Exercice 1: Jeu du jeton

Définition

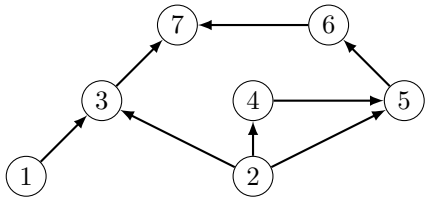
Pour $G = (S, A)$ un graphe orienté et $s \in S$, les **prédécesseurs** de s sont les sommets $t \in S$ tels que $(t, s) \in A$ et ses **successeurs** sont les $u \in S$ tels que $(s, u) \in A$. Pour la suite, les graphes considérés seront supposés sans cycle.

On s'intéresse au problème suivant : on fixe un sommet $x \in S$, appelé **sommet distingué**. On dispose d'un ensemble de **jetons** qui peuvent être placés sur les sommets du graphe, de sorte que chaque sommet ne comporte que zéro ou un jeton. S'il a un jeton, le sommet est dit **marqué**. On peut placer les jetons selon les règles suivantes :

(a) si s n'a aucun prédécesseur, on peut placer un jeton sur s ;
(b) si tous les prédécesseurs de s sont marqués, on peut placer un jeton sur s ;
(c) on peut toujours retirer un jeton de n'importe quel sommet.

Une **étape** du jeu consiste à placer ou retirer un unique jeton en suivant ces règles. À la fin du jeu, on souhaite qu'il ne reste qu'un seul jeton, placé sur le sommet distingué.

1. On considère le graphe suivant, où 7 est le sommet distingué. Montrer qu'il existe une stratégie utilisant au plus 4 jetons.



2. Montrer qu'un graphe orienté sans cycle contient au moins un sommet sans prédécesseur.
3. En déduire que pour tout graphe orienté sans cycle à n sommets, il existe toujours une stratégie pour résoudre le jeu avec n jetons.
4. On considère un arbre binaire strict (chaque nœud interne a deux enfants) à ℓ feuilles, dans lequel les prédécesseurs d'un sommet sont ses enfants et le sommet distingué est la racine. Montrer qu'il existe une stratégie utilisant au plus $\lceil \log_2 \ell \rceil + 2$ jetons.

On remplace maintenant la règle (c) par les deux règles suivantes :

(c) on peut toujours retirer un jeton d'un sommet sans prédécesseurs ;
(d) on ne peut retirer un jeton d'un sommet que si ses prédécesseurs sont tous marqués.

À partir de maintenant, on considère uniquement le graphe ligne $L_n := s_1 \to s_2 \to \ldots \to s_n$, où s_n sera toujours le sommet distingué.

5. Montrer qu'avec ces nouvelles règles :
   - Il existe toujours une stratégie pour résoudre le jeu en utilisant n jetons et $\mathcal{O}(n)$ étapes.
   - S'il existe une stratégie pour marquer s_n en k jetons et t étapes, alors il existe une stratégie partant de la configuration finale (où s_n est le seul sommet marquer) pour retrouver la configuration initiale (aucun sommet marqué) en k jetons et t étapes.
6. Montrer qu'on peut marquer s_n en $\mathcal{O}(n)$ étapes et $\mathcal{O}(\sqrt{n})$ jetons.
7. Montrer qu'on peut marquer s_n en utilisant au total $\lceil \log_2 n \rceil + 1$ jetons. Combien d'étapes comporte cette stratégie ?

On cherche maintenant à obtenir un meilleur compromis entre le nombre de jetons utilisés et le nombre d'étapes.

8. Montrer que pour tout $\varepsilon > 0$, il existe une stratégie utilisant $\mathcal{O}(\log n)$ jetons et $\mathcal{O}(n^{1+\varepsilon})$ étapes.

