A1 – Séquences d'élimination

On considère un graphe non-orienté $G = (V, E)$ où V est un ensemble non-vide de sommets et E un ensemble de paires de sommets. On appelle *séquence* une suite finie non-vide σ, et sa *longueur* est notée $|\sigma| > 0$. On rappelle qu'un *chemin* dans G est une séquence $\pi = v_1 \dots v_{|\sigma|}$ d'éléments de V telle que $\{v_i, v_{i+1}\} \in E$ pour tout $1 \leq i < |\sigma|$.

On dit que deux séquences $u_1 \dots u_n$ et $v_1 \dots v_n$ de même longueur $n \in \mathbb{N}$ se *rencontrent* s'il existe $1 \leq i \leq n$ tel que $u_i = v_i$. Une *séquence d'élimination* σ pour G est une séquence $\sigma = s_1 \dots s_{|\sigma|}$ d'éléments de V telle que, pour tout chemin π dans G tel que $|\pi| = |\sigma|$, les séquences σ et π se rencontrent.

L'objet du problème est d'étudier quels graphes admettent une séquence d'élimination, et comment ceux-ci peuvent être identifiés par un algorithme.

Question 0. Construire une séquence d'élimination pour le graphe $L_3 = (\{1, 2, 3\}, \{\{1, 2\}, \{2, 3\}\})$ représenté comme suit : $1 \text{ —— } 2 \text{ —— } 3$.

Question 1. Un chemin $\pi = v_1 \dots v_n$ dans un graphe G est dit *simple* s'il n'existe pas $1 \leq i < j < n$ tels que $\{v_i, v_{i+1}\} = \{v_j, v_{j+1}\}$. Un *cycle* dans un graphe G est un chemin simple $\pi = v_1 \dots v_n$ de longueur $n > 1$ tel que $v_1 = v_n$.

Montrer que si G a un cycle alors G n'admet pas de séquence d'élimination.

Question 2. Pour tout $k \in \mathbb{N}^*$, on note L_k le graphe $(\{1, \dots, k\}, \{\{i, i+1\} \mid 1 \leq i < k\})$. Construire une séquence d'élimination pour L_k.
Indication : On distinguera le cas où k est pair et le cas où k est impair.

Question 3. On note 2^V l'ensemble des parties de V. Une *séquence d'élimination partielle* σ pour G et pour un sous-ensemble $X \in 2^V$ est une séquence $\sigma = s_1 \dots s_{|\sigma|}$ d'éléments de V telle que, pour tout chemin π dans G tel que $|\pi| = |\sigma|$, si le dernier élément de π est dans X, alors σ et π se rencontrent.

On note $\phi_E : 2^V \to 2^V$ la fonction définie par :

$$\phi_E(X) := V \setminus \{w \mid \exists w' \in V \setminus X, \{w, w'\} \in E\}.$$

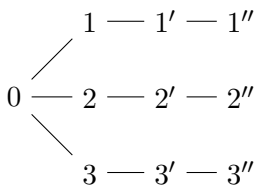
Montrer que, pour tout $X \in 2^V$, pour toute séquence d'élimination partielle σ pour G et X, pour tout $v \in V$, la séquence σ' obtenue en concaténant σ et v est une séquence d'élimination partielle pour G et $\phi_E(X) \cup \{v\}$.

Question 4. Déduire de la question précédente un algorithme pour décider, étant donné un graphe G, si ce graphe admet une séquence d'élimination, et si oui, la calculer. Discuter de l'efficacité de cet algorithme et de la longueur de la séquence obtenue.
Indication : Construire un graphe orienté sur 2^V dont les arêtes sont données par ϕ_E.

Suite des questions

Question 5. Pour tout $k \in \mathbb{N}^*$, on note S_k le graphe $(\{0\} \cup \{1, \dots, k\} \cup \{1', \dots, k'\}, \{\{0, i\} \mid 1 \leq i \leq k\} \cup \{\{i, i'\} \mid 1 \leq i \leq k\})$. Construire une séquence d'élimination pour S_k.

Question 6. Montrer que le graphe suivant n'admet pas de séquence d'élimination :



Question 7. En s'inspirant des questions précédentes, proposer une caractérisation des graphes qui admettent une séquence d'élimination. Justifier pourquoi elle est nécessaire et suffisante. Proposer un algorithme pour tester cette caractérisation, et comparer sa complexité à celle de l'algorithme obtenu en question 4. Étendre l'algorithme pour qu'il calcule également une séquence d'élimination pour les graphes qui en admettent une, et discuter de sa longueur maximale.

Corrigé

Dans ce corrigé, on considère les séquences comme des mots finis, c'est-à-dire que la juxtaposition de deux suites dénote leur concaténation, etc.

Question 0. On considère la séquence $\sigma = s_1 s_2$ définie par $s_1 = s_2 = 2$. Considérons un chemin arbitraire $\pi = v_1 v_2$. Soit $v_1 = 2$, auquel cas on a $v_1 = s_1$ et σ et π se rencontrent, soit $v_i \in \{1, 3\}$. Dans ce second cas, comme v_2 est adjacent à v_1, on a nécessairement $v_2 = 2$, de sorte qu'on a $v_2 = s_2$. Ainsi, σ et π se rencontrent. On en déduit que σ est bien une séquence d'élimination pour L_3.

Question 1. Soit $v_1 \dots v_n$ un cycle de G, avec $v_n = v_1$ et $n > 1$: noter qu'on ne peut avoir $n = 2$ (aucun sommet n'est adjacent à lui-même), ni $n = 3$ (le chemin ne pourrait être simple puisque l'on aurait $\{v_1, v_2\} = \{v_2, v_3\}$ puisque $v_1 = v_3$), donc nécessairement $n > 3$.

Observons que, pour tout $1 \leq i \leq n$, on sait que v_i a au moins deux sommets adjacents : en effet, il s'agit de v_{i-1} et v_{i+1} pour $1 < i < n$, et de v_{n-1} et v_2 pour v_1 et pour v_n (qui sont égaux) : noter qu'on a $2 \neq n - 1$ puisque $n > 3$. Ainsi, pour tout $1 \leq i \leq n$, ces deux sommets adjacents à v_i sont nécessairement distincts, car le chemin est simple.

Montrons par récurrence sur $i \in \mathbb{N}$, que, pour toute séquence σ de longueur i, on peut construire un chemin π_i de longueur i d'éléments du cycle $v_1 \dots v_n$ qui ne rencontre pas σ, de sorte que σ n'est pas une séquence d'élimination. Ceci suffit à conclure qu'il n'existe aucune séquence d'élimination pour G.

- **Cas de base** $i = 1$ **:** Soit $\sigma = s_1$ une séquence de longueur 1. On construit un chemin π_1 de longueur 1 en choisissant un sommet quelconque du cycle qui soit différent de s_1 ; c'est possible car $n > 1$ et que v_1 et v_2 sont adjacents, donc différents, ainsi le cycle contient au moins deux sommets distincts.
- **Récurrence :** Soit $i \in \mathbb{N}^*$, et supposons le résultat acquis pour i. Soit $\sigma = s_1 \dots s_{i+1}$ une séquence de longueur $i + 1$, et construisons un chemin π_{i+1} de longueur $i + 1$ qui ne rencontre pas σ. En utilisant l'hypothèse de récurrence, soit π_i un chemin de longueur i qui ne rencontre pas $s_1 \dots s_i$. Soit w son dernier élément : on sait que c'est un sommet du cycle, donc il y a deux sommets distincts du cycle qui sont adjacents à w. Choisissons un sommet w' qui soit différent de s_{i+1} parmi ces deux sommets, et construisons $\pi_{i+1} := \pi_i w'$. Il est clair que π_{i+1} ne rencontre pas σ :

en effet, pour tout $1 \leq j \leq i$, le j-ème élément de π_{i+1} est le même que celui de π_i, donc il est différent de s_j par hypothèse sur π_i, et pour $j = i+1$ le j-ème élément de π_{i+1} est différent de s_{i+1} par construction. Ceci conclut le cas de récurrence, donc on a prouvé le résultat demandé.

Question 2. On suppose d'abord que k est impair. On définit la séquence $\sigma = s_1 \dots s_{2k}$ par $s_i := i$ et $s_{i+k} := i$ pour tout $1 \leq i \leq k$. Montrons que σ est une séquence d'élimination. Considérons un chemin $\pi = v_1 \dots v_k v_{k+1} \dots v_{2k}$. On considérera la séquence définie par $\delta_i := v_i - s_i$, et on cherchera à montrer qu'il existe $1 \leq i \leq 2k$ tel que $\delta_i = 0$, ce qui permet de déduire que π et σ se rencontrent.

Pour tout $1 \leq i < 2k$, comme v_i et v_{i+1} sont adjacents, il sont de parité différente, donc il y a deux cas possibles : soit v_i est de la même parité que i pour tout $1 \leq i \leq 2k$, soit v_i est de la parité opposée à i pour tout $1 \leq i \leq 2k$.

Dans le premier cas, pour tout $1 \leq i \leq k$, la valeur δ_i est paire, car c'est la différence de deux valeurs de même parité. De plus, pour $1 \leq i < k$, comme on a $|s_{i+1} - s_i| = 1$ et $|v_{i+1} - v_i| = 1$ (car ce sont des paires de sommets adjacents), on a $|\delta_{i+1} - \delta_i| \leq 2$. On observe ensuite que $\delta_1 \geq 0$ mais que $\delta_k \leq 0$, d'où l'on déduit qu'il existe $1 \leq i \leq k$ tel que $\delta_i = 0$, ce qui conclut.

Dans le second cas, pour tout $k+1 \leq i \leq 2k$, la valeur s_i est de parité opposée à celle de i, car k est impair. On en déduit que, pour $k+1 \leq i \leq 2k$, la valeur δ_i est paire. On peut montrer comme dans le premier cas que $|\delta_{i+1} - \delta_i| \leq 2$ pour tout $k+1 \leq i < 2k$, et que $\delta_{k+1} \geq 0$ et $\delta_{2k} \leq 0$, ce qui permet de conclure comme précédemment.

Si k est pair, on définit $\sigma = s_1 \dots s_{2k+1}$ par $s_i = s_{i+k+1} = i$ pour $1 \leq i \leq k$ et $s_{k+1} = 1$. On distingue deux cas selon la parité des éléments du chemin, comme précédemment. Le premier cas est exactement comme dans le cas où k est impair. Le second cas fonctionne comme précédemment mais en considérant l'intervalle $k+2 \leq i \leq 2k+1$.

Question 3. Fixons le sous-ensemble $X \in 2^V$, la séquence d'élimination partielle σ pour X et G, et le sommet $v \in V$. Considérons la séquence $\sigma' := \sigma v$. Soit π' un chemin de G tel que $|\pi'| = |\sigma'|$, et écrivons π' comme la concaténation d'un chemin π de longueur $|\sigma|$ (donc non-vide) et du dernier sommet v' de π'. Montrons que, si $v' \in \phi_E(X) \cup \{v\}$, alors π' et σ' se rencontrent. Si $v' = v$, c'est immédiat en considérant la dernière position. Si $v' \in \phi_E(X)$, considérons le dernier sommet v'' de π : il s'agit d'un sommet adjacent à v'. Montrons par l'absurde qu'il appartient à X. En effet, si l'on supposait le contraire, on aurait $v'' \in V \setminus X$, de sorte que v' aurait un sommet adjacent dans $V \setminus X$: ceci signifie précisément que $v' \notin \phi_E(X)$, ce qui contredit notre hypothèse. On a donc $v'' \in X$. Or, comme σ est une séquence d'élimination partielle pour X et G, ceci implique que π et σ se rencontrent, donc que π' et σ' se rencontrent, ce qu'il fallait démontrer.

Question 4. Il est clair que, si G contient des sommets qui n'ont aucun sommet adjacent, alors retirer ces sommets ne change rien au fait que G admette une séquence d'élimination. On supposera donc par la suite que ces sommets ont été éliminés de G, ce qui peut se faire en temps linéaire. Si le graphe résultant est vide, le graphe original n'avait pas d'arêtes, donc admettait une séquence d'élimination de longueur 1 (prendre un sommet arbitraire). Ainsi, on suppose que G est toujours non-vide et que tout sommet de G a au moins un sommet adjacent.

On va renforcer légèrement le résultat de la question 3. Appelons *séquence d'élimination exacte* σ pour G et pour $X \in 2^V$ une séquence d'éléments de V telle que, pour tout $v \in V$, il existe un chemin de G de longueur $|\sigma|$ dont le dernier élément est v et qui ne rencontre pas σ si et seulement si on a $v \in X$. On montre que, pour tout $X \in 2^V$, pour toute séquence d'élimination exacte σ pour G et X, et pour tout $v \in V$, la séquence $\sigma' := \sigma v$ est une séquence d'élimination exacte pour $\phi_E(X) \cup \{v\}$. Vu ce qui a été montré à la question 3, il suffit de montrer une seule direction : pour tout $v' \in V \setminus (\phi_E(X) \cup \{v\})$, il existe un chemin π' de longueur $|\sigma'|$ qui ne rencontre pas σ et qui se termine par v'. Par définition de v', on sait que $v' \neq v$, et on sait que $v' \in V \setminus \phi_E(X)$. Ainsi, par définition de ϕ_E, il existe un sommet $v'' \in V \setminus X$ tel que $\{v', v''\} \in E$. Comme $v'' \in V \setminus X$, et comme σ est une séquence d'élimination exacte

pour X, il existe un chemin π de longueur $|\sigma|$ dont le dernier élément est v'' et qui ne rencontre pas σ. Considérons le chemin $\pi' := \pi v'$. Il s'agit d'un chemin, car $\{v', v''\} \in E$, il a la bonne longueur, se finit par le bon élément, et il ne rencontre pas σ' car π ne rencontre pas σ et $v' \neq v$.

