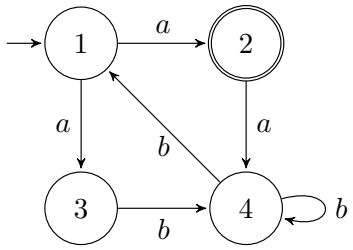
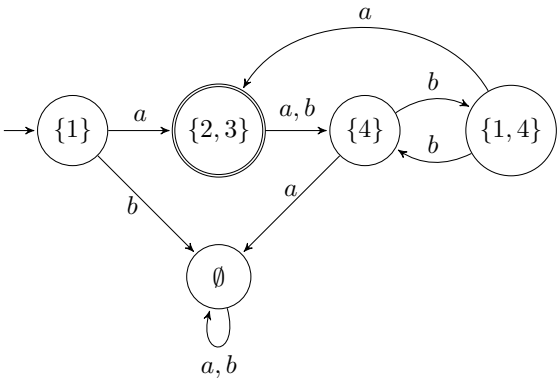
1/2L et racine L

I Algorithme de déterminisation

Déterminiser l'automate suivant en utilisant l'algorithme du cours :



Solution :



II Clôture des langages reconnaissables

Si $m = m_1...m_n$ est un mot, on définit son miroir $\widetilde{m} = m_n...m_1$. Si L est un langage, on définit son miroir $\widetilde{L} = \{\widetilde{m} \mid m \in L\}$.

1. Montrer que le miroir d'un langage reconnaissable est reconnaissable.

   Solution : Soit $A = (\Sigma, Q, I, F, E)$ un langage reconnaissant L. Alors $\widetilde{A} = (\Sigma, Q, F, I, \widetilde{E})$ reconnaît $\widetilde{A}$, où on a inversé toutes les transitions ($\widetilde{E} = \{(q_1, a, q_2) \mid (q_2, a, q_1) \in E\}$). En effet il existe un chemin $q_1 \in I \xrightarrow{m_1} q_2 \xrightarrow{m_2} ... \xrightarrow{m_k} q_{k+1} \in F$ dans A si et seulement si il existe un chemin $q_{k+1} \in F \xrightarrow{m_k} q_k \xrightarrow{m_{k-1}} ... \xrightarrow{m_1} q_1 \in I$ dans $\widetilde{A}$.

Si L est un langage sur Σ, on définit :

- $Pref(L) = \{u \in \Sigma^* \mid \exists v \in \Sigma^*,\ uv \in L\}$: ensemble des préfixes des mots de L.
- $Suff(L) = \{u \in \Sigma^* \mid \exists v \in \Sigma^*,\ vu \in L\}$: ensemble des suffixes des mots de L.
- $Fact(L) = \{u \in \Sigma^* \mid \exists v, w \in \Sigma^*,\ vuw \in L\}$: ensemble des facteurs des mots de L.

2. Montrer que si L est reconnaissable alors $Pref(L)$, $Suff(L)$, $Fact(L)$ le sont aussi.

   Solution : Soit $A = (\Sigma, Q, I, F, E)$ un langage reconnaissant L. Soit Q' l'ensemble des états co-accessibles et Q'' l'ensemble des états accessibles. Un mot m appartient à $Pref(L)$ si et seulement si il existe un chemin étiqueté par m d'un état de I vers un état de Q'. Donc (Σ, Q, I, Q', E) reconnaît $Pref(L)$. De même, (Σ, Q, Q'', F, E) reconnaît $Suff(L)$ et (Σ, Q, Q'', Q', E) reconnaît $Fact(L)$.

   Autre solution : après avoir démontré que $Pref(L)$ est reconnaissable, on peut remarquer que $Suff(L) = \widetilde{Pref(\widetilde{L})}$ ($m \in \widetilde{Pref(\widetilde{L})} \iff \widetilde{m} \in Pref(\widetilde{L}) \iff \exists v \in \Sigma^*,\ \widetilde{m}v \in \widetilde{L} \iff \exists v \in \Sigma^*,\ \widetilde{v}m \in L \iff m \in Suff(L)$, où on a utilisé le fait que $\widetilde{xy} = \widetilde{y}\,\widetilde{x}$).
   On peut aussi en déduire que $Fact(L)$ est reconnaissable en remarquant que $Fact(L) = Suff(Pref(L))$ $(= Pref(Suff(L)))$.
   En effet $m \in Suff(Pref(L)) \iff \exists u \in \Sigma^*,\ um \in Pref(L) \iff \exists u \in \Sigma^*,\ \exists v \in \Sigma^*,\ umv \in L \iff m \in Fact(L)$.