Corrigé

1. On marque $1, 2$ puis 3, on retire le jeton de 1, on marque 4 puis 5, on retire le jeton de 2 puis de 4, on marque 6 puis 7, on retire les jetons de 3, 5 et 6.
2. Supposons par l'absurde que tout sommet admet au moins un prédécesseur. On pose $s_0 \in S$ un sommet arbitraire. Pour $i \in \mathbb{N}$, on pose s_{i+1} un prédécesseur de s_i (existe par hypothèse). Alors par le principe des tiroirs, il existe $0 \leqslant i < j \leqslant n$ tel que $s_i = s_j$. Dès lors, $(s_j, s_{j-1}, \ldots, s_{i+1}, s_i)$ est un cycle, ce qui est absurde.
3. On pose les jetons selon un ordre topologique, puis on les retire tous, sauf le dernier. Un ordre topologique existe d'après la question précédente.
4. Montrons par récurrence sur ℓ qu'il existe une stratégie utilisant au plus $\lceil \log_2 \ell \rceil + 2$ jetons :
   - si $\ell = 1$, le nombre de jetons nécessaires est $1 = \lceil \log_2 \ell \rceil + 1 \leqslant \lceil \log_2 \ell \rceil + 2$;
   - supposons le résultat établi pour un nombre de feuilles $< \ell$ (avec $\ell > 1$) et soit $T = N(g, d)$ un arbre binaire strict avec ℓ feuilles. Sans perte de généralité, g a un nombre de feuilles ℓ' vérifiant $\frac{\ell}{2} \leqslant \ell' < \ell$. Par hypothèse de récurrence, on peut marquer la racine de g en utilisant au plus $\lceil \log_2 \ell' \rceil + 2 \leqslant \lceil \log_2 \ell \rceil + 2$ jetons. En retirant tous les marqueurs sauf celui sur la racine de g, on peut marquer la racine de d en utilisant au plus $\lceil \log_2(\ell - \ell') \rceil + 2 \leqslant \lceil \log_2 \ell \rceil + 1$ jetons. On peut alors retirer tous les marqueurs sauf ceux des racines de g et d et marquer la racine de T. Le nombre total de jetons utilisés est bien inférieur ou égal à $\lceil \log_2 \ell \rceil + 2$

   On conclut par récurrence.
5. - On place les jetons dans l'ordre $s_1, s_2, \ldots, s_n$, puis on les retire dans l'ordre $s_{n-1}, s_{n-2}, \ldots, s_1$.
   - Les conditions pour placer et retirer un jeton sont les mêmes. Il suffit d'appliquer la même stratégie dans l'ordre inverse, en inversant pose et retrait de jeton.
6. On pose $m = \sqrt{n}$ (on omet les parties entières). On découpe le graphe en m portions de m sommets. Avec la stratégie naïve, on peut marquer le sommet $s_{i \times m}$ en m jetons, pour $i \in [\![1, m]\!]$. Chaque marquage se fait en $\mathcal{O}(\sqrt{n})$ étapes, soit au total $\mathcal{O}(n)$ étapes, avec $\mathcal{O}(\sqrt{m})$ jetons.
7. On montre qu'il existe une stratégie pour marquer tout sommet s_i avec $i \leqslant 2^k$(et uniquement ce sommet) avec moins de $k + 1$ jetons.
   - C'est vrai pour $k = 0$;
   - on suppose le résultat vrai pour tout rang $< k$, $k > 0$ fixé. Pour marquer un sommet $2^k < i \leqslant 2^{k+1}$, on marque le sommet 2^k (et uniquement lui) en moins de k jetons. On marque alors s_i en utilisant la stratégie pour $i - 2^k$, ce qui nécessite $k + 1$ jetons (les k jetons pour $i - 2^k$, plus le jeton en 2^k). On reprend alors la stratégie inverse pour enlever le jeton en 2^k.

   On conclut par récurrence.

   Le nombre d'étapes $T(k)$ vérifie $T(k + 1) = 3T(k)$, soit $T(k) = 3^k$. On en déduit que pour marquer s_n, il faut $n^{\log_2 3}$ étapes.
8. Au lieu d'utiliser une base 2, on peut utiliser une base b pour le découpage. Cela utilise, pour marquer un sommet $\leqslant b^k$, un nombre de jetons inférieur ou égal à $(b - 1) \times k + 1$ (on marque les sommets b^{k-1}, $2b^{k-1}$, …, b^k). La relation de récurrence vérifie $T(k) \leqslant 2bT(k - 1)$, soit $T(k) \leqslant (2b)^k$. En posant $n = b^k$, le nombre de jetons nécessaires est $(b - 1)k = \mathcal{O}(\log n)$. Le nombre d'étapes est inférieur à $n^{1+\ln 2/\ln b}$. Pour $b = 2^{\frac{1}{\varepsilon}}$, on obtient le résultat attendu.

Exercice 2: Automates et palindromes

Dans ce sujet, on fixe Σ un alphabet de taille au moins 2.

