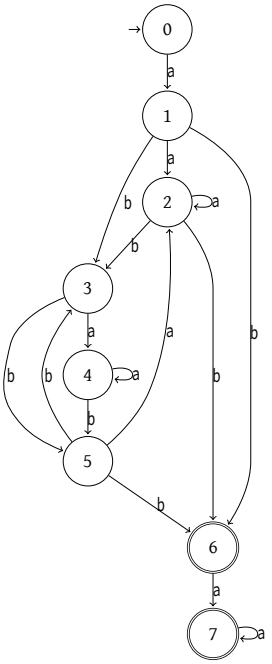
DS5-corrigé-Durée 4h

06 février 2025

1 Exercice 1 : Racine carrée d'un langage

1. Clairement, $\{a\}^* \cup \{b\}^* \subset \sqrt{L}$. Soit u un mot contenant un a ainsi qu'un b. Alors uu présente un a (dans le deuxième u) après un b (du premier u), donc n'est pas dans L. Finalement, $\sqrt{L} = \{a\}^* \cup \{b\}^*$.
2. De même que précédemment, $\sqrt{L} \cap \Sigma^*\{b\}\Sigma^*\{a\}\Sigma^* = \emptyset$ et $\sqrt{L} \cap \Sigma^*\{a\}\Sigma^*\{b\}\Sigma^* = \emptyset$. On trouve encore $\sqrt{L} = \{a\}^* \cup \{b\}^*$.
3. Il suffit de prendre $A_{3,\{1\}}$, par définition.
4. (a) L' est décrit par l'expression régulière sur l'alphabet approprié $a_1(a_2 + b_3a_4^*b_5)^*b_6a_7^*$.
   (b) $P = \{a_1\}$, $S = \{b_6, a_7\}$, $F = \{a_1a_2, a_1b_3, a_2a_2, a_2b_3, b_3a_4, b_3b_5, a_4a_4, a_4b_5, b_5a_2, b_5b_3, a_1b_6, a_2b_6, b_5b_6, b_6a_7, a_7a_7\}$.
   (c) Le langage L' ne contient pas le mot vide, donc son automate de Glushkov est :



5. On applique l'algorithme classique. Seuls les états accessibles sont considérés

	a	b
$\{0\}$	$\{1\}$	
$\{1\}$	$\{2\}$	$\{3,6\}$
$\{2\}$	$\{2\}$	$\{3,6\}$
$\{3,6\}$	$\{4,7\}$	$\{5\}$
$\{4,7\}$	$\{4,7\}$	$\{5\}$
$\{5\}$	$\{2\}$	$\{3,6\}$

Les états finals sont $\{3,6\}$ et $\{4,7\}$.

6.

$$\begin{aligned} u \in \sqrt{L} &\Leftrightarrow uu \in L \\ &\Leftrightarrow \delta^*(q_0, uu) \in F \\ &\Leftrightarrow \delta^*(\delta^*(q_0, u), u) \in F \\ &\Leftrightarrow \exists q \in Q, \delta^*(q_0, u) = q \wedge \delta^*(q, u) \in F \\ &\Leftrightarrow \exists q \in Q, \delta^*(q_0, u) \in \{q\} \wedge \delta^*(q, u) \in F \\ &\Leftrightarrow u \in L_{q_0,\{q\}} \wedge u \in L_{q_0,F}. \end{aligned}$$

7. On a alors $\sqrt{L} = \bigcup_{q \in Q} L_{q_0,\{q\}} \cap L_{q_0,F}$: c'est une union finie d'intersections finies de langages rationnels, c'est donc un langage rationnel.
8. Soit $u \in \sqrt{L} \odot \sqrt{L}$. Il existe $v \in \sqrt{L}$ tel que $u = vv$. Mais $vv \in L$ par définition de $\sqrt{L}$, donc $u \in L$. D'où l'inclusion.