On considère à présent le graphe orienté *étiqueté* $G' = (2^V, E')$ où chaque arête de E' porte une étiquette dans V, c'est-à-dire $E' \subseteq 2^V \times V \times 2^V$: on définit $E' = \{(X, v, \phi_E(X) \cup \{v\}) \mid X \in 2^V, v \in V\}$. Un *chemin d'arêtes* de G' est une séquence d'arêtes $\eta := e_1, \ldots, e_{|\eta|}$ de E' (où l'on écrira $e_i = (X_i, v_i, Y_i)$ pour tout $1 \leq i \leq |\eta|$) telle que, pour tout $1 \leq i < |\eta|$, on a $Y_i = X_{i+1}$. L'*étiquette* de η est la séquence $v_1 \ldots v_{|\eta|}$, son *départ* est X_1, et son *arrivée* est $Y_{|\eta|}$.

Montrons par récurrence sur sa longueur $i \in \mathbb{N}^*$ que, pour tout chemin d'arêtes $\eta = e_1 \ldots e_i$ de G' dont le départ est $\emptyset$, son étiquette est une séquence d'élimination partielle pour G et pour l'arrivée de η :

Cas de base $i = 1$ **:** Pour un chemin d'arêtes e_1, où $e_1 = (\emptyset, v, Y)$, il suffit de montrer que la séquence v est une séquence d'élimination partielle pour Y, c'est-à-dire, par définition de G', pour $\phi_E(\emptyset) \cup \{v\}$. Notre prétraitement assure que tous les sommets de G ont un sommet adjacent, ce qui assure que $\phi_E(\emptyset) = \emptyset$. Or il est clair que la séquence v est une séquence d'élimination partielle pour $\{v\}$, ce qui conclut.

Récurrence : Soit $i \in \mathbb{N}^*$, et supposons le résultat acquis pour i. Considérons un chemin d'arêtes $\eta = e_1 \ldots e_{i+1}$ de G' dont le départ est $\emptyset$, et montrons le résultat demandé. Par hypothèse de récurrence, comme $\eta' = e_1 \ldots e_i$ est un chemin d'arêtes dont le départ est $\emptyset$, son étiquette l est une séquence d'élimination partielle pour G et pour l'arrivée Y de η'. La dernière arête de η est par construction de la forme $(Y, v, \phi_E(Y) \cup \{v\})$. On utilise à présent la question 4 pour conclure que lv est une séquence d'élimination partielle pour G et $\phi_E(Y) \cup \{v\}$. On a donc prouvé le cas de récurrence, ce qui conclut la preuve.

Ainsi, si G' a un chemin d'arêtes η dont le départ est $\emptyset$ et dont l'arrivée est V, alors l'étiquette de η est une séquence d'élimination.

Réciproquement, supposons que G admet une séquence d'élimination $\sigma = s_1 \ldots s_n$, et montrons que G' a un chemin d'arêtes dont le départ est $\emptyset$ et dont l'étiquette est une séquence d'élimination. Observons que la construction de G' garantit qu'il y a un unique chemin d'arêtes η d'étiquette σ et de départ $\emptyset$. Montrons que son arrivée est nécessairement V, en montrant par récurrence sur $i \in \mathbb{N}^*$ que $s_1 \ldots s_i$ est une séquence d'élimination exacte pour Y_i. Le cas de base est immédiat : par construction $Y_1 = \{s_1\}$ et s_1 est une séquence d'élimination exacte pour $\{s_1\}$. Pour le cas de récurrence, on utilise la version plus forte de la question 3 que l'on a prouvé plus haut. Ainsi, par récurrence, on sait que σ est une séquence d'élimination exacte pour l'arrivée Y_n de η. Mais comme σ est une séquence d'élimination, on a $Y_n = V$. Ceci établit l'affirmation réciproque.

Ainsi, s'il existe un chemin d'arêtes dans G' dont le départ est $\emptyset$ et dont l'arrivée est V, alors le graphe G admet une séquence d'élimination, et la séquence d'élimination est alors donnée par l'étiquette de ce chemin ; et réciproquement, si G admet une séquence d'élimination, alors il existe un tel chemin. Il est alors clair que décider l'existence d'un tel chemin revient à tester si le sommet V est accessible à partir du sommet $\emptyset$ dans G', et que l'étiquette peut être calculée à partir du chemin. Le test d'accessibilité s'effectue en temps linéaire en G' une fois celui-ci construit, c'est-à-dire en temps exponentiel en G.

On a ainsi conçu un algorithme qui, étant donné un graphe G, décide en temps exponentiel en G si G admet une séquence d'élimination, et si oui, la calcule ; la séquence résultante est de taille au plus $2^{|V|}$.

Question 5. On appelle σ' la séquence définie par $(1, 0, 2, 0, \ldots, k)$ et on considère la séquence σ définie en répétant deux fois σ. On note $s_1 \ldots s_{4k-2}$ ses éléments.

On appelle sommets *pairs* de G le sommet 0 et les sommets de la forme i' pour $1 \leq i \leq k$; et *impairs* les autres sommets.

Considérons un chemin $\pi = v_1 \ldots v_{4k-2}$, et montrons que π et σ se rencontrent. Comme à la question 2, il y a deux possibilités ; soit v_i est de la même parité que i pour tout $1 \leq i \leq 4k-2$, soit il est toujours de la parité inverse.

Dans le premier cas, pour chaque valeur paire de i, on a que v_i est pair ; mais comme s_i est égal à 0 lorsque i est pair, σ et π se rencontrent à moins que v_i pour i pair soit toujours différent de 0. Mais dans ce cas, comme v_i pour i impair est pair, donc différent de 0, on déduit que π est un chemin qui ne passe pas par 0. Ainsi, π est forcément de la forme $j, j', j, j', \ldots, j, j'$ pour un certain $1 \leq j \leq k$. Mais ainsi $v_{2j-1} = j$, or $s_{2j-1} = j$ par construction, donc σ et π se rencontrent.

Dans le second cas, pour chaque valeur impaire de i, on a que v_i est pair ; mais, comme s_i est égal à 0 lorsque i est impair et supérieur ou égal à $2k+1$, on sait que σ et π se rencontrent à moins que v_i pour tout i impair, $i \geq 2k+1$, soit différent de 0. On en déduit comme précédemment que tous les éléments de π de coordonnée $\geq 2k+1$ ne passent pas par 0, et on en déduit comme avant que σ et π se rencontrent.

Question 6. Soit $\sigma = s_1 \ldots s_n$ une séquence d'éléments de V. Montrons qu'il ne s'agit pas d'une séquence d'élimination. On suppose sans perte de généralité que s_1 est différent de 0 ; en effet, si le graphe admettait une séquence d'élimination, toute séquence admettant cette séquence comme suffixe serait également une séquence d'élimination, donc on pourrait la préfixer par un sommet différent de 0 et obtenir une séquence d'élimination.

On définit un chemin $\pi = v_1, \ldots, v_n$ dans G de longueur n de la façon suivante : $v_1 := 0$, et, pour tout $1 \leq i < n$:

- Si $v_i = j''$, alors $v_{i+1} := j'$ (c'est le seul choix possible) ;
- Si $v_i = j'$, alors $v_{i+1} := j$, à moins que $\pi_{i+1} = j$, auquel cas $v_{i+1} := j''$;
- Si $v_i = j$, alors $v_{i+1} := 0$, à moins que $\pi_{i+1} = 0$, auquel cas $\pi_{i+1} := j'$;
- Si $v_i = 0$, alors $v_{i+1} := j$ où j est choisi de la façon suivante :
  - On exclut π_{i+1} ; ainsi $j \neq \pi_{i+1}$;
  - Pour chaque $k \in \{1,2,3\} \setminus \{\pi_{i+1}\}$, soit $i < l_k < n$ la plus petite position après i dont la parité est différente de i et qui satisfait $\pi_{l_k} = k$ et $\pi_{l_k+1} = k'$ (noter que l_k n'est pas nécessairement défini suivant la valeur de k). Si un des deux l_k n'est pas défini, on exclut l'autre, de sorte que l_k ne soit pas défini. Sinon, si les deux l_k sont définis, on exclut celui pour lequel l_k est minimal. (Noter que la définition garantit clairement que $l_k \neq l_{k'}$ pour $k \neq k'$, donc il ne peut y avoir d'égalités.)

  Comme 0 a trois sommets adjacents, il est clair que l'on a une manière de choisir v_{i+1} suivant ces règles, car chaque critère exclut au plus un sommet adjacent.

Il est immédiat que π ainsi défini est effectivement un chemin. Montrons que σ n'est pas une séquence d'élimination en montrant qu'elle ne rencontre pas π. Il est clair par définition que le premier élément de σ et de π sont différents. Supposons par l'absurde qu'il existe une position $0 \leq i_0 < n$ telle que $s_{i_0+1} = v_{i_0+1}$, et prenons la plus petite telle position. Par définition des v_i pour $1 < i \leq n$, la seule possibilité est que $s_{i_0+1} = v_{i_0+1} = j'$ pour un certain $j \in \{1,2,3\}$, et que $v_{i_0} = j''$. En effet, $v_{i_0} = j''$ est le seul cas où on n'a pas pu choisir v_{i_0+1} de sorte à garantir qu'il est différent de s_{i_0+1}. Ainsi, en $i := i_0 - 1$, on avait $v_{i_0-1} = j'$ et on a choisi $v_{(i_0-1)+1} := j''$ donc on avait $s_{i_0} = j$. Noter qu'en particulier $v_{i_0-1} \neq 0$.

Considérons à présent la plus grande position $1 \leq i'_0 < i_0 - 1$ telle que $v_{i'_0} = 0$: il existe forcément une telle position puisque $v_1 = 0$. Il est alors clair suivant la structure de G que $v_{i'_0+1} = j$, puisque que sinon le sous-chemin de π allant de la position $i'_0 + 1$ à la position i_0 devrait repasser par 0 afin d'atteindre j'' (à la position i_0), ce qui contredirait la maximalité de i'_0. Ainsi,comme la suite du chemin jusqu'à i_0 ne repasse pas par 0, il est facile d'observer que pour toute position i telle que $i'_0 < i \leq i_0$, si i est de même parité que i'_0, on a $v_i = j'$, et si i est de parité différente de i'_0, on a $v_i \in \{j, j''\}$. On en déduit, comme $v_{i_0} = j''$, que i_0 et i'_0 ont des parités différentes.

Pour $i := i'_0$ dans la définition de π, nous avions par définition $l_j \leq i_0$, vu que i_0 est une position qui satisfaisait les conditions demandées. Ainsi, si on a choisi de définir $v_{i'_0+1} := j$, ceci doit signifier qu'il existe $k \in \{1,2,3\}$ tel que $k \neq j$ et $k \neq s_{i'_0} + 1$ tel que l_k est défini et $l_k \leq i_0 - 1$: en fait, puisque nous connaissons s_{i_0} et s_{i_0-1}, nous savons que $l_k < l_j - 2$. Ainsi, pour une position $i < l_k < l_j$ de parité

différente de i'_0, nous avions $s_{l_k} = k$ et $s_{l_k+1} = k'$. Mais on sait par parité de $l_k - 1$ que $s_{l_k-1} = j'$, donc on a défini $s_{l_k} := j$ et $s_{l_k+1} := 0$ par nos règles ci-dessus. Ainsi, on a $s_{l_k+1} = 0$, alors que $l_k + 1 > i'_0$ et $l_k + 1 \leq l_j - 2 < i_0 - 1$, ce qui contredit la minimalité de i'_0. On a obtenu une contradiction, donc la position i_0 n'existe pas, et π permet donc de conclure que σ n'est pas une séquence d'élimination.

[La preuve de cette question est inspirée de la preuve du Théorème 1 de [BW12], en la modifiant légèrement.]

Question 7. On dit qu'un graphe non-orienté G est un *homard* (en anglais, *lobster graph*) si et seulement s'il est connexe, n'a pas de cycle, et s'il existe un chemin simple π dans G tel que, pour tout sommet v de G, la distance de v à un élément de π est au plus de 2. Il est clair que, si G est un homard, n'importe quel chemin de longueur maximale peut être utilisé comme témoin π.