Définition

Pour $u = a_0 \dots a_{n-1} \in \Sigma^*$, on appelle **miroir de** u le mot $\overline{u} = a_{n-1} \dots a_0$. Un mot u est un **palindrome** si $u = \overline{u}$. On note $\Pi \subseteq \Sigma^*$ le langage des palindromes et $\Pi_n = \Pi \cap \Sigma^n$.

1. Soit A un automate fini déterministe complet à k états. Montrer que pour $n \in \mathbb{N}$, $L(A) \cap \Sigma^{2n} = \Pi_{2n} \Rightarrow k \geqslant |\Sigma|^n$.
2. En déduire que Π n'est pas rationnel.
3. Soit A un automate fini. $L(A) \cap \Pi$ est-il rationnel ?
4. Soit A un automate fini. $\{u \in \Sigma^* | u\overline{u} \in L(A)\}$ est-il rationnel ?

Définition

On note Π_{pair} le langage des palindromes de longueur paire : $\Pi_{\mathsf{pair}} = \bigcup_{n \in \mathbb{N}} \Pi_{2n}$.

5. Proposer un algorithme qui, étant donné un automate fini A, détermine si $L(A) \cap \Pi_{\mathsf{pair}}$ est vide, fini ou infini. Préciser la complexité en temps et en espace.
6. Modifier l'algorithme précédent pour renvoyer le cardinal de $L(A) \cap \Pi_{\mathsf{pair}}$ lorsqu'il est fini, en supposant que l'automate donné en entrée est déterministe. La complexité est-elle modifiée ?
7. Modifier les algorithmes précédents pour $L(A) \cap \Pi$.

Corrigé

1. Supposons $L(A) \cap \Sigma^{2n} = \Pi_{2n}$. Supposons qu'il existe $u \neq v$ deux mots de Σ^n tels que $\delta^*(q_0, u) = \delta^*(q_0, v)$. Sachant que $L(A) \cap \Sigma^{2n} = \Pi_{2n}$, on en déduit que $u\overline{u} \in L(A)$ et que $v\overline{v} \in L(A)$. Cela implique que $\delta^*(\delta^*(q_0, u), \overline{u}) \in F$, soit $\delta^*(\delta^*(q_0, v), \overline{u}) \in F$, soit $v\overline{u} \in L(A)$. C'est absurde, car $v\overline{u}$ n'est pas un palindrome, mais il est de taille $2n$ et dans $L(A)$. On en déduit que $\forall u, v \in \Sigma^n, u \neq v \Rightarrow \delta^*(q_0, u) \neq \delta^*(q_0, v)$. En particulier, cela implique qu'il y a au moins $|\Sigma^n| = |\Sigma|^n$ états.
2. Si Π est rationnel et A un automate fini qui le reconnait, alors la propriété précédente s'applique pour toute valeur de $n \in \mathbb{N}$. Cela implique qu'il y a un nombre infini d'états, ce qui est absurde.
3. Cela est impossible, par exemple si $L(A) = \Sigma^*$ (qui est bien reconnaissable).
4. Soit $A = (Q, \Sigma, \delta, I, F)$ un automate fini pas nécessairement déterministe. Pour $q \in Q$, posons $A_q = (Q, \Sigma, \delta, I, \{q\})$, qui reconnait $\{u \in \Sigma^*, q \in \delta^*(I, u)\}$. Posons également $\overline{A}_q = (Q, \Sigma, \overline{\delta}, F, \{q\})$, où $\overline{\delta}$ est définie de la façon suivante :

   $$\forall p, p' \in Q, \forall a \in \Sigma, p' \in \delta(p, a) \Leftrightarrow p \in \overline{\delta}(p', a)$$

   c'est-à-dire où l'on retourne les transitions de A. Une preuve simple permet de voir que $u \in L(\overline{A}_q) \Leftrightarrow \delta^*(q, \overline{u}) \cap F \neq \emptyset$.

   On répond à la question par l'affirmative en remarquant que $\{u \in \Sigma^* | u\overline{u} \in L(A)\} = \bigcup_{q \in Q} L(A_q) \cap L(\overline{A}_q)$.

