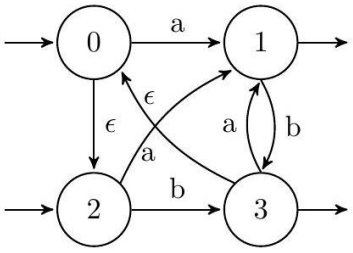
I Exercice CCP



1. Déterminiser cet automate.
2. Construire une expression régulière dénotant le langage reconnu par cet automate.
3. Décrire simplement avec des mots le langage reconnu par cet automate.

II Langage local et linéaire (Mines 2007)

Un **alphabet** Σ est un ensemble fini d'éléments appelés **lettres**. Un **mot** sur Σ est une suite finie de lettres de Σ. La **longueur** d'un mot est le nombre de lettres le composant. Le mot de longueur nulle est noté ε. Une lettre de Σ est aussi un mot de longueur 1.
On désigne par Σ^* l'ensemble des mots sur Σ, y compris le mot ε. On note Σ^p l'ensemble des mots de longueur p sur Σ.

Un **langage** est une partie de Σ^*. La **concaténation** de deux mots u et v est notée $u.v$ et la concaténation de deux langages L_1 et L_2 est notée $L_1.L_2$ $(= \{u.v \mid u \in L_1, v \in L_2\})$.

On appelle **expression rationnelle** sur l'alphabet Σ toute formule obtenue récursivement à partir des règles suivantes:

- $\emptyset$ et ε sont des expressions rationnelles
- pour tout a dans Σ, a est une expression rationnelle
- si e et f sont des expressions rationnelles, $(e+f)$, $(e.f)$ et $(e)^*$ sont des expressions rationnelles

La **longueur** $|e|$ d'une expression rationnelle e est son nombre total de symboles (lettres de l'alphabet Σ ou symboles *, +, ., $\emptyset$, ε).

On fait correspondre à chaque expression rationnelle sur l'alphabet Σ un langage sur l'alphabet Σ; ce langage sera dit **décrit par l'expression rationnelle**. On rappelle que :

- l'expression $\emptyset$ décrit le langage vide
- l'expression ε décrit le langage qui ne contient que le mot ε
- si a est dans Σ, l'expression a décrit le langage réduit au mot a
- si e_1 décrit un langage L_1 et si e_2 décrit un langage L_2 , (e_1+e_2) décrit le langage $L_1 \cup L_2$ et $(e_1.e_2)$ décrit le langage $L_1.L_2$
- si e décrit un langage L, $(e)^*$ décrit l'itération, ou étoile, de L, notée L^*, c'est-à-dire l'union de toutes les puissances de L: $L^* = \bigcup_{p\in\mathbb{N}} L^p$, avec $L^0 = \{\varepsilon\}$, $L^1 = L$ et, pour $p \geq 2$, $L^p = L.L^{p-1}$
  (L^p contient donc toutes les concaténations de p mots de L et L^* les concaténations d'un nombre quelconque de mots de L)

Deux expressions rationnelles sont dites **équivalentes** si elles décrivent le même langage.

Soit P la propriété suivante: toute expression rationnelle e est équivalente

- soit à l'expression rationnelle $\emptyset$,
- soit à l'expression rationnelle ε,
- soit à une expression rationnelle e qui ne contient pas les symboles $\emptyset$ et ε,

- soit à une expression rationnelle de la forme $(e' + \varepsilon)$, où e' est une expression rationnelle qui ne contient pas les symboles $\emptyset$ et ε.

1. Soient e_1 et e_2 deux expressions rationnelles ne contenant pas les symboles $\emptyset$ et ε. En exhibant sans démonstration des expressions rationnelles équivalentes, montrer que la propriété P est vraie pour les huit expressions suivantes: $(\varepsilon + \varepsilon)$, $(e_1 + \emptyset)$, $(\varepsilon.\emptyset)$, $(e_1.\emptyset)$, $(e_1.\varepsilon)$, $((e_1 + \varepsilon) + (e_2 + \varepsilon))$, $(e_1.(e_2 + \varepsilon))$, $((e_1 + \varepsilon).(e_2 + \varepsilon))$.
2. Esquisser une démonstration par récurrence de la propriété P dans le cas général. En particulier, on fera apparaître les différents cas.

