

TD corrigé équivalence rationnels et reconnaissables

I Petites questions

1. Montrer qu'il existe un nombre fini de langages locaux sur un alphabet fixé.
► Tout langage local est de la forme $(P\Sigma^* \cap \Sigma^*S) \setminus \Sigma^*N\Sigma^*$ où $P \subseteq \Sigma$, $S \subseteq \Sigma$, $N \subseteq \Sigma \times \Sigma$, en rajoutant éventuellement ε . Si $n = |\Sigma|$, il y a au plus 2^n choix pour P , 2^n pour S et 2^{2n} pour N (car $|\Sigma \times \Sigma| = |\Sigma||\Sigma| = 2^{2n}$). D'où au plus $2^n \times 2^n \times 2^{2n}$ valeurs différentes pour l'ensemble $(P\Sigma^* \cap \Sigma^*S) \setminus \Sigma^*N\Sigma^*$.

II Calcul des ensembles $P(L)$, $S(L)$, $F(L)$

Soit L un langage. On définit:

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

Dans la suite, on utilise le type suivant d'expression rationnelle:

```
type 'a regexp =  
  Epsilon  
  | L of 'a  
  | Somme of 'a regexp * 'a regexp  
  | Concat of 'a regexp * 'a regexp  
  | Etoile of 'a regexp;;
```

1. Écrire une fonction `has_eps : 'a regexp -> bool` déterminant si le langage d'une expression rationnelle contient le mot vide ε .

►

```
let rec has_eps e = match e with  
| Epsilon -> true  
| L(a) -> false  
| Somme(e1, e2) -> has_eps e1 || has_eps e2  
| Concat(e1, e2) -> has_eps e1 && has_eps e2  
| Etoile(e1) -> true;;
```

2. 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).

►

```
let rec union l1 l2 = match l1 with  
| [] -> l2  
| e::q -> if mem e l2 then union q l2 else e::union q l2;;
```

3. Écrire une fonction `prefixe` de type `'a regexp -> 'a list` telle que `prefixe e` renvoie $P(L(e))$.

►

```
let rec prefixe e = match e with  
| Epsilon -> []  
| L(a) -> a  
| Somme(e1, e2) -> union (prefixe e1) (prefixe e2)  
| Concat(e1, e2) -> if has_eps e1 then union (prefixe e1) (prefixe e2)  
  else prefixe e1  
| Etoile(e1) -> prefixe e1;;
```

4. Que faudrait-il modifier à **prefixe** pour obtenir une fonction **suffixe** renvoyant $S(L)$?
 ► Echanger **e1** et **e2** dans **Concat(e1, e2) ->**
5. 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 distincts 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).
 ► 1ère solution:

```
let rec produit l1 l2 = match l1 with
| [] -> []
| e::q -> union (map (fun x -> (e, x)) l2) (produit q l2);;
```

2ème solution:

```
let rec produit l1 l2 = match l1, l2 with
| [], _ -> []
| _, [] -> []
| e1::q1, e2::q2 -> (e1, e2)::(produit [e1] q2)@(produit q1 l2);;
```

6. En déduire une fonction **facteur** de type 'a regexp -> ('a * 'a) list telle que **facteur e** renvoie $F(L(e))$.
 ►

```
let rec facteur = function
| Vide -> []
| L(a) -> []
| Somme(e1, e2) -> union (facteur e1) (facteur e2)
| Concat(e1, e2) -> let l = union (facteur e1) (facteur e2) in
    union l (produit (suffixe e1) (prefixe e2))
| Etoile(e) -> union (facteur e) (produit (suffixe e) (prefixe e));;
```

III Autre preuve de rationnel \implies reconnaissable

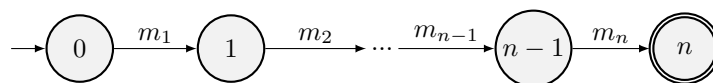
1. Soient L_1 et L_2 deux langages reconnaissables sur le même alphabet Σ . Montrer que $L_1 \cup L_2$, $L_1 L_2$ et L_1^* sont reconnaissables (en donnant un automate reconnaissant chaque langage).
 ► Soit $A_k = (\Sigma, Q_k, i_k, F_k, E_k)$ des automates déterministes reconnaissant L_k ($k = 1, 2$). Quitte à renommer des états, on peut supposer que $Q_1 \cap Q_2 = \emptyset$.
 Alors $A' = (\Sigma, Q_1 \cup Q_2, \{i_1, i_2\}, F_1 \cup F_2, E_1 \cup E_2)$ est un automate (non déterministe car avec deux états initiaux) qui reconnaît $L_1 \cup L_2$. En effet un chemin acceptant de A' est soit un chemin acceptant de A_1 (si le premier état de C est i_1), soit de A_2 (si le premier état de C est i_2).
 Remarque: on aurait aussi pu utiliser l'« automate produit » du cours.
 Pour reconnaître $L_1 L_2$, on peut « connecter » les états finaux de A_1 avec l'état initial de i_2 . Formellement, soit $A'' = (\Sigma, Q_1 \cup Q_2, i_1, F, E)$ où:

- $F = F_2$ si $\varepsilon \notin L_2$, $F_2 \cup F_1$ sinon.
- $E = E_1 \cup E_2 \cup \{(q_1, a, q_2) \mid q_1 \in F_1 \text{ et } (i_2, a, q_2) \in E_2\}$

On montre par double inclusion que $L(A'') = L_1 L_2$.

L_1^* est reconnu par $(\Sigma, Q_1, i_1, F_1 \cup \{\varepsilon\}, E_1 \cup \{(f, a, q') \mid f \in F_1 \text{ et } (i_1, a, q') \in E_1\})$, ce qui revient à « connecter » les états finaux de A_1 avec son état initial i_1 .

2. En déduire que tout langage rationnel est reconnaissable.
 ► Soit \mathcal{L} l'ensemble des langages reconnaissables. Montrons que \mathcal{L} contient l'ensemble des langages rationnels. \mathcal{L} contient les langages réduits à un mot. En effet $\{m_1 \dots m_n\}$ est reconnu par:



Comme on a montré que \mathcal{L} est stable par union (finie), tout langage fini L est aussi dans \mathcal{L} (car $L = \bigcup_{m \in L} \{m\}$).

\mathcal{L} contient donc les langages finis, est stable par concaténation, union, étoile. Or l'ensemble des langages rationnels est le plus petit ensemble ayant ces propriétés, par définition. Donc l'ensemble des langages rationnels est bien inclus dans \mathcal{L} .

IV Reconnaissable \implies rationnel avec le lemme d'Arden

- (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 .

► A^*B est bien une solution: en effet $AA^*B \cup B = (AA^* \cup \{\varepsilon\})B = A^*B$.

Montrons maintenant que A^*B est la seule solution. Soit L une solution de $L = AL \cup B$.

Montrons d'abord par récurrence sur k que $A^k B \subseteq L$.

Comme $B \subseteq AL \cup B = L$, l'initialisation pour $k = 0$ est vérifiée.

Soit $k \in \mathbb{N}$. Supposons que $A^k B \subseteq L$. Alors $A^{k+1} B \subseteq AL$. Donc $A^{k+1} B \subseteq AL \cup B = L$.

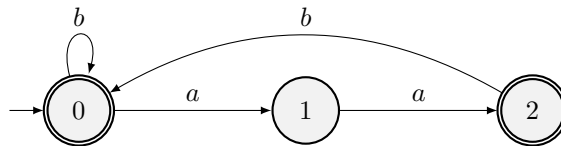
On a donc bien montré que $\forall k \in \mathbb{N}, A^k B \subseteq L$. D'où $A^*B = \bigcup_{k \in \mathbb{N}} A^k B \subseteq L$.

Supposons que $L \not\subseteq A^*B$ et considérons un mot $u \in L \setminus A^*B$ de longueur minimum. Comme $L = AL \cup B$, $u \in AL$ ou $u \in B$. Comme $u \notin A^*B$, $u \notin B$. Donc $u \in AL$: u s'écrit au' avec $a \in A$ et $u' \in L$. Si $u' \in A^*B$ alors $u = au' \in A^*B$, ce qui est faux. Donc $u' \in L \setminus A^*B$, ce qui est une contradiction avec la définition de u . Conclusion: $L \subseteq A^*B$.

Comme $L \subseteq A^*B$ et $L \supseteq A^*B$, on a bien montré que $L = A^*B$.

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.

- 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?



- On utilise la notation des expressions rationnelles pour plus de lisibilité.

$$(0) L_0 = \varepsilon + bL_0 + aL_1$$

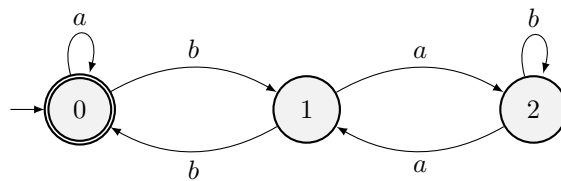
$$(1) L_1 = aL_2$$

$$(2) L_2 = \varepsilon + bL_0$$

En remplaçant L_2 dans (1), on obtient $L_1 = a + abL_0$.

En remplaçant L_1 par cette valeur dans (0), on obtient $L_0 = \varepsilon + bL_0 + aa + aabL_0 = \varepsilon + aa + (b + aab)L_0$.