   Une autre construction qui pourra servir pour la suite est de considérer $\overline{A} = (Q, \Sigma, \overline{\delta}, F, I)$, puis l'automate produit $A \times \overline{A} = (Q^2, \Sigma, \delta_\times, I \times F, F \times I)$ et enfin sa restriction en limitant les états finaux à $\{(q, q) | q \in Q\}$.
5. On remarque que $|L(A) \cap \Pi_{\mathsf{pair}}| = |\{u \in \Sigma^* | u\overline{u} \in L(A)\}|$, car $f : \Sigma^* \to \Pi_{\mathsf{pair}}$, définie par $u \mapsto u\overline{u}$ définie une bijection de $\{u \in \Sigma^* | u\overline{u} \in L(A)\}$ vers $L(A) \cap \Pi_{\mathsf{pair}}$. Le problème revient donc à déterminer si ce langage est vide, fini ou infini. La construction de l'automate produit décrit précédemment peut se faire en temps et espace quadratique. On peut alors l'émonder, tester s'il existe un état final accessible et tester l'existence d'un cycle, le tout en temps linéaire par plusieurs parcours de graphes.
6. Si l'automate est déterministe et que le langage cherché est fini, on peut procéder par programmation dynamique : le nombre de mots reconnus depuis un état $q \in Q$ est 1 ou 0 selon que l'état soit final ou non, plus le nombre de mots reconnus depuis chaque état vers lesquels q a une transition sortante (quitte à compter plusieurs fois si les transitions sont étiquetées par plusieurs lettres). Pour l'ordre des calculs,

il suffit d'utiliser un ordre donné par un tri topologique (car l'automate ne contient pas de cycle), en commençant par la fin. Le calcul aurait alors une complexité linéaire en temps et en espace, en la taille de l'automate produit (c'est-à-dire quadratique en la taille de l'automate initial), ce qui n'augmente pas la complexité totale.

Notons que si l'automate n'est pas déterministe il faudrait le déterminiser, ce qui mènerait à une complexité exponentielle. Compter le nombre de mots reconnus par un automate quelconque est ♯P-difficile.

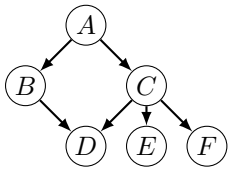
7. Pour traiter les palindromes de longueur impaire, on construit, pour chaque $a \in \Sigma$, un automate produit comme en 4, en ne considérant comme états finaux que les (q, q') où $q' \in \delta(q, a)$. Un tel automate permettra de reconnaitre les mots u tels que $ua\overline{u} \in L(A)$. On somme alors tous les cardinaux pour obtenir la valeur cherchée. La complexité est la même, en rajoutant un facteur $|\Sigma|$.

Exercice 3: Étiquetage d'accessibilité

Définition

Soit $G = (V, E)$ un graphe orienté. Un chemin dans G est une suite finie $v_1 \dots v_n$ d'éléments de V telle que $\forall 1 \leqslant i < n$, on a $(v_i, v_{i+1}) \in E$. Pour $u, v \in V$, on écrit $u \rightsquigarrow v$ quand il existe un chemin de u à v.
Un **étiquetage** de G est une fonction $f : V \longrightarrow \mathbb{N}^2$. Étant donnés $t_1 = (p_1, q_1)$ et $t_2 = (p_2, q_2)$ des éléments de $\mathbb{N}^2$, on écrit $t_1 \leqslant t_2$ lorsque $p_1 \leqslant p_2$ et $q_1 \leqslant q_2$. Un étiquetage f est appelé **étiquetage d'accessibilité** s'il satisfait la propriété : $\forall u \neq v \in V, u \rightsquigarrow v \Leftrightarrow f(u) \leqslant f(v)$.

1. Construire un étiquetage d'accessibilité pour le graphe suivant :



2. Rappeler la définition de composante connexes et composante fortement connexe (CFC) d'un graphe orienté G et la reformuler en utilisant $\rightsquigarrow$.