2 Exercice 2

1. Cette grammaire est ambiguë car le mot $babab \in L(G)$ admet les deux arbres syntaxiques suivants :

   Or ces derniers sont différents.

2. On constate que $L(G)$ est rationnel car dénoté par l'expression rationnelle $(ba)^\star b$.
3. Montrons par récurrence forte sur $n \in \mathbb{N}^\star$ la propriété $H(n)$ suivante : si $u \in L$ est un mot de taille n alors u est engendré par la grammaire G. C'est bien sûr le cas pour $n = 1$ puisque le seul mot de L de cette taille est b, engendré par la deuxième règle de G.

   Soit donc $n \in \mathbb{N}^\star$ et u un mot de L de taille $n+1$. Comme $|u| \geq 2$, ba est nécessairement préfixe de u et il existe donc $v \in (ba)^\star b = L$ tel que $u = bav$. Par hypothèse, ce mot v est engendré par G : il existe une dérivation telle que $S \Rightarrow^\star v$. On en déduit que

$$S \Rightarrow SaS \Rightarrow baS \Rightarrow^\star bav = u$$

   est une dérivation licite et donc que $u \in L(G)$. Cette récurrence montre que $L \subset L(G)$.

   Montrons réciproquement que $L(G) \subset L$ en montrons par récurrence forte sur $n \in \mathbb{N}^\star$ la propriété $H(n)$ suivante : si $u \in \Sigma^\star$ se dérive de S en n dérivations alors $u \in L$. C'est acquis pour $n = 1$: le seul mot de $\Sigma^\star$ qu'on peut obtenir en une dérivation est $b \in L$.

Soit donc $n \in \mathbb{N}^\star$ et u un mot dans $L(G)$ tel que $S \Rightarrow^{n+1} u$. Comme ce mot est obtenu en au moins 2 dérivations, les règles de G nous informent que la première est nécessairement $S \to SaS$ (sans quoi ce serait $S \to b$ et dans ce cas u serait obtenu en une seule dérivation). Donc la dérivation permettant d'obtenir u se décompose en :

$$S \Rightarrow SaS \Rightarrow^n u$$

On en déduit qu'il existe $v, w \in \Sigma^\star$ et $k_1, k_2 \in [\![1, n]\!]$ tels que $u = vaw$, $S \Rightarrow^{k_1} v$, $S \Rightarrow^{k_2} w$ et $k1 + k2 = n$. L'hypothèse de récurrence (forte) s'applique à v et w et on en déduit que ces deux mots appartiennent au langage dénoté par $(ba)^\star b$ donc qu'il existe $r_1, r_2 \in \mathbb{N}$ tels que $v = (ba)^{r_1} b$ et $w = (ba)^{r_2} b$. Par conséquent, $u = (ba)^{r_1} ba (ba)^{r_2} b = (ba)^{r_1+r_2+1} b \in L$.

4. On sait à présent que $L(G) = (ba)^\star b$; il s'agit donc de trouver une grammaire non ambiguë engendrant ce langage. On peut proposer par exemple la grammaire dont les règles sont :

$$S \to Tb \qquad T \to baT \,|\, \varepsilon$$

les règles sur T permettant de générer le facteur dans $(ba)^\star$ et la première de rajouter le b final.

Cette grammaire G' est non ambiguë car pour tout mot dans $L(G')$, il existe une unique dérivation permettant de le construire (donc évidemment un seul arbre syntaxique) ; cette unicité découlant du fait que dans cette grammaire un non terminal se dérive toujours en un mot qui contient au plus un seul non terminal.

5. La question demande de montrer que tout langage rationnel peut être engendré par une grammaire non ambiguë. Soit donc L un langage rationnel. Par le théorème de Kleene, il existe un automate fini $A = (\Sigma, Q, \{q_0\}, F, \delta)$ qui reconnaît L qu'on peut loisiblement supposer déterministe.