La caractérisation proposée est la suivante (*) : *un graphe admet une séquence d'élimination si et seulement s'il s'agit d'une union de homards.*

Observons dans un premier temps que, si une des composantes connexes G' d'un graphe G n'admet pas de séquence d'élimination, alors G n'en admet pas non plus : en effet, une séquence d'élimination pour G est en particulier une séquence pour G'. Réciproquement, si toutes les composantes connexes de G admettent une séquence d'élimination, alors c'est le cas de G, et une séquence d'élimination s'obtient simplement en concaténant une séquence d'élimination pour chaque composante : ceci utilise le fait qu'une séquence dont un sous-mot est une séquence d'élimination est elle-même une séquence d'élimination. Ainsi, il suffit de montrer l'énoncé suivant : *un graphe connexe admet une séquence d'élimination si et seulement s'il s'agit d'un homard.*

Par la question 1, si G a un cycle, alors il n'admet pas de séquence d'élimination, donc on pourra supposer sans perte de généralité que G est un graphe connexe acyclique, c'est-à-dire un arbre non orienté. Il est facile de constater que G est un homard si et seulement si G ne contient pas le motif de la question 6 (formellement : aucune restriction de G n'est isomorphe à ce graphe). Ceci justifie que, si G n'est pas un homard, alors il contient le motif de la question 6, et n'admet pas de séquence d'élimination (il suffit de se restreindre à ce motif et d'ignorer le reste du graphe). Ainsi, il suffit de montrer l'autre implication : si G est un homard alors il admet une séquence d'élimination.

Considérons une bipartition de G, c'est-à-dire une fonction $\chi : V \to \{0, 1\}$, telle que pour tous sommets v et v', si v et v' sont adjacents alors $\chi(v) \neq \chi(v')$. Comme G est sans cycle, il est clair qu'une telle fonction χ existe. Soit $V_0 \sqcup V_1$ la partition de V définie par $V_b := \chi^{-1}(b)$ pour $b \in \{0, 1\}$. Nous allons construire une séquence d'élimination partielle σ' pour V_0. Une fois ceci fait, nous construirons une séquence d'élimination σ de la façon suivante : si $|\sigma'|$ est impair, on prendra $\sigma := \sigma\sigma$, et si $|\sigma'|$ est pair on prendra $\sigma := \sigma w \sigma$ où w est un sommet arbitraire de G. On peut aisément montrer que, si σ' est une séquence d'élimination partielle pour V_0, alors σ est une séquence d'élimination : en effet, pour tout chemin $\pi = v_1 \dots v_n$, pour tous $1 \leq i, j \leq n$, on a $\chi(v_i) = \chi(v_j)$ si et seulement si i et j sont de même parité. Faisons à présent une distinction de cas sur la parité de $|\sigma'|$. En premier lieu, si $|\sigma'|$ est impair, comme $|\sigma'|$ et $2\,|\sigma'|$ sont de parité différente, l'un de $v_{|\sigma'|}$ et de $v_{2|\sigma'|}$ est dans V_0, donc, si l'on considère l'un des deux chemins $v_1, \dots, v_{|\sigma'|}$ et $v_{|\sigma'|+1}, \dots, v_{2|\sigma|}$, l'un des deux doit rencontrer la copie correspondante de la séquence d'élimination partielle σ' pour V_0, ce qui permet de conclure. En second lieu, si $|\sigma'|$ est pair, comme $|\sigma'|$ et $2\,|\sigma'| + 1$ sont de parité différente, l'un de $v_{|\sigma'|}$ et de $v_{2|\sigma'|+1}$ est dans V_0, donc, si l'on considère l'un des deux chemins $v_1, \dots, v_{|\sigma'|}$ et $v_{|\sigma'|+2}, \dots, v_{2|\sigma|+1}$, l'un des deux doit rencontrer la copie correspondante de σ' et on conclut de la même manière. Ainsi, il suffit de construire une séquence d'élimination partielle σ' pour V_0.

Soit $\pi' = w_1 \dots w_m$ un chemin simple de longueur maximale dans G qui permet de constater que G est un homard.. Quitte à échanger V_0 et V_1, on supposera sans perte de généralité que $w_0 \in V_0$. Pour tout $1 \leq i \leq m$, notons $m_i \in \mathbb{N}$ le nombre de sommets adjacents à w_i qui ne sont pas dans π', et nommons ces sommets adjacents $w_i^1, \dots, w_i^{m_i}$. Pour $1 \leq i \leq m$, notons σ_i la séquence $(w_i, w_i^1, w_i, w_i^2, \dots, w_i, w_i^{m_i}, w_i)$, comme à la question 5. En particulier, si $m_i = 0$ alors $\sigma_i = (w_i)$. Montrons que $\sigma' = \sigma_1 \dots \sigma_m$ est une séquence d'élimination partielle pour V_0. Écrivons $\sigma' = z_1 \dots z_q$. Par symétrie du problème, quitte à

inverser σ', π', et σ, il suffit de montrer que, pour tout chemin $\pi = v_1 \dots v_q$ dont le *premier* élément est dans V_0, nécessairement π et σ' se rencontrent. Il n'y a rien à montrer si $v_1 = w_1$ ou $v_q = w_m$, donc on suppose que ce n'est pas le cas.

Notons ϕ la fonction définie sur V par $\phi(w_i) := w_i$ pour tout $1 \leq i \leq m$, $\phi(w_i^j) := w_i$ pour tout $1 \leq i \leq m$ et $1 \leq j \leq m_i$, et $\phi(w) := w_i$ pour toute autre sommet w de G (nécessairement une feuille) dont l'unique sommet adjacent est de la forme w_i^j pour un certain $1 \leq j \leq m_i$. On raisonne alors sur l'évolution des différences $\delta_i := \phi(z_i) - \phi(v_i)$ pour $1 \leq i \leq q$ comme à la question 2 : on a $\delta_1 \leq 0$, $\delta_q \geq 0$, on a $|\delta_{i+1} - \delta_i| \leq 2$ pour tout $1 \leq i < q$ et par parité il ne peut y avoir de croisement, c'est-à-dire qu'on ne peut avoir $\delta_i = -1$ et $\delta_{i+1} = 1$. Ainsi, il y a une position $1 \leq i \leq q$ où $\phi(z_i) = \phi(v_i)$. On considère alors le σ_j pour $1 \leq j \leq m$ où se trouve cette position (noter que $\phi(z_i) = j$). On raisonne alors comme à la question 5 en utilisant le fait que σ_j est une séquence d'élimination pour le sous-graphe de G formé par les sommets dont l'image par ϕ est j.

On utilise la caractérisation (*) pour déterminer en temps linéaire si un graphe d'entrée G admet une séquence d'élimination, et si oui, calculer en temps linéaire une telle séquence. Ceci prouve également que tout graphe qui admet une séquence d'élimination en admet une de longueur linéaire. On calcule d'abord en temps linéaire les composantes connexes de G, que l'on traite indépendamment en temps linéaire en chaque composante.

Pour le graphe de chaque composante, on applique deux fois une opération d'*élimination des feuilles*, qui procède en temps linéaire : on marque tous les sommets de degré 1, puis on les élimine ainsi que leur unique arête incidente. On teste ensuite en temps linéaire si le graphe obtenu est une union de chemins, c'est-à-dire qu'on teste si chaque sommet a un degré de 0, 1, ou 2, et s'il n'y a pas de cycle, c'est-à-dire qu'une exploration en largeur depuis les sommets de degré 0 et 1 atteint tous les sommets du graphe. Il est facile d'observer que, si cette procédure échoue, alors le graphe initial n'était pas un homard ; et qu'à l'inverse, si elle réussit, alors le chemin restant permet d'obtenir facilement un chemin témoin du fait la composante composante connexe est un homard.

À partir de ce chemin témoin, on utilise la construction précédemment décrite pour calculer une séquence d'élimination de longueur linéaire en la composante connexe, en temps linéaire en celle-ci, ce qui permet de conclure.

[La preuve de cette question est inspirée de [BW12].]

[Pour des questions ouvertes qui généralisent le sujet de ce problème, on peut voir par exemple [Ama17].]

Références

[Ama17] Antoine Amarilli. The monk problem, 2017. `https://a3nm.net/work/research/questions/#the-monk-problem`.

[BW12] John R. Britnell and Mark Wildon. Finding a princess in a palace : A pursuit-evasion problem, 2012. `https://arxiv.org/abs/1204.5490`.

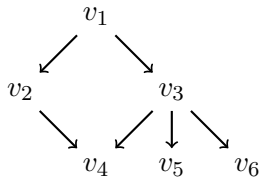
A2 – Étiquetage d'accessibilité dans les graphes

On considère un graphe orienté $G = (V, E)$ où V est un ensemble non-vide de sommets et $E \subseteq V \times V$ est un ensemble de couples de sommets. On rappelle qu'un *chemin* dans G de longueur $n \in \mathbb{N}$ est une suite finie $w_1 \dots w_n$ d'éléments de V telle que, pour tout $1 \leq i < n$, on a $(w_i, w_{i+1}) \in E$. Pour $u, v \in V$, on écrit $u \rightsquigarrow v$ quand il existe $n \in \mathbb{N}$ et un chemin $w_1 \dots w_n$ dans G tel que $u = w_1$ et $w_n = v$.

Un *étiquetage* de G est une fonction $f : V \to \mathbb{N}^2$. Étant donné une paire d'éléments $t_1 = (p_1, q_2)$ et $t_2 = (p_2, q_2)$ de $\mathbb{N}^2$, on écrit $t_1 \leq t_2$ pour indiquer que $p_1 \leq p_2$ et $q_1 \leq q_2$. Un étiquetage f est appelé *étiquetage d'accessibilité* s'il satisfait la propriété suivante : pour tous sommets $u \neq v$ de V, on a $u \rightsquigarrow v$ si et seulement si $f(u) \leq f(v)$.

L'objet du problème est d'étudier quels graphes admettent un étiquetage d'accessibilité, et comment ces graphes peuvent être identifiés et étiquetés par un algorithme.

Question 0. Construire un étiquetage d'accessibilité pour le graphe suivant :



Question 1. Rappeler la définition des composantes connexes et des composantes fortement connexes (CFC) du graphe orienté G, et la reformuler en utilisant $\rightsquigarrow$.

Question 2. On définit 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 comme suit :

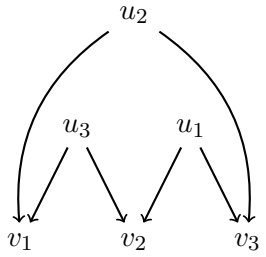
$$E_C := \{(K_1, K_2) \in V_C \times V_C \mid (K_1 \neq K_2) \wedge (\exists v_1 \in K_1, v_2 \in K_2 \text{ tels que } v_1 \rightsquigarrow v_2)\}$$

Expliquer brièvement pourquoi il n'existe pas de *cycle* de G_C, c'est-à-dire qu'il n'y a pas de chemin $w_1 \dots w_n$ de longueur $n \geq 3$ dans G_C tel que $w_1 = w_n$. On dit que G_C est *acyclique*.

Montrer que G admet un étiquetage d'accessibilité si et seulement si G_C admet un étiquetage d'accessibilité.

Question 3. Montrer que G admet un étiquetage d'accessibilité si et seulement si chacune de ses composantes connexes admet un étiquetage d'accessibilité.

Question 4. Montrer que le graphe suivant n'admet pas d'étiquetage d'accessibilité :



Suite des questions

Question 5. 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.

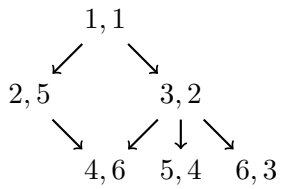
Montrer que, si G a un conjugué alors G admet un étiquetage d'accessibilité, et expliquer comment le calculer à partir de G'.

Indication : On pourra montrer dans un premier temps que le graphe orienté $(V, E \cup E')$ est acyclique.

Question 6. Montrer que, réciproquement, si G est acyclique et admet un étiquetage d'accessibilité, alors G a un conjugué.

Corrigé

Question 0. Une solution possible est :



Question 1. C'est une question de cours. Les composantes connexes sont les classes d'équivalence de la clôture symétrique, réflexive et transitive de la relation $\rightsquigarrow$ sur V. Les CFC sont les classes d'équivalence pour la relation d'équivalence sur V définie par $(u = v) \vee ((u \rightsquigarrow v) \wedge (v \rightsquigarrow u))$.

Question 2. Le graphe G_C est acyclique. En effet, procédons par l'absurde, et supposons qu'il existe un cycle $K_1 \dots K_n$ dans G_C. On en déduit qu'il existe une séquence $(v_1, v'_2) \dots (v_{n-1}, v'_n)$ de couples de sommets de G tels qu'on ait $(v_i, v'_{i+1}) \in K_i \times K_{i+1}$ pour tout $1 \leq i < n$, et que, pour tout $1 \leq i < n$, on ait $v_i \rightsquigarrow v'_{i+1}$. Par ailleurs, pour $1 \leq i < n$, comme v_{i+1} et v'_{i+1} appartiennent tous deux à la même CFC K_{i+1}, on a soit $v_{i+1} = v'_{i+1}$ soit $v_{i+1} \rightsquigarrow v'_{i+1}$. Enfin, comme $K_1 = K_n$ par définition d'un cycle, v'_n et v_1 appartiennent tous deux à K_1, donc on a soit $v'_n = v_1$, soit $v'_n \rightsquigarrow v_1$. En concaténant ces chemins et ces égalités, on obtient un cycle de G qui passe par plusieurs CFC, ce qui contredit la maximalité des CFC par lesquelles passe le cycle.