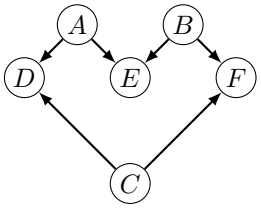
Définition

Soit $G = (V, E)$ un graphe orienté. On défini le **graphe des CFC** de G comme le graphe orienté $G_C = (V_C, E_C)$ où V_C est l'ensemble des CFC de G et où E_C est défini par :

$$E_C := \{(K_1, K_2) \in V_C \times V_C | (K_1 \neq K_2) \wedge (\exists v_1 \in K_1, v_2 \in K_2 \text{ tels que } v_1 \rightsquigarrow v_2)\}$$

3. Expliquer pourquoi il n'existe pas de cycle dans G_C.

   Montrer que G admet un étiquetage d'accessibilité si et seulement si G_C admet un étiquetage d'accessibilité.
4. Montrer que G admet un étiquetage d'accessibilité si et seulement si chacune de ses composantes connexes admet un étiquetage d'accessibilité.
5. Montrer que le graphe suivant d'admet pas d'étiquetage d'accessibilité :



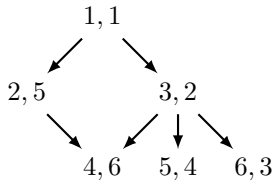
Définition

On dit que deux sommets $u \neq v$ d'un graphe acyclique G sont **comparables** si $u \rightsquigarrow v$ ou $v \rightsquigarrow u$. On dit qu'un graphe orienté $G' = (V, E')$ est **conjugué** à $G = (V, E)$ s'il est acyclique et si, pour tous $u, v \in V$, si $u \neq v$, alors il y a exactement un graphe parmi G et G' dans lequel u et v sont comparables.

6. Montrer que si G a un conjugué G', alors G admet un étiquetage d'accessibilité. Expliquer comment le calculer à partir de G'.
   Indication : On montrera que $(V, E \cup E')$ est acyclique.
7. Montrer que, réciproquement, si G est acyclique et admet un étiquetage d'accessibilité, alors il admet un conjugué.

Corrigé

1. Une solution possible est :



2. Les composantes connexes sont les classes d'équivalence de la clôture symétrique, réflexive et transitive de $\rightsquigarrow$. Les CFC sont les classes d'équivalence de la relation d'équivalence définie par $u \rightsquigarrow v \wedge v \rightsquigarrow u$.
3. Si on suppose qu'il existe un cycle dans G_C, par exemple $K_1, \ldots, K_n$, alors pour tous i, j, pour tous $u_i, u_j \in K_i \times K_j$, $u_i \rightsquigarrow u_j$ dans G, et donc tous les éléments des K_i sont dans la même CFC.
   Supposons que G admet un étiquetage d'accessibilité, noté f. Soit $\phi : V_C \to V$ qui à une CFC associe un représentant de la CFC. Soit $f_C = f \circ \phi$. On observe facilement que f_C est un étiquetage d'accessibilité. Réciproquement, soit f_C un étiquetage d'accessibilité de G_C. On pose $f : v \mapsto f_C(K_v)$ où K_v est la CFC de v. À nouveau, on vérifie facilement les hypothèses.
4. Si G admet un étiquetage d'accessibilité, il est clair que c'est le cas pour chacune de ses CC (en considérant la restriction).
   Supposons que chaque CC de G possède un étiquetage d'accessibilité. Procédons par récurrence sur le nombre k de CC.
   - $k = 1$ est trivial (c'est G lui-même).
   - Supposons le résultat vrai pour tout graphe avec strictement moins de $k \in \mathbb{N}$ CC, $k \geqslant 2$. Partitionnons G en G_1 et G_2 en regroupant des CC dans l'une ou l'autre des deux parties. Chacun des G_1, G_2 possèdent moins de k CC, donc par HR, il existe $f_1 = (p_1, q_1)$ et $f_2 = (p_2, q_2)$ des étiquetages d'accessibilité pour ces deux sous-graphes. Choisissons M plus grand que tous les entiers apparaissant dans les images de f_1 et f_2, et posons $f : v \mapsto \begin{cases} (p_1(v) + M, q_1(v)) & \text{si } v \in V_1 \\ (p_2(v), q_2(v) + M) & \text{sinon} \end{cases}$. Alors f est bien un étiquetage d'accessibilité pour G.
5. On dit que $u, v \in V$ sont incomparables s'il n'existe pas de chemin de l'un à l'autre. Dans ce graphe, les couples incomparables sont (A, B), (A, C), (A, F), (B, C), (B, D), (C, E), (D, E), (D, F), (E, F).
   Procédons par l'absurde et supposons $f = (p, q)$ un étiquetage d'accessibilité. Pour deux sommets incomparables, on a forcément l'une des situations :
   - $p(u) < p(v)$ et $q(u) > q(v)$, qu'on notera $u \lhd v$
   - $u \lhd v$

