. On sait que

- *La relation R_M raffine la relation R_L, c'est-à-dire, si deux mots sont équivalents sous R_M, ils le sont sous R_L.*
- *Les états de M peuvent être identifiés aux classes déquivalence de R_M : q correspond à la classe $[u]$ telle que $\hat{\delta}(s, u) = q$.*
- *Les états de l'automate minimal pour L sont les classes déquivalence de R_L.*
- *Si deux classes déquivalence de R_M sont des parties de la même classe déquivalence de R_L, les états correspondants peuvent être identifiés.*
- *Deux états q et p de M sont distincts (ne peuvent pas être identifiés en un seul) s'il exist un mot $u \in \Sigma^*$ tel que $\hat{\delta}(q, u) \in F$ et $\hat{\delta}(p, u) \notin F$ (ou inversement).*

Ce que nous avons fait, c'est de trouver des $u \in \Sigma^*$ qui permettent de différentier des états de M. On peut les trouver dans le tableau en concatenant les lettres indiquées dans les cases.

– *dans les cases avec $0, \varepsilon$, le mot recherché est simplement ε;*
– *dans les cases avec $1, a$, le mot est $a\varepsilon = a$'*
– *dans les cases avec $2, a$, o doit chercher plus loin, mais dans notre cas c'est simple, le mot sera aa.*

Voici comment (exemple). Les états q_5, q_6 sont différents car $\delta(q_5, a) = q_6$, $\delta(q_6, a) = q_7$ et $\delta(q_6, a) = q_7 \notin F$,$\delta(q_7, a) = q_8 \in F$. Donc $\hat{\delta}(q_5, aa) = q_7 \notin F$ et $\hat{\delta}(q_6, aa) = q_8 \in F$.

Exercice 13 *Ecrivez un algorithme pour trouver les mots qui disniguent les états différents d'un AFD en utilisant un tableau comme celui de l'exemple.*