Considérons alors la grammaire dont les non terminaux sont $\{V_q \,|\, q \in Q\}$, l'axiome est V_{q_0}, les terminaux sont les lettres de Σ et dont les règles sont données par :

- Pour toute transition $q \xrightarrow{a} q'$ dans A, on ajoute la règle $V_q \to aN_{q'}$.
- Pour tout $q \in F$, on ajoute la règle $V_q \to \varepsilon$.

Cette grammaire engendre L de manière non ambiguë graĉe au déterminisme de A.

3 Problème

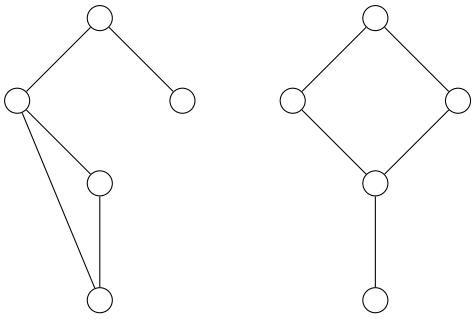
1. La somme des degrés d'un graphe est paire car égale à deux fois le nombre d'arêtes donc le fait que la somme des valeurs des degrés soit paire est un condition nécessaire.
2. Si $(d_2, \ldots, d_\ell)$ est graphique alors soit $G = (S, A)$ un graphe simple associé. On pose $G' = (S \cup \{s_1\}, A')$ avec $A' = A \cup \bigcup_{x \in \{2, d_1+1\}} (s_1, x)$, autrement dit on rajoute un sommet que l'on relie aux d_1 premiers sommets de la suite. Ainsi les degrés du graphe obtenu sont $(d_1, d_2+1, \ldots, d_{d_1+1}+1, d_{d1+2}, \ldots, d_\ell)$ (qui est bien graphique à condition d'avoir $d_1 \geq d_2 + 1$ et $d_{d_1+1} > d_{d_1+2}$, mais ce n'est pas demandé). Cette construction est valable pour toute valeur de d_1 avec $d_1 \leq \ell - 1$ car il faut avoir suffisamment de sommets à relier au nouveau sommet.
3. Soit $(d_1, \ldots, d_\ell)$ graphique.
   Il existe un graphe G dont c'est la suite des degrés des sommets.
   Raisonnons par l'absurde, et supposons que qu'il n'existe pas de graphe tel que dans l'énoncé. Choisissons alors un graphe G dans lequel s_1 est adjacent à un maximum de sommet de $(2, \ldots, d_1 + 1)$, mais pas à tous.
   On choisit un tel sommet s_j tel que j minimal.
   On montre qu'il existe $k > j$ et $l \in [n]$ avec $l \neq j, l \neq k, l \neq 1$ tels que :
   — $\{s_1, s_k\} \in A$
   — $\{s_l, s_j\} \in A$
   — $\{s_k, s_l\} \notin A$
   — Tout d'abord, comme s_1 a d_1 voisins, et que s_j n'est pas voisin de s_1, s_1 a nécessairement un sommet voisin s_k où $k > j$, sans quoi l'ensemble des voisins de s_1 serait $\{s_i, i \in [j-1]\}$, au nombre de $j - 1 \leq d_1 - 1 < d_1$, ce qui est impossible.
   — On cherche maintenant à montrer qu'il existe un voisin s_l de s_j qui n'est pas voisin de s_k.
   Pour cela, on remarque que $d(s_k) = d_k \leq d(s_j) = d_j$.
   Par ailleurs, s_1 est voisin de s_k mais pas de s_j.
   Notons S'_k (respectivement S'_j) l'ensemble des sommets autres que $k, j, 1$ qui soient voisins de s_k (respectivement de s_j).
      — Si s_j et s_k sont voisins, $|S'_k| = d_k - 2 < d_j - 1 = |S'_j|$.

— Si s_j et s_k ne sont pas voisins, $|S'_k| = d_k - 1 < d_j = |S'_j|$.

Dans tous les cas, on peut conclure que $S'_j \setminus S'_k \neq \emptyset$: il existe donc bien un sommet s_l voisin de s_j mais pas de s_k.