   Notons alors A, B, C par u_1, u_2 et u_3, et D, E, F par v_1, v_2, v_3. On a que (u_i, v_i) est incomparable. Deux cas sont alors possibles : soit $u_i \lhd v_i$ pour deux valeurs différentes de $i \in \{1, 2, 3\}$, soit $v_i \lhd u_i$ pour deux valeurs différentes de $i \in \{1, 2, 3\}$. Dans le premier cas, soient $i \neq j$ tels que $u_i \lhd v_i$ et $u_j \lhd v_j$. Comme v_i et v_j sont incomparables, on a soit $v_i \lhd v_j$, soit $v_j \lhd v_i$. Dans le premier sous-cas, on a donc $q(v_i) > q(v_j)$ et par transitivité $q(u_i) > q(v_j)$ mais ceci contredit le fait que $f(u_i) \leqslant f(v_j)$ puisque $u_i \rightsquigarrow v_j$. Tous les autres cas se traitent de manière similaire.
6. Montrons que $(V, E \cup E')$ est acyclique. Supposons par l'absurde qu'il possède un cycle. Comme G et G' sont acycliques, ce cycle doit passer par des arêtes de G et de G'. En considérant le chemin franchi par le cycle dans G et dans G', on peut écrire le cycle comme $v_1 \ldots v_n$ où $n \geqslant 2$ est impair, et si i est impair, alors $v_i \rightsquigarrow v_{i+1}$ dans G et si i est pair, $v_i \rightsquigarrow v_{i+1}$ dans G' (ces cas sont exclusifs par définition du conjugué). Choisissons le cycle qui minimise n.
   Tout d'abord, v_1 et v_2 sont comparables dans G et si $v_3 = v_1$, alors v_1 et v_2 sont comparables dans G', ce qui est absurde. On en déduit que $v_1 \neq v_3$ et que $n \geqslant 5$. Ainsi, soit v_1 et v_3 sont comparables dans G, soit dans G'.
   - Dans le premier cas, on a forcément $v_1 \rightsquigarrow v_3$ dans G (sinon, on aurait $v_3 \rightsquigarrow v_3$ dans G), mais comme $v_3 \rightsquigarrow v_4$ dans G, alors $v_1 \rightsquigarrow v_4$ dans G. Ainsi, $v_1 v_4 \ldots v_n$ est un cycle a moins d'alternances qu'initialement (absurde).
   - Dans le second cas, on montre de même que $v_{n-1} v_3 \ldots v_n$ est un cycle avec moins d'alternances.

   Montrons à présent que si G a un conjugué G', alors G admet un étiquetage d'accessibilité. Notons G'_R le graphe obtenu à partir de G' en retournant les arêtes. Par symétrie, ce graphe est toujours acyclique

et il est clair que c'est également un conjugué de G. Ainsi, les graphes orientés $G'' = (V, E \cup E')$ et $G''_R = (V, E \cup E'_R)$ sont tous deux acycliques.

Soit alors $p : V \to \mathbb{N}$ une fonction injective telle que si $u \rightsquigarrow v$ dans G'', alors $p(u) < p(v)$ (existe car le graphe est acyclique). On construire de même q pour G''_R et on pose $f = (p, q)$.

On montre alors que f est bien un étiquetage d'accessibilité. Si $u \rightsquigarrow v$ dans G, alors dans G'' et G''_R également, et donc on a bien $f(u) \leqslant f(v)$. Sinon, $u \rightsquigarrow v$ dans G' et $v \rightsquigarrow u$ dans G'_R (ou l'inverse) et donc on a ni $f(u) \leqslant f(v)$, ni $f(v) \leqslant f(u)$.

7. Soit $f = (p, q)$ un étiquetage d'accessibilité de G. Soit $G' = (V, E')$ tel que pour $u \neq v$, $(u, v) \in E'$ si et seulement si u et v sont incomparables dans G et $p(u) < p(v)$. Montrons que G' est bien un conjugué de G, c'est-à-dire qu'il est acyclique et une paire est comparables dans G' si et seulement si elle est incomparable dans G.

   Il est clair que G' est acyclique (car $p(u) < p(v)$ pour toute arête de G). L'implication retour découle de la définition de E'. Il ne reste que l'implication directe :

   Soient $u \rightsquigarrow v$ deux sommets comparables dans G'. Soit $u = w_1 \dots w_n = v$ un tel chemin. Par définition de G', on a forcément $q(w_i) > q(w_{i+1}$ (car w_i et w_{i+1} sont incomparables dans G) par transitivité, $q(u) > q(v)$, et donc u et v sont incomparables dans G (car $p(u) < p(v)$).