D'après la propriété P, on peut se passer de $\emptyset$ dans l'écriture d'une expression rationnelle en ne perdant que la possibilité de décrire le langage vide. On définit donc le type d'expression rationnelle suivant en OCaml (où `'a` est le type des lettres et `L(a)` désigne une lettre `a`):

3. Écrire une fonction `has_eps : 'a regexp -> bool` déterminant si le langage décrit par une expression rationnelle contient le mot vide ε. Quelle est sa complexité, en fonction de la longueur de l'expression rationnelle?

II.1 Ensembles $P(L)$, $S(L)$, $F(L)$

Soit L un langage sur un alphabet Σ. On définit:

- $P(L) = \{a \in \Sigma \mid a\Sigma^* \cap L \neq \emptyset\}$ (ensemble des premières lettres des mots de L)
- $S(L) = \{a \in \Sigma \mid \Sigma^* a \cap L \neq \emptyset\}$ (ensemble des dernières lettres des mots de L)
- $F(L) = \{u \in \Sigma^2 \mid \Sigma^* u \Sigma^* \cap L \neq \emptyset\}$ (ensemble des facteurs de longueur 2 des mots de L)

Ces ensembles seront représentés par des listes en OCaml.

1. Écrire une fonction `union : 'a list -> 'a list -> 'a list` telle que, si `l1` et `l2` sont des listes sans doublon (ce qu'on suppose être le cas...), `union l1 l2` renvoie une liste sans doublon contenant les éléments des deux listes. Par exemple, `union [1; 2] [3; 1]` peut renvoyer `[1; 2; 3]` (l'ordre des éléments de la liste de retour n'importe pas).
   Donner la complexité de `union`.
2. Écrire une fonction `prefixe` de type `'a regexp -> 'a list` telle que `prefixe e` renvoie $P(L)$, où L est le langage décrit par `e`.
3. Comment modifier `prefixe` pour obtenir une fonction `suffixe` renvoyant $S(L)$?
4. Écrire une fonction `produit : 'a list -> 'b list -> ('a * 'b) list` effectuant le « produit cartésien » de 2 listes: si `l1` et `l2` sont des listes, `produit l1 l2` renvoie une liste de tous les couples composés d'un élément de `l1` et un élément de `l2`. Par exemple, `produit [1; 2] [3; 1]` peut renvoyer
   `[(1, 3); (1, 1); (2, 3); (2; 1)]` (l'ordre des éléments de la liste de retour n'importe pas).
5. En déduire une fonction `facteur` de type `'a regexp -> ('a * 'a) list` telle que `facteur e` renvoie $F(L)$, où L est le langage décrit par `e`. Chaque mot de longueur deux est représenté par un couple.

II.2 Langages locaux

On dit qu'un langage L sur un alphabet Σ est un langage **local** si un mot u de Σ^* autre que le mot ε appartient à L si et seulement si (en utilisant les définitions de la partie précédente):

- la lettre initiale de u est dans $P(L)$,
- la lettre finale de u est dans $S(L)$,
- et tout facteur de longueur 2 de u est dans $F(L)$.

Par exemple, $L = \{ab, bab\}$ n'est pas local car $P(L) = \{a, b\}$, $S(L) = \{b\}$ et $F(L) = \{ab, ba\}$ mais $u = abab \notin L$.
Noter qu'un langage local peut contenir ou non le mot ε. Un langage local L est donc caractérisé de façon unique par l'ensemble $P(L)$ des lettres initiales permises, l'ensemble $S(L)$ des lettres finales permises, l'ensemble $F(L)$ des facteurs de longueur 2 permis et le fait de contenir ou non le mot ε.