On montre maintenant que G admet un étiquetage d'accessibilité si et seulement si G_C en admet un. Pour la première implication, étant donné un étiquetage d'accessibilité f de G, on choisit une fonction $\phi : V_C \to V$ telle que $\phi(K)$ pour chaque CFC K est un sommet quelconque de K, et on définit un étiquetage d'accessibilité f_C de G_C par $f_C(K) := f(\phi(K))$. On observe facilement que, pour toutes CFC $K \neq K'$, on a $K \rightsquigarrow K'$ dans G_C si et seulement si, pour *tous* les sommets $v \in K$ et $v' \in K'$, on a $v \rightsquigarrow v'$ dans G. Ainsi, s'il existe une paire $K \neq K'$ de V_C où $f_C(K) \leq f_C(K')$ mais $K \not\rightsquigarrow K'$, on en déduit que $f(\phi(K)) \leq f(\phi(K'))$ alors que $\phi(K) \not\rightsquigarrow \phi(K')$. Le raisonnement si $f_C(K) \not\leq f_C(K')$ mais $K \rightsquigarrow K'$ est analogue.

Réciproquement, étant donné un étiquetage d'accessibilité f_C de G_C, on définit un étiquetage d'accessibilité f de G par $f(v) := f_C(K_v)$, où K_v est la CFC à laquelle appartient V. Vérifions que f est effectivement un étiquetage d'accessibilité. Pour deux sommets $u \neq v$ de G appartenant à deux CFC différentes, le raisonnement est le même que ci-dessus. Pour deux sommets $u \neq v$ de G appartenant à la même CFC K, on a $f(u) = f(v)$, de sorte que $f(u) \leq f(v)$ et $f(v) \leq f(u)$, et on a bien $u \rightsquigarrow v$ et $v \rightsquigarrow u$ par définition de la CFC K.

Question 3. Il est immédiat que, si $G = (V, E)$ admet un étiquetage d'accessibilité f, alors c'est le cas de toutes ses composantes connexes : il suffit de considérer la restriction de f à chaque composante. Montrons l'affirmation réciproque.

On procède par récurrence avec prédécesseurs sur le nombre de composantes connexes. S'il y a une seule composante connexe, l'affirmation est triviale. Pour l'étape de récurrence, supposons le résultat acquis pour tout graphe orienté acyclique avec $i \in \mathbb{N}$ composantes connexes ou moins, et supposons que G a $i+1$ composantes connexes. Partitionnons G en deux sous-graphes G_1 et G_2 qui sont chacun une union de composantes connexes de G : formellement, on partitionne V en V_1 et V_2 de sorte qu'il n'y ait pas de sommets de V_1 adjacents à des sommets de V_2, et on note $E_b := E \cap (V_b)^2$ pour $b \in \{1, 2\}$, et $G_b = (V_b, E_b)$. Il est clair que G_1 et G_2 ont au plus i composantes connexes, donc, par hypothèse de récurrence, G_1 et G_2 admettent chacun un étiquetage d'accessibilité f_1 et f_2 : on note par commodité $f_b(v_b) = (p_b(v), q_b(v))$ pour tout $b \in \{1, 2\}$ et $v_b \in V_b$. On montre comment construire un étiquetage d'accessibilité f de G.

Choisissons un entier $M \in \mathbb{N}$ strictement plus grand que tous les entiers qui apparaissent dans l'image de f_1 et de f_2. Définissons la fonction $f'_1 : V_1 \to \mathbb{N}^2$ par $f'_1(v_1) := (p_1(v_1) + M, q_1(v_1))$ pour tout $v_1 \in V_1$, et la fonction $f'_2 : V_2 \to \mathbb{N}^2$ par $f'_2(v_2) := (p_2(v_2), q_2(v_2) + M)$ pour tout $v_2 \in V_2$. Il est facile de voir que f'_1 et f'_2 sont toujours des étiquetages d'accessibilité de V_1 et V_2 respectivement, puisque, pour tout $b \in \{1, 2\}$, pour tout $u_b, v_b \in V_b$, on a $f_b(u_b) \leq f_b(v_b)$ si et seulement si $f'_b(u_b) \leq f'_b(v_b)$. Définissons à présent l'étiquetage f de G suivant f'_1 et f'_2 : la fonction f associe au sommet v la valeur $f'_1(v)$ si $v \in V_1$ et $f'_2(v)$ si $v \in V_2$. On note encore $f(v) = (p(v), q(v))$ pour tout $v \in V$.

Montrons que f est bien un étiquetage d'accessibilité de G. Soit $u \neq v$ deux sommets de G. S'ils sont tous deux dans G_1 on a effectivement $f(u) \leq f(v)$ si et seulement si $u \rightsquigarrow v$, parce que cette dernière affirmation est vraie dans G si et seulement si elle est vraie dans G_1, ce qui est le cas si et seulement si $f'_1(u) \leq f'_1(v)$ (car f'_1 est un étiquetage d'accessibilité de G_1), or $f(u) = f'_1(u)$ et $f(v) = f'_1(v)$. Le même raisonnement s'applique si u et v sont tous deux dans G_2. Si on a $u \in V_1$ et $v \in V_2$, on a $p(u) \geq M$ par définition de f'_1 et $p(v) < M$ par définition de M, alors que $q(u) < M$ et $q(v) \geq M$ pour les mêmes raisons, ainsi on a bien $f(u) \not\leq f(v)$ et $u \not\rightsquigarrow v$ par définition de G_1 et G_2. Le cas où $u \in V_2$ et $v \in V_1$ est symétrique. On a donc bien défini un étiquetage d'accessibilité f pour G, ce qui conclut l'étape de récurrence, et termine la preuve.

Question 4. On dit que deux sommets $u \neq v$ sont *incomparables* si on a $u \not\rightsquigarrow v$ et $v \not\rightsquigarrow u$: c'est-à-dire qu'ils ne sont pas comparables au sens de la terminologie introduite à la question 5. Notons que u_1 et v_1, u_2 et v_2, u_3 et v_3, sont incomparables.

Procédons par l'absurde, et supposons que G ait un étiquetage d'accessibilité f. On notera $f(v) = (p(v), q(v))$ pour tout $v \in V$. Pour deux sommets incomparables u et v, on sait qu'on a $f(u) \not\leq f(v)$ et $f(v) \not\leq f(u)$. Ainsi, il est clair qu'on a forcément $p(u) \neq p(v)$ et $q(u) \neq q(v)$, et l'une des deux situations suivantes doit se produire :

— $p(u) < p(v)$ et $q(u) > q(v)$, noté $u <' v$
— $p(u) > p(v)$ et $q(u) < q(v)$, noté $v <' u$

En considérant nos trois paires de sommets incomparables, on peut donc distinguer deux cas possibles : soit $u_i <' v_i$ pour deux valeurs différentes de $i \in \{1, 2, 3\}$, soit $v_i <' u_i$ pour deux valeurs différentes de $i \in \{1, 2, 3\}$.

Dans le premier cas, soient $i \neq j$ deux valeurs de $\{1, 2, 3\}$ telles que $u_i <' v_i$ et $u_i <' v_j$, et donc $q(u_i) > q(v_i)$ et $q(u_j) > q(v_j)$. Comme v_i et v_j sont également incomparables, soit $v_i <' v_j$, soit $v_j <' v_i$. Dans le premier sous-cas, on a donc $q(v_i) > q(v_j)$, et par transitivité on a $q(u_i) > q(v_j)$, mais ceci contredit le fait que $f(u_i) \leq f(v_j)$, puisque $u_i \rightsquigarrow v_j$. Dans le second sous-cas, on sait que $q(v_j) > q(v_i)$ et on conclut de même. La preuve du second cas est exactement analogue au premier cas, en inversant les rôles de p et de q. Ainsi, dans tous les cas, on aboutit à une contradiction, donc on déduit qu'il ne peut exister d'étiquetage d'accessibilité pour G.

Question 5. Montrons dans un premier temps le résultat proposé dans l'indication. Supposons par l'absurde que le graphe $(V, E \cup E')$ ait un cycle. Comme G et G' sont acycliques, ce cycle doit passer à la fois par des arêtes de G et par des arêtes de G'. En considérant le chemin franchi par le cycle dans G et dans G', on peut écrire le cycle comme $v_1 \dots v_n$ où $n \geq 2$ est impair, et, pour tout $1 \leq i < n$, si i est impair alors $v_i \rightsquigarrow v_{i+1}$ dans G, et si i est pair alors $v_i \rightsquigarrow v_{i+1}$ dans G'. Rappelons que ces cas sont mutuellement exclusifs par définition du conjugué. On choisit le cycle de sorte que le nombre d'alternances n soit minimal, et on cherche à aboutir à une contradiction.

On montre d'abord qu'on ne peut avoir $n = 3$. En effet, dans ce cas, v_1 et v_2 sont comparables dans G et v_2 et $v_3 = v_1$ sont comparables dans G', donc ceci contrevient à la définition du conjugué. On a donc $n \geq 5$.

Intéressons-nous à présent à v_1 et v_3. On doit avoir $v_1 \neq v_3$, car sinon, on aurait que v_1 et v_2 sont comparables dans G et que v_2 et $v_3 = v_1$ sont comparables dans G', ce qui contrevient à la définition du conjugué. Ainsi, $v_1 \neq v_3$, donc, par définition du conjugué, il y a deux cas : soit v_1 et v_3 sont comparables dans G, soit ils le sont dans G'. Dans le premier cas, s'ils sont comparables dans G, on ne peut avoir $v_3 \rightsquigarrow v_1$ dans G, parce que comme $v_1 \rightsquigarrow v_2$ dans G, on aurait alors $v_3 \rightsquigarrow v_2$ dans G par transitivité, or $v_2 \rightsquigarrow v_3$ dans G', ce qui contredirait la définition du conjugué. Ainsi, on a forcément $v_1 \rightsquigarrow v_3$ dans G, mais comme $v_3 \rightsquigarrow v_4$ dans G, on en déduit que $v_1 \rightsquigarrow v_4$ dans G par transitivité de G, et donc $v_1 v_4 \dots v_n$ est également un cycle dont le nombre d'alternances est strictement plus petit, une contradiction.

Dans le second cas, si v_1 et v_3 sont comparables dans G', on ne peut avoir $v_3 \rightsquigarrow v_1$ dans G' parce qu'on aurait alors $v_2 \rightsquigarrow v_1$ dans G' par transitivité, ce qui contredit $v_1 \rightsquigarrow v_2$ dans G. On a donc $v_1 \rightsquigarrow v_3$ dans G', et donc $v_{n-1} v_3 \cdots v_n$ est également un cycle et son nombre d'alternances est strictement plus petit, une contradiction. On a donc démontré le résultat de l'indication.

Montrons à présent que, si $G = (V, E)$ a un conjugué, alors G admet un étiquetage d'accessibilité. Soit $G' = (V, E')$ le conjugué. Notons $G'_- = (V, E'_-)$ le graphe obtenu en inversant les arêtes de G', c'est-à-dire que $E'_- := \{(y, x) \mid (x, y) \in E'\}$. Par symétrie, ce graphe est toujours acyclique, et il est clair que c'est également un conjugué de G, puisque deux sommets sont comparables dans G' si et seulement s'ils le sont dans G'_-. Ainsi, en utilisant le résultat proposé dans l'indication, les graphes orientés $G'' = (V, E \cup E')$ et $G''_- = (V, E \cup E'_-)$ sont tous deux acycliques.

On calcule à présent un tri topologique de G'', c'est-à-dire une fonction injective $p : V \to \mathbb{N}$ qui garantit que, pour tous $u \neq v$ de G'', si $u \rightsquigarrow v$ alors $p(u) < p(v)$ (mais la réciproque n'est pas nécessairement vraie). L'existence d'un tri topologique utilise le fait que G'' et G''_- sont acycliques. Une telle fonction peut se calculer en temps linéaire en G'' en parcourant G'' en profondeur d'abord et en numérotant chaque sommet par ordre inverse de la date de fin de visite. On calcule de même un tri topologique q de G''_-, et on définit un étiquetage f de G à partir de p et q.

Il reste à montrer que f est effectivement un étiquetage d'accessibilité. Montrons d'abord que, pour tous $u \neq v$ in G, si $u \rightsquigarrow v$ alors $f(u) \leq f(v)$. Dans ce cas, on a $u \rightsquigarrow v$ dans G'' et dans G''_-, donc $p(u) \leq p(v)$ et $q(u) \leq q(v)$, et ainsi on a bien $f(u) \leq f(v)$. Le cas où $v \rightsquigarrow u$ est symétrique. Montrons réciproquement que si u et v sont incomparables dans G alors on a $f(u) \not\leq f(v)$ et $f(v) \not\leq f(u)$. Par définition du conjugué, u et v sont comparables dans G', et ils sont comparables dans l'autre direction dans G'_-. Ceci implique que $u \rightsquigarrow v$ dans l'un de G'', G''_-, et $v \rightsquigarrow u$ dans l'autre. Ainsi, on a $p(u) < p(v)$ et $q(u) > q(v)$, ou $p(u) > p(v)$ et $q(u) < q(v)$. Dans les deux cas, on a bien $f(u) \not\leq f(v)$ et $f(v) \not\leq f(u)$. On conclut donc que f est bien un étiquetage d'accessibilité, ce qui conclut la preuve.