Exercice 4: Shuffle d'un langage

Définition

Soit Σ un alphabet. Pour $u \in \Sigma^*$ et $A \subseteq \Sigma$, on note $\mathsf{proj}(u, A)$ la **projection** de u sur A, obtenue en supprimant de u toutes les lettres qui ne sont pas dans A. Par exemple, $\mathsf{proj}(abcabc, \{a, b\}) = abab$. Pour L un langage, on note $\mathsf{proj}(L, A) = \{\mathsf{proj}(u, A) | u \in L\}$.

1. Montrer que si L est rationnel, alors la projection de L sur un alphabet est rationnelle.

Définition

Pour le reste du sujet, on fixe $k \geqslant 1$ et $\tilde{\Sigma} = (A_1, \ldots, A_k)$ un k-uplet d'alphabets finis, non nécessairement disjoints. On note $\Sigma = A_1 \cup \ldots \cup A_k$.

Soit $L \subseteq \Sigma^*$ un langage. On appelle **shuffle** de L (par rapport à $\tilde{\Sigma}$) le langage :

$$\mathsf{sh}(L) := \{u \in \Sigma^* | \forall i \in [\![1; k]\!], \exists u_i \in L, \mathsf{proj}(u, A_i) = \mathsf{proj}(u_i, A_i)\}$$

2. Soit $\tilde{\Sigma} = (\{a, b\}, \{b, c\})$ et $L = \{acb, bac\}$. Calculer $\mathsf{sh}(L)$.
3. Montrer que le shuffle d'un langage rationnel est rationnel.

Définition

Soit L un langage. On dit que L est un **shuffle rationnel** si $L = \mathsf{sh}(L)$ et L est rationnel. On dit que c'est un **semi-shuffle rationnel** s'il existe $m \geqslant 1$ et des shuffles rationnels $L_1, \ldots, L_m$ tels que $L = L_1 \cup \ldots \cup L_m$.

4. Soit $\tilde{\Sigma} = (\{a, b\}, \{b, c\})$ et $L = (a + b + c)^*(ac + ca)(a + b + c)^*$. Montrer que L n'est pas un shuffle rationnel.
5. Montrer que le langage précédent n'est pas un semi-shuffle rationnel.
6. Soit $\tilde{\Sigma} = (\{a\}, \{c\})$. Montrer que la classe des semi-shuffles rationnels (par rapport à $\tilde{\Sigma}$) est strictement plus grande que la classe des shuffles rationnels.
7. Montrer que la classe des shuffles rationnels (pour un $\tilde{\Sigma}$ bien choisi) n'est pas stable par complémentaire.
8. Montrer que la classe des semi-shuffles rationnels est stable par complémentaire.

Corrigé

1. On montre ce résultat par induction :
   - Si $L = \emptyset$ ou $L = a \in \Sigma$, alors le résultat est immédiat (la projection est soit $\emptyset$, soit a).
   - Supposons que la projection de L_1 et L_2 sur A soit rationnelle. Alors :
     * $\mathsf{proj}(L_1 L_2, A) = \mathsf{proj}(L_1, A)\mathsf{proj}(L_2, A) \in \mathrm{Rat}(\Sigma)$
     * $\mathsf{proj}(L_1 \cup L_2, A) = \mathsf{proj}(L_1, A) \cup \mathsf{proj}(L_2, A) \in \mathrm{Rat}(\Sigma)$
     * $\mathsf{proj}(L_1^*, A) = \mathsf{proj}(L_1, A)^* \in \mathrm{Rat}(\Sigma)$
2. $\mathsf{sh}(L) = \{abc, acb, bac, bca, cab, cba\}$.
3. Posons $L_i = \{u \in \Sigma^* | \exists v \in L, \mathsf{proj}(u, A_i) = \mathsf{proj}(v, A_i)\} = \{u \in \Sigma^* | \mathsf{proj}(u, A_i) \in \mathsf{proj}(L, A_i)\}$. Alors $\mathsf{sh}(L) = \bigcap_{i=1}^{k} L_i$.