On construit alors G' à partir de G en retirant les arêtes s_1s_k et s_ls_j, et en ajoutant les arêtes s_1s_j et s_ls_k. Tous les degrés sont préservés par cette transformation, donc on obtient G' dont la liste des degrés est bien celle donnée, et qui contredit la minimalité de j.

4. On enlève le premier sommet et on obtient un graphe avec la liste des sommets demandés. L'hypothèse garantit que la liste obtenue est bien décroissante.
5. On applique le théorème : $[4;3;3;3;3]$ est graphique ssi $[2;2;2;2]$ l'est et on voit bien qu'un cycle de longueur 4 convient.

   De même, $[6;4;4;2;2;1;1]$ est graphique ssi $[3;3;1;1;0;0]$ l'est ssi $[2;0;0;0;0]$ l'est. Or on voit bien qu'on ne peut pas construire un graphe avec ces degrés puisque le sommet de degré 2 ne peut être relié à personne.
6. ```
   let compare_entiers n y =
    if m <n then 1
    else if m >n then -1
    else 0;;
   ```
7. ```
   let rec decr_n n l = match l with
   |[]-> if n=0 then Some [] else None
   |h::t when n = 0 -> (match (decr_n 0 t) with
                      |None-> None
                      |Some(lbis)->  Some (h::lbis))
   |h::t -> (match decr_n (n-1) t with
             |None-> None
             |Some(lbis)-> Some ((h-1)::lbis));;
   ```
8. ```
   let rec havel_hakimi l = match l with
      |[]->true
      |h::t-> match decr_n h t with
             |None-> false
             |Some(lp)-> havel_hakimi (List.sort compare lp)
   ```
9. 
10. $\lambda^* = (6,2,2,2,1)$.
11. $\alpha_1 = 3, \alpha_2 = 2, \beta_1 = 4$ et $\beta_2 = 2$.
12. 5 est inséré en fin de la première ligne et on obtient :

1	1	2	2	4	5
2	3	3			
3					
4					

    Pour insérer 1, on remplace le premier 2 de la première ligne par 1, on l'insère ensuite dans la deuxième ligne à la place de 3 qui est lui aussi propagé et ceci jusqu'à obtenir :

1	1	1	2	4
2	2	3		
3	3			
4				

13. La propriété qui permet de montrer la correction de cet algorithme est que :

$$\forall (i,j), \forall j' \geq j, T(i,j) < T(i+1,j')$$

    Ainsi, lorsque l'on va insérer l'élément $T(i,j)$ dans la ligne $i+1$ ce sera dans une colonne qui précède au sens large la colonne j et ainsi on peut garantir que $T(i,j)$ est bien plus grand strictement que la valeur qui est au dessus d'elle.
   ```

14.
```
let rec insereligne k l = match l with
    |[]-> 0,[k]
    |h::t when h<=k-> let mb,rb = insereligne k t in mb, h::rb
    |h::t-> h,k::t
```

15.
```
let rec insertion k t = match t with
    |[]->[]
    |l1::next -> let m,new = insereligne k l1 in
                 if m=0 then newl::next
                 else newl::(insertion m next)
```

16.
```
 let rec construit l = match l with
     |[]-> []
     |h::t-> let tab = construit t in insertion h tab
```

17. On obtient : $\begin{array}{c|c|c|c|c|c} 2 & 4 & 5 & 5 & 6 & 6 \\ 1 & 1 & 3 & 2 & 5 & 3 \end{array}$

18. $\lambda_i^* = \beta_i + i = \alpha_i + 1 + i = \lambda_i + 1$.

19. $(s,t) \to (t+1, s)$ convient.

20. Il suffit de suivre pas à pas en utilisant la dernière case d'insertion pour (s,t) qui n'est pas celle où est inséré k mais la dernière valeur propagée.

21. En notant i l'entier dont il est question, cela correspond au degré de i dans le graphe initial.