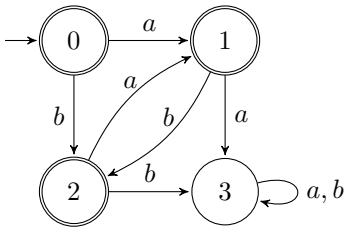
3. Montrer que si L est rationnel alors $Pref(L)$, $Suff(L)$, $Fact(L)$ le sont aussi (puisqu'on va montrer que rationnel = reconnaissable, c'est une preuve alternative à la précédente).

> Solution : Soit e est une expression rationnelle dont le langage est L. On définit par induction une expression rationnelle $P(e)$ de langage $Pref(L)$:
>
> - Si $e = a \in \Sigma$: $P(e) = \varepsilon + a$.
> - Si $e = e_1 + e_2$: $P(e) = P(e_1) + P(e_2)$.
> - Si $e = e_1 e_2$: $P(e) = P(e_1) + e_1 P(e_2)$.
> - Si $e = e_1^*$: $P(e) = e_1^* P(e_1)$.
>
> De même pour $Suff(L)$ et $Fact(L)$.

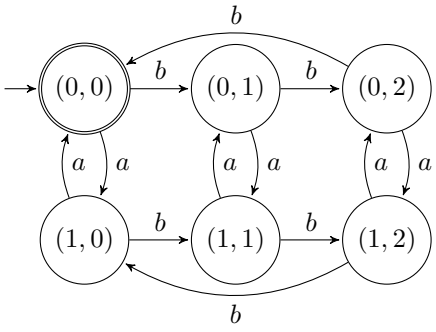
III Reconnaissable ou non ?

Pour chacun de ces langages, dire s'il est reconnaissable ou non. Justifier.

1. L_1 = mots sur $\{a, b\}$ sans lettres consécutives égales.

> Solution : Idée : on se retrouve dans l'état 1 (respectivement 2) si la dernière lettre lue est un a (resp. b).
>
> 
>

2. L_2 = mots sur $\{a, b\}$ ayant un nombre pair de a et dont le nombre de b est multiple de 3.

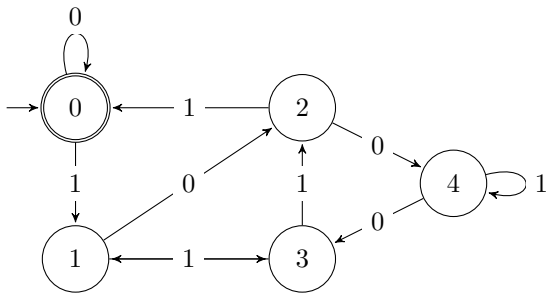
> Solution : On peut construire un automate reconnaissant les mots ayant un nombre pair de a en utilisant 2 états (suivant la parité du nombre de a lus jusqu'à présent). De façon similaire, les mots ayant un nombre de b multiple de 3 peuvent être reconnus par un automate avec 3 états, pour chaque reste modulo 3 du nombre de b déjà lus.
> Pour vérifier les deux en même temps, on peut utiliser l'automate « produit » vu en cours, pour reconnaître l'intersection des deux langages précédents. L'état nommé (i, j) correspond à la lecture d'un mot dont le nombre de a est i modulo 2 et le nombre de b est j modulo 3 :
>
> 
>

3. $L_3 = \{m \in \{a, b\}^* \mid |m|_a = |m|_b\}$ (où $|m|_a$ est le nombre de a du mot m).