   Remarquons alors que si L est un langage rationnel et A un alphabet, alors $M = \{u \in \Sigma^* | \mathsf{proj}(u, A) \in L\}$ est rationnel.

   En effet, soit $\mathcal{A} = (Q, A, \delta, q_0, F)$ un automate fini déterministe complet reconnaissant L. Posons $\mathcal{B} = (Q, \Sigma, \delta', q_0, F)$ où δ' est définie, pour $q \in Q$, par :
   - si $a \in A$, alors $\delta'(q, a) = \delta(q, a)$;
   - si $a \in \Sigma \setminus A$, alors $\delta'(q, a) = q$ (on rajoute des boucles sur chaque état pour lire les lettres en dehors de A).

Alors $\mathcal{B}$ reconnaît M.

Finalement, par la question 1 et par clôture par intersection, on en déduit que le shuffle d'un langage rationnel est rationnel.

4. $\mathsf{sh}(L) = (a+b+c)^*a(a+b+c)^* \cap (a+b+c)^*c(a+b+c)^* \neq L$.
5. Supposons que L est un semi-shuffle rationnel. Alors il existe L_1, ..., L_m des shuffle rationnels tels que $L = \cup_{i=1}^m L_i$. Posons alors, pour $k \in [\![0;m]\!]$: $u_k = b^k acb^{m-k}$. On remarque aisément que $u_k \in L$. Par le principe des tiroirs, on en déduit qu'il existe $0 \leqslant i < j \leqslant m$ et $\ell \in [\![0;m]\!]$ tels que $u_i, u_j \in L_\ell$. Posons alors $u = b^i ab^{j-i} cb^{m-j}$. On remarque que $\mathsf{proj}(u_i, \{a,b\}) = b^i ab^{m-i} = \mathsf{proj}(u, \{a,b\})$ et que $\mathsf{proj}(u_j, \{b,c\}) = b^j cb^{m-j} = \mathsf{proj}(u, \{b,c\})$. L_ℓ étant supposé un shuffle rationnel, cela implique que $u \in L_\ell \subseteq L$. Par l'absurde, on en déduit le résultat.
6. Soit $\tilde{\Sigma} = (\{a\}, \{c\})$. On pose $L_1 = a^*$ et $L_2 = c^*$. Il est clair que L_1 et L_2 sont des shuffles rationnels. De plus, si on pose $L = L_1 + L_2$, alors $\mathsf{sh}(L) = (a+c)^* \neq L$.
7. On pose $L = \{abc\}$ et $\tilde{\Sigma} = (\{a,b\}, \{b,c\})$. Alors on a $acb \in \overline{L}$ et $bac \in \overline{L}$. Si $\overline{L}$ était un shuffle rationnel, cela impliquerait que $abc \in \overline{L}$, ce qui est absurde.
8. On commence par des résultats faciles à montrer : la rationalité est donnée par la stabilité des langages rationnels par les opérations ensemblistes. De plus, l'union finie de semi-shuffles rationnels est clairement un semi-shuffle rationnel. L'intersection de deux shuffles rationnels est un shuffle rationnel. L'intersection de semi-shuffles rationnels est donc un semi-shuffle rationnel (par les deux propriétés précédentes et la distributivité).
   Finalement, montrons que le complémentaire d'un shuffle rationnel est un semi-shuffle rationnel, ce qui terminera la preuve :
   Soit L un shuffle rationnel. Alors on a vu que

$$L = \bigcap_{i=1}^{k} \{u \in \Sigma^* | \mathsf{proj}(u, A_i) \in \mathsf{proj}(L, A_i)\}$$

   On en déduit :

$$\overline{L} = \bigcup_{i=1}^{k} \{u \in \Sigma^* | \mathsf{proj}(u, A_i) \notin \mathsf{proj}(L, A_i)\}$$

   Montrons alors que $L_i = \{u \in \Sigma^* | \mathsf{proj}(u, A_i) \notin \mathsf{proj}(L, A_i)\}$ est un shuffle rationnel. On a déjà trivialement $L_i \subseteq \mathsf{sh}(L_i)$. Réciproquement, soit $u \notin L_i$. Alors $\exists v \in L, \mathsf{proj}(u, A_i) = \mathsf{proj}(v, A_i)$ et de plus, $\forall w \in L_i, \mathsf{proj}(u, A_i) \neq \mathsf{proj}(w, A_i)$. Cela montre que $u \notin \mathsf{sh}(L_i)$.