[L'indication est le Lemme 3.51 de [DM41], et la preuve du résultat principal est inspirée du Théorème 3.61 de [DM41].]

Question 6. Soit f un étiquetage d'accessibilité de G : on notera $f(v) = (p(v), q(v))$ pour tout sommet $v \in V$. On construit le graphe orienté $G' = (V, E')$ où, pour tous $u \neq v$ de V, on a $(u, v) \in E$ ssi 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 que toute paire de sommets est incomparable dans G ssi elle est comparable dans G'.

Il est clair que G' est acyclique, parce que pour toute arête (u, v) de G, on a $p(u) < p(v)$. Ainsi, il ne reste plus qu'à montrer l'équivalence. Pour l'implication directe, soit deux sommets $u \neq v$ incomparables dans G. Ceci implique que $p(u) \neq p(v)$, parce que sinon on aurait forcément $f(u) \leq f(v)$ ou $f(v) \leq f(u)$, ce qui contredirait le fait que f est un étiquetage d'accessibilité. Ainsi, soit $p(u) < p(v)$, soit $p(v) < p(u)$, de sorte que u et v sont comparables dans G'.

Pour l'affirmation réciproque, soit $u \rightsquigarrow v$ deux sommets comparables de G', et soit $w_1 \dots w_n$ un chemin qui en témoigne, avec $u = w_1$ et $w_n = v$. Pour tout $1 \leq i < n$, comme w_i et w_{i+1} sont incomparables dans G par définition de G', on a forcément $q(w_i) > q(w_{i+1})$: en effet, si on avait $q(w_i) \leq q(w_{i+1})$, on aurait $f(w_i) \leq f(w_{i+1})$, ce qui contredirait le fait que f est un étiquetage d'accessibilité. Ainsi, par transitivité, on a $q(u) > q(v)$, mais $p(u) < p(v)$ par définition, donc f permet de conclure que u et v sont effectivement incomparables dans G.

Ainsi, on a montré qu'on pouvait construire, à partir d'un étiquetage d'accessibilité, un conjugué de G. Par la question précédente, on conclut qu'un graphe acyclique G admet un étiquetage d'accessibilité si et seulement s'il admet un conjugué.

Références

[DM41] Ben Dushnik and Edwin W. Miller. Partially ordered sets. *American journal of mathematics*, 1941.

A3 – Automates et langages probabilistes

Pour tout ensemble non-vide X, une *distribution* sur X est une fonction π de X dans l'ensemble $\mathbb{Q}_+$ des rationnels positifs. Le *support* de π est $\text{supp}(\pi) := \{x \in X \mid \pi(x) > 0\}$. On supposera toujours que le support d'une distribution est un ensemble *fini*.

Le *produit* de π par $v \in \mathbb{Q}_+$, noté $v \cdot \pi$, est la distribution sur X définie par $(v \cdot \pi)(x) := v \times \pi(x)$ pour tout $x \in X$. Pour π et π' deux distributions sur X, leur *somme*, notée $\pi + \pi'$, est la distribution sur X définie par $(\pi + \pi')(x) := \pi(x) + \pi'(x)$ pour tout $x \in X$. On dit que la distribution π est *normalisée* si on a $\sum_{x \in X} \pi(x) = 1$. On note $\Pi(X)$ l'ensemble des distributions normalisées sur X.

On fixe un alphabet Σ. Un *langage probabiliste* L est une distribution sur l'ensemble Σ^* des mots sur l'alphabet Σ. Pour $a \in \Sigma$, on note aL le langage probabiliste défini par $(aL)(w) := L(w')$ si w peut s'écrire comme aw' (c'est-à-dire que w n'est pas vide et commence par a) et $(aL)(w) := 0$ sinon. On note L_ϵ le langage probabiliste normalisé défini par $L_\epsilon(\epsilon) := 1$ et $L_\epsilon(w) := 0$ pour tout $w \neq \epsilon$.

Un *automate probabiliste* est une paire $A = (Q, \delta)$ où $Q = \{1, \ldots, n\}$ est l'ensemble d'états et δ est une fonction de Q dans $\Pi(\{\$\} \cup (\Sigma \times Q))$ telle que, pour tout $q \in \{1, \ldots, n\}$, pour tout $(a, q') \in \text{supp}(\delta(q))$, on a $q' > q$.

On définit par récurrence le langage probabiliste $\mathcal{L}(A, i)$ accepté par A à l'état $i \in Q$: on a $\mathcal{L}(A, n) := L_\epsilon$, et, pour $1 \leq i < n$, en notant $\pi := \delta(i)$, on a :

$$\mathcal{L}(A, i) := \pi(\$) \cdot L_\epsilon + \sum_{\substack{a \in \Sigma \\ j \in Q}} \pi(a, j) \cdot (a\mathcal{L}(A, j))$$

Le *langage* $\mathcal{L}(A)$ accepté par A est le langage probabiliste $\mathcal{L}(A, 1)$.

Question 0. Construire un automate probabiliste A tel que $\mathcal{L}(A)$ soit le langage probabiliste normalisé L sur $\{a, b\}$ défini par :
- ϵ a probabilité 0.1 dans L ;
- a a probabilité 0.2 dans L ;
- abb a probabilité 0.3 dans L ;
- b a probabilité 0.4 dans L.

Question 1. Montrer que, pour tout automate probabiliste A, le langage probabiliste $\mathcal{L}(A)$ est normalisé.

Question 2. Montrer que, réciproquement, pour tout langage probabiliste normalisé L, on peut construire un automate probabiliste A_L tel que $\mathcal{L}(A_L) = L$.

Question 3. Un automate probabiliste $A = (Q, \delta)$ est dit *déterministe* si, pour tout $q \in Q$, pour chaque $a \in \Sigma$, il existe au plus un unique $q' \in Q$ tel que $(a, q') \in \text{supp}(\delta(q))$. Montrer que, pour tout langage probabiliste normalisé L, on peut construire un automate probabiliste déterministe A'_L tel que $\mathcal{L}(A'_L) = L$.

Indication : On pourra décomposer L en le mot vide et en les sous-langages, pour $a \in L$, des mots de $\text{supp}(L)$ qui commencent par a.

Suite des questions

Question 4. On appelle *uniforme* un langage probabiliste normalisé L tel que $L(u) = L(v)$ pour tous $u, v \in \mathrm{supp}(L)$. Pour $n \in \mathbb{N}$, on considère le langage probabiliste uniforme L_n sur $\Sigma = \{a, b\}$ dont le support est le langage régulier $(a + b)^n$. Calculer, en fonction de n, le nombre d'états des automates probabiliste A_{L_n} et A'_{L_n} défini par la construction des deux questions précédentes. Proposer un automate probabiliste déterministe A'_n à $n + 1$ états tel que $\mathcal{L}(A'_n) = L_n$, et justifier brièvement sa correction.

Question 5. Soit $w = a_1 \dots a_n$ un mot de Σ^* de longueur $n \in \mathbb{N}$. Pour tout $S \subseteq \{1, \dots, n\}$, on note $w_{|S}$ le sous-mot de w obtenu en conservant les lettres placées aux positions de S : formellement, en notant $i_1, \dots, i_{|S|}$ les éléments de S dans l'ordre croissant, on définit $w_{|S} := w_{i_1} \cdots w_{i_{|S|}}$. Pour tout $w' \in \Sigma^*$, on note $L_{w'}$ le langage probabiliste normalisé qui donne probabilité 1 à w'. On définit le langage probabiliste $L_{\subseteq w}$ pour tout $w \in \Sigma^*$ de longueur $n \in \mathbb{N}$ comme suit :

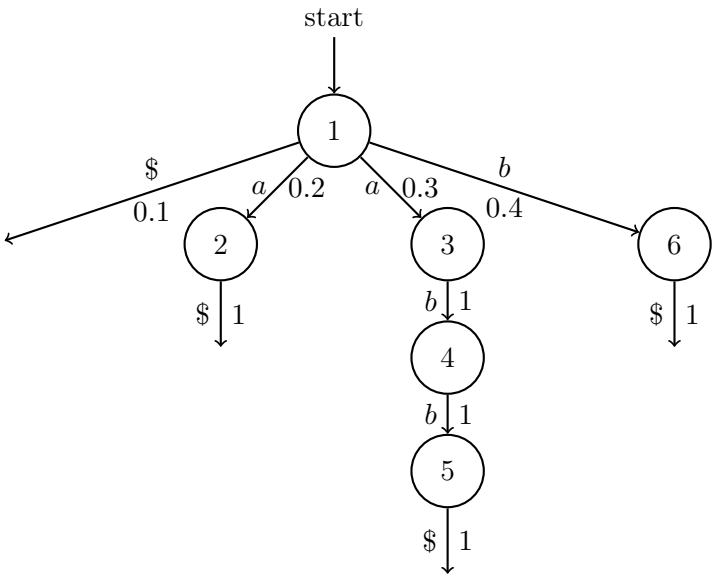
$$L_{\subseteq w} := \sum_{S \subseteq \{1,\dots,n\}} 2^{-n} \cdot L_{w_{|S}}.$$

Proposer un automate probabiliste A_w à $n+1$ états (non nécessairement déterministe) tel que $\mathcal{L}(A_w) = L_{\subseteq w}$, et prouver sa correction.

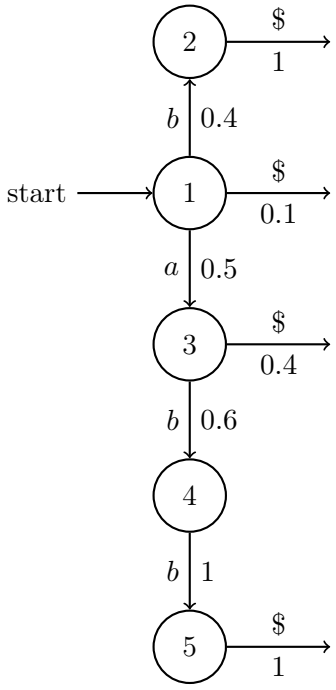
Question 6. Soit A un automate déterministe *non-probabiliste* tel que le langage $\mathcal{L}(A)$ accepté par A soit fini et non-vide. Expliquer comment construire un automate probabiliste déterministe A' tel que $\mathcal{L}(A')$ soit le langage probabiliste normalisé uniforme de support $\mathcal{L}(A)$.

Corrigé

Question 0. On peut par exemple prendre l'automate probabiliste suivant, dont on vérifie sans peine qu'il satisfait les conditions de la définition :



Cet automate correspond au résultat de la question 2 sur le langage décrit. On peut aussi accepter celui qui correspond au résultat de la question 3, à savoir :



Question 1. On montre l'affirmation pour $\mathcal{L}(A, i)$ par récurrence descendante finie avec prédécesseurs pour i de n à 1. Le cas de base correspond à $\mathcal{L}(A, n)$, qui vaut L_ϵ, et ce langage probabiliste est normalisé. Pour le cas de récurrence, supposons que l'affirmation est vraie pour un certain $1 < i \leq n$, c'est-à-dire que $\mathcal{L}(A, j)$ est normalisé pour tout $j \geq i$, et montrons que $\mathcal{L}(A, i-1)$ est normalisé. On appellera *masse* d'une distribution la somme de ses valeurs (qui vaut 1 pour les distributions normalisées) : on observe facilement que la masse de la somme de deux distributions est la somme des masses, et que la masse du produit d'une distribution par une valeur $v \in \mathbb{Q}_+$ est le produit de v et de la masse de la distribution originale. On va montrer que $\mathcal{L}(A, i-1)$ a pour masse 1. Dans la formule qui définit $\mathcal{L}(A, i-1)$, la masse du premier terme de la somme est $\pi(\$)$, et celui du deuxième terme de la somme est la somme de la masse, pour $a \in A$ et pour $j \in Q$ (ou, de façon équivalente par la condition sur le support, pour $j > i$), de $\pi(a, j)$; en effet, comme $\mathcal{L}(A, j)$ pour $j > i-1$ est normalisé par hypothèse de récurrence, il est clair que c'est également le cas de $a\mathcal{L}(A, j)$ pour tout $a \in \Sigma$, et ainsi la masse de l'expression sommée est $\pi(a, j)$. Ainsi, la masse totale est $\pi(\epsilon) + \sum_{a \in \Sigma, j > i-1} \pi(a, j)$, c'est-à-dire 1 car pi est une distribution normalisée. Ainsi, $\mathcal{L}(A, i-1)$ est normalisé, ce qui conclut le cas de récurrence. Du résultat que nous avons établi par récurrence, on déduit que $\mathcal{L}(A) = \mathcal{L}(A, 1)$ est normalisé, ce qui termine la preuve.