> Solution : Supposons que L_3 soit reconnaissable. Soit n l'entier donné par le lemme de l'étoile et $u = a^n b^n$.
> $u \in L$ et $|u| \geq n$ donc, par le lemme de l'étoile, il existe des mots x, y, z tels que $|xy| \leq n$, $y \neq \varepsilon$ et $\forall k \in \mathbb{N}$, $xy^k z \in L$.
> Comme $|xy| \leq n$, y ne contient que des a. En prenant $k = 0$, on trouve que $xy^0 z = xz \in L$. Mais xz contient strictement plus de b que de a, ce qui est absurde.

4. L_4 = écritures en base 2 des multiples de 5.

Solution : On va construire un automate dont les états sont les restes possibles modulo 5. Si l'automate est dans un état q a déjà lu $n_1...n_p$ (c'est à dire $\overline{n_1...n_p}^2 \equiv q[5]$) et qu'il lit n_{p+1} alors il doit aller dans l'état $2q + n_{p+1}$ car $\overline{n_1...n_p n_{p+1}}^2 = 2 \times \overline{n_1...n_p}^2 + n_{p+1} \equiv 2q + n_{p+1}[5]$.
Ce langage est donc reconnu par l'automate $(\Sigma, Q, I, F, \delta) = (\{0,1\}, \{0,1,2,3,4\}, \{0\}, \{0\}, \delta)$ où $\delta(q,a) = 2q + a$ [5] :



5. $L_5 = \{a^p \mid p \text{ est un nombre premier}\}$.

Solution : Supposons que L_5 soit reconnaissable. Soit n l'entier donné par le lemme de l'étoile.
Soit $p \geq n+2$ un nombre premier et $u = a^p$.
$u \in L$ et $|u| \geq n$ donc, par le lemme de l'étoile, il existe i_1, i_2, i_3 tels que $i_1 + i_2 \leq n$, $i_2 > 0$ et $\forall k \in \mathbb{N}$, $a^{i_1}(a^{i_2})^k a^{i_3} = a^{i_1+ki_2+i_3} \in L$ (on a utilisé le fait que u ne contient que des a).
Soit $k = i_1 + i_3$ et $q = i_1 + ki_2 + i_3$. Alors, d'après le lemme de l'étoile, $a^q \in L$. Mais $q = (i_1 + i_3)(1 + i_2)$ avec $1 + i_2 > 1$ et $i_1 + i_3 \geq i_3 > 1$ (car $p \geq n+2$) donc q n'est pas premier, ce qui est absurde.

IV Algorithmes sur les automates

1. À quelle condition nécessaire et suffisante simple le langage reconnu par un automate est vide? Décrire un algorithme pour le savoir.

Solution : Il est vide si et seulement si il n'existe pas de chemin d'un état initial vers un état final. On peut le décider en effectuant un parcours du graphe de l'automate.

2. À quelle condition nécessaire et suffisante simple le langage reconnu par un automate est fini? Décrire un algorithme pour le savoir.

Solution : Il est fini si et seulement si son graphe n'a pas de cycle. Nous avons vu un algorithme pour cela dans le cours sur les graphes.

3. Décrire un algorithme pour déterminer si deux automates admettent le même langage.

Solution : Soient $A_1 = (\Sigma, Q_1, i_1, F_1, \delta_1)$ et $A_2 = (\Sigma, Q_2, i_2, F_2, \delta_2)$. On peut construire des automates A et A' reconnaissant $L(A_1)\backslash L(A_2)$ et $L(A_2)\backslash L(A_1)$, en utilisant « l'automate produit » décrit dans le cours. Il suffit alors de déterminer si $L(A) = \emptyset$ et $L(A') = \emptyset$, en utilisant la question 3.

4. Soit A un automate à n états. Montrer que si $L(A)$ est non vide alors il contient un mot de longueur $\leq n-1$.

Solution : Soit $m = m_1 m_2 ... m_k \in L(A)$ un mot de longueur minimum accepté par A. Soit $q_1 \xrightarrow{m_1} q_2 \xrightarrow{m_2} ... \xrightarrow{m_k} q_{k+1}$ un chemin acceptant de A dont l'étiquette est m. Supposons $k \geq n$. q_1, ..., q_{k+1} ne peuvent pas tous être différents (car il n'y a que n états), disons $q_i = q_j$ avec $i < j$. Alors $q_1 \xrightarrow{m_1} q_2 \xrightarrow{m_2} ... \xrightarrow{m_i} q_i = q_j \xrightarrow{m_{j+1}} ... \xrightarrow{m_k} q_{k+1}$ est un chemin d'un état initial à un état final donc $m_1...m_i m_{j+1}...m_k$ est un mot accepté de longueur strictement inférieure à k, ce qui est absurde.