1. On considère l'alphabet $\Sigma = \{a, b\}$. Montrer que les langages sur Σ décrits par les expressions rationnelles aa^* et $(ab)^*$ sont locaux.
2. Donner une expression rationnelle décrivant le langage local L sur $\Sigma = \{a, b, c\}$ tel que $\varepsilon \in L$, $P(L) = \{a, c\}$, $S(L) = \{c\}$ et $F(L) = \{ab, ba, bc\}$.

3. L'intersection de deux langages locaux est-il forcément local?
4. Montrer que l'union de deux langages locaux n'est pas forcément local.
5. Soit L_1 un langage local sur l'alphabet Σ_1 et L_2 un langage local sur l'alphabet Σ_2. On suppose $\Sigma_1 \cap \Sigma_2 = \emptyset$. Montrer que $L_1 \cup L_2$ est local.
   On donnera $P(L_1 \cup L_2)$, $S(L_1 \cup L_2)$, $F(L_1 \cup L_2)$ en fonction de $P(L_1)$, $S(L_1)$, $F(L_1)$, $P(L_2)$, $S(L_2)$, $F(L_2)$.
6. Soit L un langage local. Montrer que L^* est local.
   On donnera $P(L^*)$, $S(L^*)$, $F(L^*)$ en fonction de $P(L)$, $S(L)$, $F(L)$.
7. Montrer que la concaténation de deux langages locaux n'est pas forcément local.
8. Soit L_1 un langage local sur l'alphabet Σ_1 et L_2 un langage local sur l'alphabet Σ_2. On suppose $\Sigma_1 \cap \Sigma_2 = \emptyset$. Montrer que $L_1.L_2$ est local.
   On donnera $P(L_1.L_2)$, $S(L_1.L_2)$, $F(L_1.L_2)$ en fonction de $P(L_1)$, $S(L_1)$, $F(L_1)$, $P(L_2)$, $S(L_2)$, $F(L_2)$.
9. Écrire une fonction `appartient : 'a list -> 'a list -> 'a list -> ('a * 'a) list -> bool -> bool` telle que, si `u` est une liste représentant un mot et `p`, `s`, `f` des listes représentants $P(L)$, $S(L)$, $F(L)$ pour un certain langage local L, et si `eps` indique si ε appartient à L, `appartient u p s f eps` détermine si `u` appartient à L.
10. On définit $N(L) = \Sigma^2 \backslash F(L)$.
    Pour un langage L quelconque, quelle relation existe t-il entre $L \backslash \{\varepsilon\}$ et $(P(L)\Sigma^* \cap \Sigma^* S(L)) \setminus (\Sigma^* N(L) \Sigma^*)$?
    Et pour un langage local L?
11. On admet dans cette question que le complémentaire d'un langage rationnel est rationnel (c'est à dire: L rationnel $\implies$ $\Sigma^* \backslash L$ rationnel).
    En déduire que l'intersection et la différence de deux langages rationnels est rationnel, puis que tout langage local est rationnel.
12. Réciproquement, tout langage rationnel est-il local?

II.3 Langages linéaires

Une expression rationnelle e sur un alphabet Σ est dite **linéaire** si chaque lettre de Σ apparaît au plus une fois dans e.

1. Montrer par récurrence (ou induction structurelle) que toute expression rationnelle linéaire décrit un langage local.
2. Donner un langage local qui ne peut pas être décrit par une expression rationnelle linéaire.
3. Écrire une fonction `lineaire : 'a regexp -> bool` déterminant si une expression rationnelle e est linéaire, de complexité au plus quadratique en la taille de e. Peut-on améliorer cette complexité?