Question 2. Pour tout mot $w = a_1 \cdots a_n$ de Σ, on construit un automate probabiliste $A_w = (Q_w, \delta_w)$ tel que $\mathcal{L}(A_w)$ soit le langage probabiliste normalisé qui donne probabilité 1 à w et 0 à tout mot différent de w. On prend simplement $Q_w = \{1, \ldots, n+1\}$, on définit $\delta(n+1)$ comme la distribution normalisée donnant probabilité 1 à \$, et pour $1 \leq q \leq n$ on définit $\delta(q)$ comme la distribution normalisée donnant probabilité 1 à $(a_i, q+1)$. Il est clair que A_w satisfait les conditions d'un automate probabiliste, et que son langage est comme demandé ; on peut formellement montrer par récurrence descendante que $\mathcal{L}(A, i)$ pour $1 \leq i \leq n+1$ est la distribution normalisée donnant probabilité 1 au suffixe de longueur $(n+1) - i$ de w.

On construit à présent $A = (Q, \delta)$ en construisant, pour chaque mot non-vide $w \in \mathrm{supp}(L)$, l'automate $A_{w'}$ pour w' le suffixe de longueur $|w| - 1$ de w. (On construit un seul $A_{w'}$ même si plusieurs mots partagent le même suffixe w'.) On renumérote ensuite les états de chaque $A_{w'}$ à des intervalles disjoints consécutifs d'entiers à partir de 1, et on ajoute à Q un état 1 où $\delta(1)$ est défini comme suit : on donne probabilité $L(\epsilon)$ à \$, et, pour chaque mot non-vide $w \in \mathrm{supp}(L)$, en notant a la première lettre de w et

w' le suffixe de longueur $|w|-1$ de w, on donne probabilité $L(w)$ à $(a, q_{w'})$ où $q_{w'}$ est le numéro de l'état initial de $A_{w'}$. Il est clair que le résultat est bien un automate probabiliste et que son langage probabiliste est comme demandé.

Question 3. On montrera l'affirmation demandée par récurrence sur la longueur du plus long mot du support de L, c'est-à-dire que l'on montrera, pour tout $i \in \mathbb{N}$, la propriété suivante : pour tout langage probabiliste normalisé L, si le plus long mot de $\mathrm{supp}(L)$ a une longueur de i au plus, alors il existe un automate probabiliste A tel que $\mathcal{L}(A) = L$. Ceci permet d'établir le résultat, car le support d'un langage probabiliste est toujours fini.

Pour le cas de base $i = 0$, le seul langage probabiliste normalisé qui convient est le langage L_ϵ, qui est accepté par l'automate probabiliste déterministe $(\{1\}, \delta_\epsilon)$ où $\delta_\epsilon(1)$ est la distribution normalisée qui donne la probabilité 1 à \$.

Pour le cas de récurrence, supposons que l'on a prouvé l'affirmation au rang $i-1$ pour $i \in \mathbb{N}^*$, et démontrons-la au rang i. Fixons le langage probabiliste normalisé L. Pour chaque $a \in \Sigma$, soit L_a le langage probabiliste (généralement non normalisé) obtenu comme la restriction de L au domaine des mots sur l'alphabet Σ qui commencent par a. Soit s_a la masse de L_a : on appelle $\Sigma' \subseteq \Sigma$ l'ensemble $\{a \in \Sigma \mid s_a > 0\}$. Pour $a' \in \Sigma'$, le langage probabiliste $L'_{a'} := \frac{1}{s_{a'}} \cdot L_{a'}$ est alors normalisé. De plus, il est clair que l'on a :

$$L = L(\epsilon) \cdot L_\epsilon + \sum_{a' \in \Sigma'} s_{a'} \cdot L'_{a'}$$

Pour chaque $a' \in \Sigma'$, le langage L'_a est un langage probabiliste normalisé, et la longueur de son mot le plus long est strictement plus petite que la longueur du mot le plus long de L. Ainsi, par hypothèse de récurrence, pour chaque $a' \in \Sigma'$, on peut construire un automate probabiliste $A_{a'}$ tel que $\mathcal{L}(A_{a'}) = L_{a'}$; on note $n_{a'}$ son nombre d'états.

On construit à présent l'automate A pour L à partir des $A_{a'}$ pour $a' \in \Sigma'$. On renumérote d'abord les états des $A_{a'}$ par des intervalles d'entiers consécutifs disjoints, l'union disjointe de ces intervalles formant l'intervalle allant de 2 à $2 + \sum_{a' \in \Sigma'} n_{a'}$ exclu : on nomme $i_{a'}$ pour $a' \in \Sigma'$ le numéro du premier état de l'automate $A_{a'}$ après renumérotation. On ajoute ensuite un état 1 à l'automate A dont l'image par δ associe la probabilité $L(\epsilon)$ à \$, et associe, pour $a' \in \Sigma'$, la probabilité $s_{a'}$ au couple $(a', i_{a'})$. Noter que ceci garantit que la condition de déterminisme à l'état 1, et cette condition est remplie aux autres états par hypothèse de récurrence.

On vérifie que cette définition satisfait les conditions d'un automate probabiliste. Il est clair, comme L est normalisé, que $L(\epsilon) + \sum_{a'} s_{a'} = 1$, et ainsi $\delta(1)$ est normalisée ; les autres images de δ sont normalisées car les $A_{a'}$ sont des automates probabilistes. De plus, il est clair que la condition sur le support de δ est respectée, car c'est immédiat pour 1, et c'est vrai (même après renumérotation) sur les autres états, car les $A_{a'}$ sont des automates probabilistes. Il est ensuite aisé de vérifier que l'on a bien $\mathcal{L}(A) = L$, car l'équation ci-dessus correspond à la définition de $\mathcal{L}(A, 1)$, ce qui nous permet de conclure grâce à l'hypothèse de récurrence sur les $\mathcal{L}(A_{a'})$ pour $a' \in \Sigma'$. Ainsi, le cas de récurrence est montré, ce qui conclut la preuve.

Question 4. On montre facilement que, pour tout $n \in \mathbb{N}$, le nombre d'états de A_{L_n} est de $1 + 2^{n-1} \times n$. En effet, il y a d'abord l'état initial, et il y a ensuite un automate par suffixe w' dont la taille est $|w'| = (n-1)+1 = n$, et le nombre de suffixes w' est de 2^{n-1}. (Si on ne fait pas l'optimisation consistant à ne pas construire de $A_{w'}$ redondants, on obtient $1 + 2^n \times n$.)

On montre par récurrence sur $n \in \mathbb{N}$ que le nombre d'états de A'_{L_n} est de $2^{n+1} - 1$:

Cas de base $n = 0$ **:** L'automate A'_{L_0} a 1 état, ce qui est égal à $2^{0+1} - 1$;

Cas de récurrence : Si l'on suppose pour un certain $n \in \mathbb{N}$ que le nombre d'états de A'_{L_n} est de $2^{n+1} - 1$ (hypothèse de récurrence), alors le nombre d'états de $A'_{L_{n+1}}$ dans la construction précédente est de $1 + 2(2^{n+1} - 1)$, c'est-à-dire $2^{n+2} - 1$, ce qui prouve le cas de récurrence.

On peut à présent construire $A'_n = (\{1, \ldots, n+1\}, \delta'_n)$ en définissant $\delta'_n(n+1)$ comme la distribution normalisée qui donne probabilité 1 à \$, et en définissant $\delta'_n(i)$ pour $1 \leq i \leq n$ comme la distribution normalisée qui donne probabilité $1/2$ à $(a, i+1)$ et $1/2$ à $(b, i+1)$: noter que ceci remplit la condition de déterminisme. On prouve aisément par récurrence descendante finie que $\mathcal{L}(A'_n, i) = L_{n+1-i}$ pour tout $1 \leq i \leq n+1$, d'où $\mathcal{L}(A'_n) = L_n$, ce qui conclut.

Question 5. Pour construire l'automate et justifier de sa correction, on considérera les langages probabilistes $L_{\subseteq w,i}$ pour $1 \leq i \leq n+1$, définis comme $L_{\subseteq w,i} := L_{\subseteq w[i,\ldots,n]}$, où on dénote par $w[i \ldots n]$ le suffixe de longueur $n-i+1$ de w, qui commence à la position i (ainsi $w[n+1 \ldots n]$ est le mot vide ϵ). On a donc $L_{\subseteq w,1} = L_{\subseteq w}$.

On va d'abord montrer l'équation suivante :

$$L_{\subseteq w} = 2^{-n} \cdot L_\epsilon + \sum_{1 \leq i \leq n} 2^{-i} a_i \cdot L_{\subseteq w,i}.$$

Intuitivement, cette équation affirme que le mot vide a probabilité 2^{-n}, et que pour des mots non-vides, on saute un certain nombre de caractères, avec une probabilité qui décroît exponentiellement en le nombre de caractères sautés, puis on lit le prochain caractère et on lit un mot défini sur le suffixe qui suit.

Pour montrer cette équation, on va récrire l'équation originale qui définit $L_{\subseteq w}$ afin de faire apparaître les $L_{\subseteq w,i}$. Dans cette équation originale, récrivons la somme sur $S \subseteq \{1, \ldots, n\}$ pour traiter séparément le cas où $S = \emptyset$, et distinguer les autres cas suivant la plus petite valeur de S. On obtient :

$$L_{\subseteq w} = 2^{-n} \cdot L_\epsilon + \sum_{1 \leq i \leq n} \sum_{\substack{S \subseteq \{1,\ldots,n\} \\ |S| \geq 1 \\ \min S = i}} 2^{-n} \cdot L_{w_{|S}}.$$

On observe à présent que l'on peut récrire la somme interne : au lieu d'énumérer les sous-ensembles de $\{1, \ldots, n\}$ non-vides dont le plus petit élément vaut i, il est équivalent d'énumérer les sous-ensembles de $\{i+1, \ldots, n\}$ que l'on peut ajouter à i (ce sont intuitivement les éléments suivants de l'ensemble). Au demeurant, on sait, une fois qu'on a choisi le plus petit élément i de S, que le premier caractère de $L_{w_{|S}}$ est a_i, ce qui nous permet de récrire l'équation comme :

$$L_{\subseteq w} = 2^{-n} \cdot L_\epsilon + \sum_{1 \leq i \leq n} \sum_{S' \subseteq \{i+1,\ldots,n\}} 2^{-n} \cdot (a_i L_{w_{|S'}}).$$

On factorise à présent 2^{-i}, et on observe ensuite que, pour tous langages probabilistes L_1 et L_2 et $a \in \Sigma$, on a $a(L_1 + L_2) = (aL_1) + (aL_2)$. Ainsi, on peut factoriser a_i, et obtenir :

$$L_{\subseteq w} = 2^{-n} \cdot L_\epsilon + \sum_{1 \leq i \leq n} 2^{-i} a_i \cdot \sum_{S' \subseteq \{i+1,\ldots,n\}} 2^{-(n-i)} L_{w_{|S'}}.$$

On peut à présent, pour chaque valeur $1 \leq i \leq n$ de la somme extérieure, soustraire $i+1$ à toutes les valeurs de S', et remplacer w par son suffixe $w[i+1 \ldots n]$ commençant à la position $i+1$. Ainsi, on obtient :

$$L_{\subseteq w} = 2^{-n} \cdot L_\epsilon + \sum_{1 \leq i \leq n} 2^{-i} a_i \cdot \sum_{S' \subseteq \{1,\ldots,n-i\}} 2^{-(n-i)} L_{w[i+1\ldots n]_{|S'}}.$$

On reconnaît alors la définition de $L_{\subseteq w,i}$, ce qui nous permet d'aboutir à l'équation annoncée.

Cette équation suggère la construction de l'automate $A_w = (\{1, \ldots, n+1\}, \delta)$ comme suit : on fixe $\delta(n+1)$ comme la distribution normalisée qui donne probabilité 1 à \$, et pour $1 \leq i < n+1$, on définit $\delta(i)$ comme donnant la probabilité 2^{n-i+1} à \$ et, pour tout $i < j \leq n+1$, comme donnant la probabilité 2^{j-i} au couple (a_j, j). On prouve par une récurrence descendante immédiate que, pour tout $1 \leq i \leq n+1$, on a $\mathcal{L}(A, i) = L_{\subseteq w,i}$: le cas de récurrence utilise l'équation ci-dessus, appliquée aux suffixes de w.

Question 6. On considère l'automate déterministe non-probabiliste $A = (Q, q_1, F, \delta)$, où Q est l'ensemble fini d'états (on note $n := |Q|$), $q_1 \in Q$ est l'état initial, $F \subseteq Q$ est l'ensemble des états finaux, et $\delta : Q \times \Sigma \to Q$ est la fonction partielle de transition. Comme $\mathcal{L}(A)$ est fini, il est clair que, quitte à éliminer les états inutiles de A, on peut supposer que A ne contient pas de *cycle*, c'est-à-dire qu'il n'y a pas de séquence d'états $q_1, \ldots, q_n$ tel que $q_n = q_1$ et, pour tout $1 \leq i < n$, il existe $a_{i+1} \in \Sigma$ tel que $q_{i+1} \in \delta(q_i, a_{i+1})$. En effet, l'existence d'un tel cycle implique que le langage accepté par A est infini (à condition que les états du cycle soient accessibles et coaccessibles, ce qui est assuré par notre pré-traitement de A). L'absence de tels cycles assure que l'on peut renuméroter les états de sorte à avoir $Q = \{1, \ldots, n\}$ et de sorte que, pour tous $1 \leq i, j \leq n$ et $a \in \Sigma$, si $j \in \delta(i, a)$ alors $j > i$: le numéro de chaque état peut être obtenu par un tri topologique, c'est-à-dire comme le numéro dans l'inverse du tri par date de fin de visite dans une exploration en profondeur d'abord. De plus, comme on a retiré les états inutiles, on peut supposer que l'état initial est 1 (car l'état 1 est inutile sinon).