5. On dit qu'un automate est **émondé** si, pour tout état q, il existe d'une part un chemin d'un état initial à q et d'autre part un chemin de q à un état final. Montrer que tout automate est équivalent à un automate émondé.

Solution : Il suffit d'enlever de l'automate tous les états qui ne conviennent pas (ainsi que toutes les transitions vers/depuis ces états). Comme un chemin acceptant ne peut clairement pas passer par un de ces états, ceci ne change pas les mots acceptés par l'automate.

V Oral ENS info

On fixe un alphabet Σ avec $|\Sigma| > 1$. Un mot $w \in \Sigma^*$ est un palindrome s'il s'écrit $w = a_1 \cdots a_n$ et qu'on a $a_i = a_{n-i+1}$ pour tout $1 \leq i \leq n$. On note $\Pi \subseteq \Sigma^*$ le langage des palindromes. Pour un automate fini A sur Σ, on note $L(A)$ le langage reconnu par A.

1. Soit $\Pi_n := \Pi \cap \Sigma^n$. Montrer que pour tout automate fini déterministe complet A, pour tout $n \in \mathbb{N}$, si $L(A) \cap \Sigma^{2n} = \Pi_{2n}$, alors A a au moins $|\Sigma|^n$ états.
2. En déduire que le langage Π n'est pas régulier.
3. Étant donné un automate fini A sur Σ, peut-on calculer un automate A_Π qui reconnaisse $L(A) \cap \Pi$?
4. Pour tout mot $u = b_1 \cdots b_m$ de Σ^*, on note $\overline{u} := b_m \cdots b_1$ son miroir. Étant donné A, peut-on calculer un automate A'_Π qui reconnaisse $\{u \in \Sigma^* \mid u\overline{u} \in L(A)\}$?
5. On appelle Π_{pair} l'ensemble des palindromes de longueur paire, i.e., $\Pi_{\text{pair}} = \bigcup_{n \in \mathbb{N}} \Pi_{2n}$. Proposer un algorithme qui, étant donné un automate fini A sur Σ, détermine si $L(A) \cap \Pi_{\text{pair}}$ est vide, fini, ou infini. Discuter de sa complexité en temps et en espace.
6. Modifier l'algorithme de la question 4 pour calculer la cardinalité de $L(A) \cap \Pi_{\text{pair}}$ quand cet ensemble est fini, en faisant l'hypothèse que l'automate d'entrée A est déterministe. Comment la complexité est-elle affectée?
7. Modifier l'algorithme des questions 4 et 5 pour qu'il s'applique à $L(A) \cap \Pi$.

Solution :