III Langage local, linéaire et automate de Glushkov

1. Construire l'automate de Glushkov reconnaissant $(ab + b)^* b$.
2. Montrer que si e est une expression linéaire alors $L(e)$ est un langage local.
3. Montrer qu'il existe un nombre fini de langages locaux sur un alphabet fixé.
4. Soient $\mathcal{L}_{\text{rat}}$, $\mathcal{L}_{\text{loc}}$, $\mathcal{L}_{\text{lin}}$ l'ensemble des langages rationnels, locaux et linéaires (c'est à dire décrits par une expression rationnelle linéaire). Quelles sont les inclusions entre ces 3 ensembles? Ces inclusions sont-elles strictes?

IV Stabilité des langages rationnels

Pour chacune des propositions suivantes, expliquer pourquoi elle est vraie ou donner un contre-exemple :

1. Si L est rationnel et $L \subseteq L'$ alors L' est rationnel.
2. Si L' est rationnel et $L \subseteq L'$ alors L est rationnel.
3. Si L est rationnel alors L^* est rationnel.
4. Si L^* est rationnel alors L est rationnel.
5. Si L est rationnel sur un alphabet Σ alors $\Sigma^* \setminus L$ (complémentaire de L) est rationnel.

6. Une union finie de langages rationnels est rationnel (L_1, ..., L_n rationnels $\implies \bigcup_{k=1}^{n} L_k$ rationnel).

7. Une union dénombrable de langages rationnels est rationnel (L_1, ..., L_n, ... rationnels $\implies \bigcup_{k=1}^{\infty} L_k$ rationnel).

8. Une intersection finie de langages rationnels est rationnel (L_1, ..., L_n rationnels $\implies \bigcap_{k=1}^{n} L_k$ rationnel).

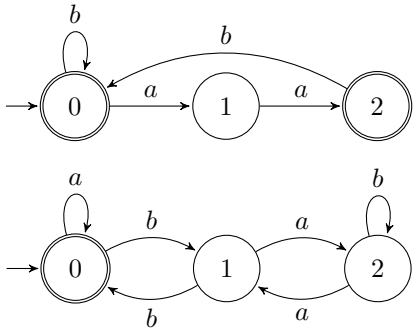
9. Une intersection dénombrable de langages rationnels est rationnel (L_1, ..., L_n, ... rationnels $\implies \bigcap_{k=1}^{\infty} L_k$ rationnel).

V Reconnaissable $\implies$ rationnel avec le lemme d'Arden

1. (Lemme d'Arden) Soient A et B deux langages tels que $\varepsilon \notin A$. Montrer que l'équation $L = AL \cup B$ (d'inconnue le langage L) admet pour unique solution A^*B.

Soit $(\Sigma, Q, q_0, F, \delta)$ un automate déterministe. Si $q_i \in Q$ est un état, on pose $L_i = \{m \in \Sigma^* \mid \delta^*(q_i, m) \in F\}$ (étiquettes des chemins de q_i vers un état final). En écrivant des relations entre les L_i puis en résolvant de proche en proche avec le lemme d'Arden, on obtient une expression rationnelle pour le langage L_0 de l'automate.

2. Appliquer cette méthode pour trouver une expression rationnelle pour chacun des automates ci-dessous. En remplaçant a par 0 et b par 1, que reconnaît le 2ème automate?



VI Reconnaissable $\implies$ rationnel par prog. dyn. ($\approx$ Floyd-Warshall)

Soit $(\Sigma, Q, 0, F, \delta)$ un automate déterministe dont les états sont des entiers de 0 à n (et 0 est l'état initial).
Soit $L(i, j, k)$ le langage des étiquettes des chemins de i à j n'utilisant que des états intermédiaires strictement inférieurs à k.

1. Montrer que $L(i, j, 0)$ est rationnel, pour tous les états i, j.
2. Prouver une équation de récurrence sur $L(i, j, k)$.
3. En déduire, par récurrence, que tout langage reconnaissable est rationnel.

VII Racine d'un langage

Soit L un langage reconnaissable sur Σ. Montrer que $\sqrt{L} = \{m \in \Sigma^* \mid m^2 \in L\}$ est un langage reconnaissable.