On calcule ensuite, pour chaque état $1 \leq i \leq n$, le nombre m_i de mots acceptés par l'automate modifié en prenant l'état i pour état final. Ceci peut être calculé pour i de n à 1 en prenant $m_n := 1$ (l'état n ne peut pas avoir de transitions sortantes, et il est forcément final parce qu'il est utile), et pour $1 \leq i < n$ on calcule m_i comme la somme, pour $a \in \Sigma$ tel qu'il existe $j_a > i$ tel que $j_a \in \delta(i, a)$ (où j_a est unique pour i et a parce que l'automate A est déterministe et reste déterministe après les modifications effectuées), de m_{j_a}, somme à laquelle on ajoute 1 si l'état i est final. Il est clair que ce résultat est correct parce que les ensembles de mots sur lesquels on somme pour chaque valeur de i sont disjoints (il y a le mot vide, et les autres ensembles contiennent uniquement des mots non-vides qui diffèrent par leur première lettre).

On définit enfin l'automate probabiliste $A' = (Q, \delta')$ comme suit, ce qui revient intuitivement à donner une probabilité aux transitions de A : pour tout $1 \leq i \leq n$, on définit $\delta'(i)$ comme la distribution qui donne la probabilité 0 à \$ si $i \notin F$ et $\frac{1}{m_i}$ sinon, et donne à (a, j) pour $a \in \Sigma$ et $j > i$ la probabilité 0 si $j \notin \delta(a, i)$ et m_j/m_i sinon : notons que ceci satisfait la condition de déterminisme, par déterminisme de l'automate A. Il est facile de constater que le résultat est bien un automate probabiliste, que la distribution est correctement normalisée, et on prouve par récurrence descendante sur l'état que le langage probabiliste de l'automate est correct (à savoir qu'il capture le langage probabiliste normalisé uniforme dont le support est le langage de l'automate non-probabiliste A initial où l'on rend initial l'état que nous avons numéroté i).

A4 – Ensembles semilinéaires

Un ensemble $S \subseteq \mathbb{N}$ est dit *linéaire* s'il existe $b \in \mathbb{N}$, $n \in \mathbb{N}$, et $v_1, \ldots, v_n \in \mathbb{N}^*$ tels que :

$$S = \{b + \sum_{1 \leq i \leq n} x_i v_i \mid x_1, \ldots, x_n \in \mathbb{N}\}.$$

Un ensemble $S \subseteq \mathbb{N}$ est dit *semilinéaire* s'il existe $m \in \mathbb{N}$ et des ensembles linéaires $S_1, \ldots, S_m$ tels que $S = \bigcup_{1 \leq i \leq m} S_i$.

Question 0. Expliquer pourquoi tout sous-ensemble fini de $\mathbb{N}$ est semilinéaire.

Question 1. Donner un exemple de sous-ensemble de $\mathbb{N}$ qui n'est *pas* semilinéaire.

Question 2. On dit qu'un ensemble $S \subseteq \mathbb{N}$ est *ultimement périodique* s'il existe $t \in \mathbb{N}$ et $p \in \mathbb{N}^*$ tels que, pour tout $s \geq t$, on a $s \in S$ si et seulement si $s + p \in S$. Montrer que tout ensemble ultimement périodique est semilinéaire.

Question 3. Montrer que, réciproquement, tout ensemble semilinéaire est ultimement périodique. **Indication :** On pourra montrer que la définition de l'ultime périodicité peut se reformuler avec une implication plutôt qu'une équivalence.

Question 4. Un ensemble linéaire est dit *unaire* si on a $n = 1$, et *nullaire* si on a $n = 0$. Déduire des questions précédentes que tout ensemble semilinéaire peut s'écrire comme une union finie d'ensembles linéaires unaires et d'ensembles linéaires nullaires. Proposer un algorithme qui calcule cette représentation.

Suite des questions

Question 5. On s'intéresse à présent aux langages réguliers sur l'alphabet $\Sigma = \{a\}$. On admettra le théorème de Kleene : un langage est régulier si et seulement s'il est décrit par une expression rationnelle, ce qui est le cas si et seulement s'il est reconnu par un automate fini.

Montrer que, pour tout ensemble semilinéaire S, le langage $\{a^i \mid i \in S\}$ sur l'alphabet $\Sigma = \{a\}$ est un langage régulier.

Question 6. Montrer que, pour toute expression rationnelle e sur l'alphabet $\Sigma = \{a\}$, si l'on note $L(e)$ le langage capturé par e, alors l'ensemble $\{n \in \mathbb{N} \mid a^n \in L(e)\}$ est semilinéaire. Conclure.

Corrigé

Question 0. Pour $S \subseteq \mathbb{N}$ fini de cardinal $k \in \mathbb{N}$, on écrit $S = \{b_1, \ldots, b_k\}$, et on définit l'ensemble $S_i = \{b_i\}$ pour tout $1 \leq i \leq k$. Il suffit ainsi de montrer que tout ensemble singleton est linéaire. C'est le cas, car tout ensemble singleton $\{s\}$ peut être exprimé en prenant $b := s$ et $n := 0$.

Question 1. Pour un exemple de sous-ensemble de $\mathbb{N}$ qui n'est pas semilinéaire, on peut par exemple prendre l'ensemble des puissances de deux, $\{2^i \mid i \in \mathbb{N}\}$. Pour montrer que cet ensemble n'est pas linéaire, on montre que, pour tout ensemble linéaire S tel que $n > 0$, si S contient les puissances de deux, alors S contient un élément qui n'est pas une puissance de deux. En effet, procédons par l'absurde et supposons qu'il y ait un ensemble linéaire S qui ne contienne que des puissances de deux et qui satisfasse $n > 0$. Prenons i tel que $2^i > v_1$, et $j \in \mathbb{N}$ tel que $b + j \cdot v_1 > 2^i$. Comme $b + j \cdot v_1 \in S$, par hypothèse, il existe $k > i$ tel que $b + j \cdot v_1 = 2^k$, et l'élément de S qui s'exprime comme $b + (j+1)v_1$ est alors $> 2^k$ mais il est égal à $2^k + v_1 < 2^k + 2^i < 2^{k+1}$, donc ce n'est pas une puissance de deux, une contradiction. Ainsi, un ensemble semilinéaire qui contient un ensemble linéaire avec $n > 0$ ne contient pas que des puissances de deux, donc pour exprimer l'ensemble des puissances de deux comme un ensemble semilinéaire, il faut que tous les ensembles linéaires de l'union satisfassent $n = 0$. Mais il est clair qu'un tel ensemble semilinéaire est fini, donc il ne peut pas non plus être l'ensemble des puissances de deux.

On peut aussi montrer le résultat en prenant n'importe quel autre ensemble de croissance rapide. Alternativement, on peut prendre l'ensemble $\mathcal{P}$ des nombres premiers, et montrer que tout ensemble linéaire tel que $n > 0$ contient un nombre composite, à savoir, $b + b \cdot v_1$ si $b \geq 2$, ou $4 \cdot v_1$ si $b = 0$, ou $b + (2 + v_1) \cdot v_1 = (v_1 + 1)^2$ si $b = 1$.

Question 2. Soit S un ensemble ultimement périodique, et soit $t \in \mathbb{N}$ et $p \in \mathbb{N}$ les valeurs correspondantes. On écrit $S = S' \cup S''$, où $S' := S \cap \{1, \ldots, p-1\}$, et $S'' := S \cap (\mathbb{N} \setminus \{1, \ldots, p-1\})$. L'ensemble S' est fini, donc il est semilinéaire d'après la question 0. Ainsi, il suffit de montrer que S' est semilinéaire. Pour $0 \leq i < p$, on pose $S'_i := \{s \in S' \mid (s - t) \bmod p = i\}$. Il est clair qu'on a $S' = \bigcup_{0 \leq i < p} S'_i$, donc il suffit de montrer que, pour tout $0 \leq i < p$, l'ensemble S'_i est semilinéaire. Fixons $0 \leq i < p$. Si l'ensemble S'_i est vide, alors il est clairement semilinéaire ; sinon, supposons qu'il contienne un élément s. On a alors $s \geq t$; comme S est ultimement périodique, on observe facilement que S'_i est alors l'ensemble $\{t + jp + i \mid j \in \mathbb{N}\}$. Ceci permet de déduire que S'_i est un ensemble linéaire, car on peut l'écrire avec $b := t + i$, $n := 1$, et $v_1 := p$. Ainsi, on a prouvé le résultat demandé pour les S'_i, donc pour S', donc pour S.

Question 3. On montre suivant l'indication que l'on peut donner une définition alternative équivalente pour les ensembles ultimement périodiques : un ensemble S est ultimement périodique si et seulement s'il existe $t \in \mathbb{N}$ et $p \in \mathbb{N}$ tels que, pour tout $s \geq t$, si $s \in S$ alors $s + p \in S$ (mais on n'impose pas que l'implication réciproque soit vraie). En effet, il est clair que la condition originale donnée dans l'énoncé

implique cette condition reformulée. À l'inverse, pour un ensemble S qui satisfait la condition reformulée, montrons que S satisfait la condition originale. Fixons $t \in \mathbb{N}$ et $p \in \mathbb{N}$ donnés par la condition reformulée. Considérons la fonction f qui à $i \in \mathbb{N}$ associe l'ensemble $\{j \in \{0, \ldots, p-1\} \mid t + pi + j \in S\}$. La condition reformulée assure que la fonction f est croissante au sens de l'inclusion ensembliste, puisque, pour tout $i \in \mathbb{N}$ et $0 \leq j < p$, si $t + ip + j \in S$ alors $t + (i+1)p + j \in S$. Ainsi, comme f est croissante et bornée par $\{0, \ldots, p-1\}$, il existe i_0 tel que, pour tout $i \geq i_0$, on a $f(i) = f(i_0)$. (Si on ne désire pas montrer ce fait en considérant l'ordre induit par l'inclusion ensembliste, on peut le montrer en considérant la fonction qui associe à $i \in \mathbb{N}$ le cardinal de $f(i)$, et constater que les valeurs de cette fonction forment une suite croissante et bornée, d'où l'existence de i_0, et on obtient à partir de là l'affirmation sur f.)

On pose à présent $t' := t + i_0 p$ et $p' := p$, et on vérifie que S est ultimement périodique au sens de la condition originale pour t' et p'. En effet, pour tout $s \geq t'$, en considérant la division euclidienne de $s - t'$ par p, on peut écrire $s = t + i'p + j$, où $i' \geq i_0$ et $j \in \{0, \ldots, p-1\}$. On sait à présent que $f(i') = f(i'+1)$, et ainsi $s \in S$ ssi $j \in f(i')$ ssi $j \in f(i'+1)$ ssi $s + p \in S$, ce qui établit que la condition originale est vérifiée. Ainsi, on a montré que la condition reformulée est équivalente à la condition originale. On utilisera la condition reformulée dans ce qui suit.

Il suffit de montrer que tout ensemble linéaire est ultimement périodique. En effet, une fois ceci fait, pour un ensemble semilinéaire $S_1 \cup \cdots \cup S_m$, en prenant $t_1, \ldots, t_m$ et $p_1, \ldots, p_m$ les valeurs des S_i, en prenant $t = \max_i t_i$ et $p = \prod_i p_i$, il est clair que, pour tout $n \geq t$, si $n \in S$, alors on a $n \in S_i$ pour un certain $1 \leq i \leq m$. En notant $p'_i := \prod_{j \neq i} p_i$, comme $n \geq t_i$, on peut appliquer p'_i fois la périodicité, et conclure que $n + p'_i p_i \in S$. Ainsi S est ultimement périodique (en utilisant la condition reformulée).

Montrons ainsi l'implication demandée pour les ensembles linéaires. Pour un ensemble linéaire S défini par $b, n \in \mathbb{N}$ et $v_1, \ldots, v_n \in \mathbb{N}^*$, si $n = 0$, il est clair que si l'on prend $t := b + 1$ et p quelconque, alors S est ultimement périodique (avec n'importe laquelle des deux conditions). Si $n > 0$, il est clair que si l'on prend $t := b$ et $p := v_1$, on a que pour tout $n \geq t$, si $n \in S$ alors $n + p \in S$, et ainsi S est ultimement périodique (avec la condition reformulée).

Question 4. D'après la question 3, un ensemble semilinéaire est ultimement périodique. À présent, d'après la question 2, et en observant le détail de la preuve, on peut écrire tout ensemble ultimement périodique comme un ensemble semilinéaire où chaque ensemble linéaire est nullaire ou unaire.