1. Soit A un tel automate fini déterministe d'ensemble d'états Q, et fixons $n \in \mathbb{N}$. Considérons la fonction $f : \Sigma^n \to Q$ qui associe à $w \in \Sigma^n$ l'état q (unique) auquel A aboutit en lisant w. Montrons que f est injective. Procédons par l'absurde et supposons que $f(u) = f(v)$ pour $u \neq v$ de Σ^n. On sait que $u\overline{u}$ est un palindrome de longueur $2n$ donc il y a un chemin étiqueté par $\overline{u}$ de $f(u)$ à un état final de A. En combinant ce chemin avec le chemin de l'état initial à q étiqueté par v, on conclut que l'automate accepte $v\overline{u}$. Comme $u \neq v$, c'est pourtant un mot de longueur $2n$ qui n'est pas un palindrome, contradiction. Ainsi, f est injective, donc $|Q| \geq |\Sigma^n| \geq |\Sigma|^n$.
2. Procédons par l'absurde et supposons que Π soit régulier. Soit A un automate fini déterministe complet qui reconnaisse Π, et soit n son nombre d'états. Comme $L(A) = \Pi$ par hypothèse, on sait que $L(A) \cap \Sigma^{2n} = \Pi_{2n}$, ainsi d'après la question précédente A a au moins $|\Sigma|^n \geq 2^n$ états, mais il en a n et on a $n < 2^n$, donc contradiction. Ainsi Π n'est pas régulier.
3. C'est manifestement déraisonnable : pour A un automate reconnaissant le langage régulier Σ^*, la question nous demanderait de calculer un automate qui reconnaisse $\Sigma^* \cap \Pi = \Pi$, et on sait d'après la question précédente qu'il n'en existe pas.
4. De façon un peu contre-intuitive, c'est possible. Posons $A = (Q, I, F, \delta)$ où Q est l'ensemble d'états de A, où I est l'ensemble d'états initiaux, où F est l'ensemble d'états finaux, et où $\delta \subseteq Q \times \Sigma \times Q$ est la relation de transition. On construit d'abord en temps linéaire l'automate $\overline{A} := (Q, F, I, \overline{\delta})$ où $\overline{\delta} := \{(q', a, q) \mid (q, a, q') \in \delta\}$. Il est clair que $\overline{A}$ reconnaît $\overline{L(A)} := \{\overline{u} \mid u \in L(A)\}$, puisqu'il y a une correspondance bijective entre les chemins acceptants dans A et dans $\overline{A}$, et l'effet de cette bijection sur l'étiquette des chemins est l'opération miroir.

   On construit ensuite (en temps quadratique en A) l'automate produit $A \times \overline{A}$ défini comme suit : $A \times \overline{A} := \left(Q \times Q, I \times F, F', \delta \times \overline{\delta}\right)$ où la relation de transition est définie comme $\overline{\delta} := \left\{\left((q, \overline{q'}), a, (q', \overline{q}) \mid \ (q, a, q') \in \delta \wedge \left(\overline{q'}, a, \overline{q}\right) \in \overline{\delta}\right.\right.$, et où les états finaux sont $\{(q, q) \mid q \in Q\}$. Il est clair que l'automate produit peut atteindre l'état $(q, \overline{q})$ en lisant un mot $w \in \Sigma^*$ si et seulement si l'automate A peut atteindre l'état q en lisant w (dans la première composante) et l'automate $\overline{A}$ peut atteindre l'état $\overline{q}$ en lisant w (dans la seconde composante), ce qui est le cas, par définition de $\overline{A}$, si et seulement s'il y a un chemin dans A étiqueté par $\overline{u}$ de $\overline{q}$ à un état final de F. En particulier, l'automate produit peut atteindre l'état (q, q) en lisant un mot $w \in \Sigma^*$ si et seulement si l'automate A peut atteindre q en lisant w et l'automate A a un chemin étiqueté par $\overline{u}$ de q à un état final. Ainsi, la définition de F' assure que l'automate produit accepte un mot $w \in \Sigma^*$ si et seulement s'il existe un état q tel que A a un chemin acceptant pour $w\overline{w}$ qui passe par un certain état q après la lecture de w, c'est-à-dire si et seulement si A accepte $w\overline{w}$. Ceci établit que la construction est correcte.
5. La construction de l'automate produit A'_Π de la question précédente s'effectue en temps quadratique en A. Il est clair que $L(A) \cap \Pi_{\text{pair}}$ a la même cardinalité que $L\left(A'_\Pi\right)$, puisque la fonction $f : \Sigma^* \to \Pi_{\text{pair}}$ définie par $f(u) := u\overline{u}$ pour tout $u \in \Sigma^*$ définit une bijection entre $L\left(A'_\Pi\right)$ et $L(A) \cap \Pi_{\text{pair}}$. Ainsi, il suffit de déterminer si $L\left(A'_\Pi\right)$ est vide, fini, ou infini.

On détermine d'abord si $L(A'_\Pi)$ est non-vide en vérifiant qu'il existe un chemin d'un état initial de A'_Π à un état final de A'_Π, en temps linéaire, par exemple avec un DFS.

On détermine ensuite si A'_Π contient un cycle d'états accessibles et co-accessibles. Pour ce faire, on détermine les états accessibles et co-accessibles par un parcours de graphe, et ensuite on utilise un DFS pour déterminer si on rencontre un cycle dans ces sommets. Ceci est toujours en temps linéaire, et conclut.