D'après le lemme d'Arden, $L_0 = (b + aab)^*(\varepsilon + aa)$. On vérifie que c'est bien le langage de l'automate.



- On obtient les équations suivantes:

$$(0) L_0 = \varepsilon + aL_0 + bL_1$$

$$(1) L_1 = aL_2 + bL_0$$

$$(2) L_2 = aL_1 + bL_2$$

En appliquant le lemme d'Arden à (2) (avec $A = \{b\}$ et $B = aL_1$), on trouve $L_2 = b^*aL_1$.

En reportant dans (1), on trouve $L_1 = ab^*aL_1 + bL_0 = (ab^*a)^*bL_0$ en appliquant le lemme d'Arden avec $B = bL_0$ et $A = ab^*a$.

En remplaçant dans (1), on trouve $L_0 = \varepsilon + aL_0 + b(ab^*a)^*bL_0 = \varepsilon + (a + b(ab^*a)^*b)L_0$ d'où $L_0 = (a + b(ab^*a)^*b)^*$.

En remplaçant a par 0 et b par 1, on remarque que l'automate reconnaît les multiples de 3 écrits en base 2 (l'état i correspond à un reste de i modulo 3). On obtient donc l'expression rationnelle $(0 + 1(01^*0)^*1)^*$ pour les multiples de 3 en base 2.

V Reconnaissable \implies rationnel avec un algorithme similaire à 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 .
 ► Soit i et j deux états (éventuellement égaux) et a_1, \dots, a_p toutes les étiquettes des transitions de i vers j . Alors $L(i, j, 0) = \{a_1, \dots, a_p\}$ est fini donc rationnel.
2. Prouver une équation de récurrence sur $L(i, j, k)$.
 ► Soit $u \in L(i, j, k)$ l'étiquette d'un chemin C de i vers j n'utilisant que des états intermédiaires strictement inférieurs à k . Si C n'utilise pas l'état k alors $u \in L(i, j, k-1)$. Sinon C est la concaténation d'un chemin de i à k , d'un certain nombre de cycles de k à k , puis d'un chemin de k à j . On en déduit:

$$L(i, j, k) = L(i, j, k-1) \cup (L(i, k, k-1)L(k, k, k-1)^*L(k, j, k-1))$$

3. En déduire, par récurrence, que tout langage reconnaissable est rationnel.
 ► Montrons par récurrence sur $k \in \{0, \dots, n\}$, $P(k)$: « pour tous états i, j , $L(i, j, k)$ est rationnel ». $P(0)$ est vraie d'après la question 1.
 Supposons $P(k)$ vraie pour un $k > 0$. Soient i, j deux états. Alors $L(i, j, k-1)$, $L(i, k, k-1)$, $L(k, k, k-1)$, $L(k, j, k-1)$ sont rationnels par hypothèse de récurrence donc $L(i, j, k) = L(i, j, k-1) \cup (L(i, k, k-1)L(k, k, k-1)^*L(k, j, k-1))$ est rationnel comme concaténation et étoile de langages rationnels.

VI 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.

► Soit $A = (\Sigma, Q, i, F, \delta)$ un automate déterministe reconnaissant L dont les états sont des entiers $0, \dots, n-1$.

Soit $L_{j,k}$ le langage des étiquettes des chemins de l'état j à l'état k dans A . $L_{j,k}$ est reconnaissable, par $(\Sigma, Q, j, \{k\}, \delta)$.

De plus, on montre que $\sqrt{L} = \bigcup_{q \in Q} \bigcup_{f \in F} L_{i,q} \cap L_{q,f}$:

- Soit $m \in \sqrt{L}$. Soit $q_m = \delta^*(i, m)$ et $f_m = \delta^*(q_m, m)$. Alors $m \in L_{i,q_m}$ et $m \in L_{q_m,f_m}$ et $f_m \in F$ (car $m^2 \in L$).
 Donc $m \in L_{i,q_m} \cap L_{q_m,f_m} \subseteq \bigcup_{q \in Q} \bigcup_{f \in F} L_{i,q} \cap L_{q,f}$.
- Soit $m \in \bigcup_{q \in Q} \bigcup_{f \in F} L_{i,q} \cap L_{q,f}$: il existe alors $q \in Q$ et $f \in F$ tels que $m \in L_{i,q} \cap L_{q,f}$. Alors $\delta^*(i, m^2) = \delta^*(q, m) = f \in F$. D'où $m^2 \in L$, donc $m \in \sqrt{L}$.

Ainsi $\sqrt{L} = \bigcup_{q \in Q} \bigcup_{f \in F} L_{i,q} \cap L_{q,f}$ est rationnel comme union et intersection de langages rationnels (une union de langages

rationnels est rationnel par définition, et une intersection de langages rationnels est rationnel en utilisant par exemple l'« automate produit » du cours).