6. Dans le cas où l'automate produit est déterministe, on peut compter le nombre de mots qu'il accepte par de la programmation dynamique. Spécifiquement, le nombre de mots acceptés à partir d'un état q est 1 ou 0 selon que q est final ou non, plus le nombre de mots acceptés à partir des états vers lesquels q a des transitions sortantes. Noter que, comme l'automate est déterministe, les étiquettes de ces transitions sont différentes, ainsi les ensembles de mots concernés sont disjoints, et c'est effectivement correct de faire la somme comme expliqué.

   On n'a en fait pas vraiment besoin que l'automate produit soit déterministe : il suffit qu'il soit inambigu, c'est-à-dire que tout mot accepté a un unique chemin acceptant. Dans ce cas, la construction présentée reste correcte, parce que si un état q a des transitions étiquetées par la même lettre vers deux états distincts q_1 et q_2, alors la définition de l'inambiguïté impose que l'ensemble des mots acceptés à partir de q_1 et celui des mots acceptés à partir de q_2 sont disjoints.

   Il suffit alors d'observer que, si A est déterministe (ou, en fait, s'il est inambigu), alors $\overline{A}$ est toujours inambigu (même si pas forcément déterministe). Maintenant, le produit $A \times \overline{A}$ avec l'ensemble d'états finaux indiqué est également inambigu : si un mot $u \in \Sigma^*$ avait un chemin vers (q, q) et vers (q', q') dans l'automate produit avec $q \neq q'$, alors on saurait que $u\overline{u}$ a deux chemins acceptants distincts dans A (un où l'état intermédiaire est q, un autre où c'est q'), ce qui contredirait le fait que A soit inambigu. Ainsi, si A est inambigu on peut construire le produit avec la même complexité qu'avant, il est inambigu, et on compte la taille du langage qu'il accepte comme expliqué. Comme cet ensemble est en bijection avec $L(A) \cap \Pi_{\text{pair}}$ (comme prouvé à la question 4), on a établi le résultat.

7. L'automate produit A' de la question 4 ne gère intuitivement que les u qui se complètent en un palindrome de longueur paire. Ceci dit, pour chaque $a \in \Sigma$, on peut adapter l'ensemble d'états finaux de l'automate produit pour construire un automate A'_a qui reconnaisse exactement $\{u \in \Sigma^* \mid ua\overline{u} \in L(A)\}$, c'est-à-dire le langage des mots u qui se complètent en un palindrome de longueur impaire accepté par A en ajoutant a comme lettre centrale, puis le miroir de u. On définit A'_a pour chaque $a \in \Sigma$ exactement comme A' mais avec l'ensemble d'états finaux $F'_a := \{(q, q') \mid (q, a, q') \in \delta\}$, ce qui est clairement correct : un mot $u \in \Sigma^*$ a un chemin acceptant dans A'_a finissant en (q, q') si et seulement s'il y a un chemin d'un état initial de A à q étiqueté par u, une transition dans A étiquetée par a de q à q', et un chemin de q' à un état final de A étiqueté par $\overline{u}$ dans A. Du reste, si A est inambigu alors tous ces automates produits le sont, par le même raisonnement qu'à la question précédente. On peut ainsi appliquer la construction de la question précédente à A' et à A'_a pour chaque $a \in \Sigma$, ce qui ne fait que rajouter un facteur $|\Sigma|$ à la complexité : on peut obtenir la cardinalité du nombre de palindromes reconnus en sommant les quantités pour A' et pour chaque A'_a (en traitant ∞ de la manière attendue). Ceci est correct parce que les palindromes de $\Pi \cap L(A)$ se partitionnent entre ceux de longueur paire et ceux de longueur impaire dont la lettre centrale est $a \in \Sigma$ pour chaque a. (En revanche, noter que le langage de A' et celui des A'_a ne sont pas forcément disjoints, vu qu'il est tout à fait possible, par exemple, que A accepte $u\overline{u}$ et $ua\overline{u}$ pour diverses valeurs de $a \in \Sigma$. Autrement dit, il faut bien sommer la cardinalité de ces langages même si leur union n'est pas disjointe, pour la raison qu'on vient d'expliquer.)