Pour calculer cette représentation, on peut par exemple procéder de la façon suivante. Étant donné un ensemble semilinéaire $S = S_1 \cup \cdots \cup S_m$, chaque S_i s'écrivant comme b_i, n_i, et $v^i_1, \ldots, v^i_{n_i}$, on pose $p := \mathrm{lcm}_{1 \leq i \leq m, 1 \leq j \leq n_i} v^i_j$, où lcm désigne le plus petit commun multiple, $p' := \max_{1 \leq i \leq m, 1 \leq j \leq n_i} v^i_j$, et $b := \max_i b_i + p(p'+1)$.

On calcule ensuite explicitement les éléments de S plus petits que b. On effectue ce calcul pour chaque $1 \leq i \leq m$, en considérant le graphe (défini implicitement) $G_i := (\{b_i, \ldots, b\}, \{(k, k + v^i_j) \mid 1 \leq j \leq n_i, b_i \leq k \leq b - v^i_j\})$, et en déterminant par un parcours de G_i les valeurs accessibles à partir de b_i. On écrit ainsi $S \cap \{1, \ldots, b\}$ comme une union d'ensembles linéaires nullaires.

On observe maintenant que notre choix de b garantit que chaque S_i est ultimement périodique de période p à partir de $t := b$ (au sens de la définition originale d'ultime périodicité). Pour ce faire, il suffit de raisonner sur les restes dans la division euclidienne par p qui sont accessibles. Si S_i est déjà nullaire, il n'y a rien à faire : aucun reste n'est accessible. Sinon, il y a p restes, seul un reste est accessible initialement (à savoir, celui de b_i), tout reste accessible doit pouvoir être atteint en au plus p étapes, et à chaque étape on augmente la valeur du nombre atteint par au plus p'. Ainsi, à partir de b, chaque S_i est périodique. Il suffit donc de regarder, pour chaque $1 \leq i \leq m$, quelles valeurs dans $\{b_i + pp', \ldots, b_i + p(p'+1) - 1\}$ sont atteintes, pour savoir quels ensembles linéaires unaires il faut écrire.

Cet algorithme s'exécute en temps polynomial en la *valeur* des entiers dans la description de S fournie en entrée, c'est-à-dire en temps polynomial si ces entiers sont écrits en unaire ; et en temps simplement exponentiel sinon.

Question 5. Comme l'ensemble des langages réguliers est clos par union, il suffit de montrer le résultat pour S un ensemble linéaire. Soit $n \in \mathbb{N}$ et $v_1, \ldots, v_n \in \mathbb{N}^*$ comme dans la définition. On écrit le langage demandé comme $a^b(\sum_{1 \leq i \leq n} a^{v_i})^*$. La correction de cette construction est immédiate.

Question 6. On procède par induction sur l'expression rationnelle e.

Si e dénote l'ensemble vide, alors le résultat est clair, en prenant $m := 0$. Si e dénote le langage comportant uniquement le mot vide, alors le résultat est clair, en prenant $m := 1$ et S_1 avec $b := 0$ et $n := 0$. Si e dénote le langage comportant uniquement le mot a, alors on prend $m := 1$ et S_1 avec $b := 1$ et $n := 0$.

Si $e = e_1 + e_2$ pour deux expressions rationnelles e_1 et e_2, si l'on suppose par induction que les ensembles S_1 et S_2, définis à partir de e_1 et e_2 comme dans l'énoncé, sont semilinéaires, alors on exprime S comme l'union des ensembles linéaires de S_1 et de ceux de S_2.

Si $e = e_1 e_2$, en définissant S_1 et S_2 comme précédemment, il suffit de montrer que, pour toute paire S et S' d'ensembles linéaires, l'ensemble $S + S'$ défini comme $\{x + x' \mid x \in S, x' \in S'\}$ est linéaire. En effet, on étend ensuite le résultat aux semilinéaires en exploitant la distributivité de cet opérateur $+$ sur l'union. Si l'on écrit S comme $b, n \in \mathbb{N}$ et $v_1, \ldots, v_n \in \mathbb{N}^*$, et de même $b', n' \in \mathbb{N}$ et $v'_1, \ldots, v'_{n'} \in \mathbb{N}^*$ pour S', il est clair qu'on peut exprimer $S + S'$ comme $b + b'$, $n + n'$, $v_1, \ldots, v_n, v'_1, \ldots, v'_{n'}$. Ceci conclut le cas de la concaténation.

Si $e = (e')^*$, on définit S' par induction comme précédemment, et on écrit $S' = S_1 \cup \cdots \cup S_m$, où $S_1, \ldots, S_m$ sont des ensembles linéaires. Pour tout ensemble linéaire S avec $b, n \in \mathbb{N}$ et $v_1, \ldots, v_n \in \mathbb{N}^*$, on note S^+ l'ensemble linéaire défini par b, $n+1$, et $v_1, \ldots, v_{n+1}$ avec $v_{n+1} := b$, et on définit À présent, pour tout ensemble $T \subseteq \{1, \ldots, m\}$, on définit $S_T := \sum_{i \in T} S_i^+$, pour la somme sur les ensembles linéaires définie au cas précédent. On définit $S := \bigcup_{T \subseteq \{1,\ldots,m\}} S_T$, et on montre que S est l'ensemble demandé pour e. Pour ce faire, il faut montrer que S contient exactement les sommes finies d'éléments de S'.

Pour la direction directe, soit $n \in S$, et montrons que n s'écrit comme une somme finie d'éléments de S'. Il existe $T \subseteq \{1, \ldots, m\}$ tel que $n \in S_T$, c'est-à-dire $n \in \sum_{i \in T} S_i^+$. Ainsi, n s'écrit comme $n' + n^+$, où n' est un élément de $\sum_{i \in T} S_i$, et n^+ est une somme finie des b des S_i pour $i \in T$. Mais n' est donc une somme finie d'éléments de S', et n^+ l'est également, donc $n = n' + n^+$ est également une somme finie d'éléments de S', ce qui établit la direction directe.

Pour la direction inverse, soit une somme finie $n = n_1 + \cdots + n_k$ pour $k \in \mathbb{N}$ et $n_i \in S'$ pour tout $1 \leq i \leq k$, et montrons que $n \in S$. Quitte à permuter les termes de la somme, pour T un sous-ensemble de $\{1, \ldots, m\}$ qui couvre tous les linéaires utilisés parmi les n_i, on peut écrire $n = \sum_{i \in T} p_i$, où chaque p_i est une somme *non-vide* d'éléments de S_i. À présent, il est clair qu'une somme non-vide d'éléments de S_i est un élément de S_i^+, dont n est une somme d'éléments de S_i^+ pour $i \in T$, d'où l'on conclut que $n \in S$.

On a donc prouvé l'affirmation par induction sur e, ce qui conclut la preuve.

En utilisant le théorème de Kleene, on peut en déduire que, sur l'alphabet $\Sigma = \{a\}$, les langages réguliers sont exactement les ensembles de mots dont les longueurs sont un ensemble semi-linéaire de $\mathbb{N}$.

A5 – Formules de provenance

Pour tout ensemble fini non-vide de *variables* $\mathcal{X} = \{X_1, \ldots, X_m\}$, on appelle *formule sur* $\mathcal{X}$ une formule de la logique propositionnelle ayant $\mathcal{X}$ comme ensemble de variables.

Question 0. Rappeler la définition d'une formule de la logique propositionnelle.

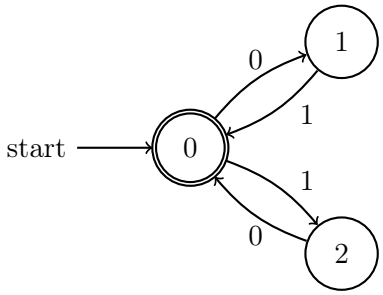
Une *valuation* de $\mathcal{X}$ est une fonction $\nu : \mathcal{X} \to \{0, 1\}$ qui attribue une valeur booléenne à chaque variable de $\mathcal{X}$. Étant donné une valuation ν, la *valeur* sous ν d'une formule Φ sur $\mathcal{X}$ est une valeur dans $\{0, 1\}$ que l'on écrit $\nu(\Phi)$ et que l'on définit comme en cours.

On fixe l'alphabet $\Sigma = \{0, 1, ?\}$. Étant donné un mot $w = a_1 \cdots a_n$ de longueur n sur l'alphabet Σ, son *ensemble de variables* est $\mathcal{X}_n = \{X_1, \ldots, X_n\}$. Une *valuation* de w est une valuation de $\mathcal{X}_n$. Chaque valuation ν de w permet de définir un mot $\nu(w) = a'_1 \cdots a'_n$ sur l'alphabet $\{0, 1\}$ comme suit :

$$\text{pour tout } 1 \leq i \leq n, \text{ on a } a'_i := \begin{cases} a_i & \text{si } a_i \in \{0, 1\} \\ \nu(X_i) & \text{si } a_i = ? \end{cases}$$

Étant donné un mot w de longueur n sur l'alphabet Σ et un automate A sur l'alphabet $\{0, 1\}$, une *formule de provenance pour w sur A* est une formule Φ sur $\mathcal{X}_n$ telle que, pour toute valuation ν de w, on a $\nu(\Phi) = 1$ si et seulement si A accepte $\nu(w)$.

Question 1. Calculer une formule de provenance pour le mot $w = 0??1??$ sur l'automate suivant :



Question 2. Proposer un algorithme naïf qui, étant donné un mot $w \in \Sigma^*$ et un automate A, calcule une formule de provenance pour w sur A. Discuter de l'efficacité de l'algorithme proposé.

Question 3. Étant donné un mot $w \in \Sigma^*$ de longueur n et un automate A ayant Q comme ensemble d'états, pour $q \in Q$ et pour $0 \leq i \leq n$, une *formule partielle de provenance pour w sur A en (i, q)* est une formule $\Phi_{i,q}$ sur $\mathcal{X}_i = \{X_1, \ldots, X_i\}$ telle que, pour toute valuation ν de $\mathcal{X}_i$, on a $\nu(\Phi_{i,q}) = 1$ si et seulement si A peut aboutir à l'état q depuis l'état initial en lisant le préfixe de longueur i de $\nu(w)$.

Pour $0 \leq i < n$ et $q \in Q$, proposer une expression de $\Phi_{i+1,q}$ en fonction des $\Phi_{i,q'}$ pour $q' \in Q$. **Indication :** On distinguera plusieurs cas selon la valeur de a_{i+1}.

Question 4. Déduire de la question précédente un algorithme plus sophistiqué qui, étant donné un mot $w \in \Sigma^*$ et un automate A, calcule une formule de provenance pour w sur A. Discuter de l'efficacité de cet algorithme. Comment pourrait-on rendre l'algorithme plus efficace en modifiant la représentation des formules ?

Corrigé

[La construction présentée dans cet exercice est une version simplifiée de celle de [ABS15], adaptée à un langage sur les mots.]

Question 0. C'est une question de cours. On pourra par exemple prendre la définition suivante : une *formule de la logique propositionnelle* Φ sur $\mathcal{X}$, ou simplement *formule*, est donnée par la grammaire suivante :

— Si $X_i \in \mathcal{X}$ alors X_i est une formule ;
— 0 et 1 sont des formules ;
— Si Φ est une formule alors $\neg\Phi$ est une formule ;
— Si Φ et Φ' sont des formules alors $\Phi \wedge \Phi'$ et $\Phi \vee \Phi'$ sont des formules.

Question 1. On obtient par exemple :

$$\Phi := X_2 \wedge \neg X_3 \wedge ((X_5 \wedge \neg X_6) \vee (\neg X_5 \wedge X_6)).$$

Question 2. Notons n la longueur de w. On énumère simplement toutes les valuations du mot w. Pour chaque valuation ν, on vérifie si $\nu(w)$ est accepté par A. On écrit ensuite la formule comme une disjonction, sur les valuations ν telles que $\nu(w)$ est accepté par A, de la formule indiquant qu'on a exactement cette valuation. Formellement :

$$\Phi := \bigvee_{\substack{\nu:\mathcal{X}_n\to\{0,1\} \\ A \text{ accepte } \nu(w)}} \Big(\bigwedge_{\substack{X_i\in\mathcal{X}_n \\ \nu(X_i)=1}} X_i \Big) \wedge \Big(\bigwedge_{\substack{X_i\in\mathcal{X}_n \\ \nu(X_i)=0}} \neg X_i \Big).$$

Le nombre de valuations de w étant exponentiel en n, cet algorithme s'exécute en temps exponentiel en son entrée, ce qui n'est pas efficace.

Question 3. On note $\delta \subseteq Q \times \{0,1\} \times Q$ la relation de transition de l'automate A : pour tout $(q,l,q') \in Q \times \{0,1\} \times Q$, on a $(q,l,q') \in \delta$ si et seulement si l'automate A a une transition étiquetée par l allant de q à q'.

Soit $0 \leq i < n$. On considère le préfixe w' de longueur i de $\nu(w)$, et on note $a' := \nu(a_{i+1})$. On distingue deux cas, suivant que $a_{i+1} \in \{0,1\}$, ou que $a_{i+1} = ?$.

Dans le premier cas, si $a_{i+1} \in \{0,1\}$, on définit pour tout $q \in Q$:

$$\Phi_{i+1,q} := \bigvee_{\substack{q'\in Q \\ (q',a_{i+1},q)\in\delta}} \Phi_{i,q'}$$

En effet, pour toute valuation ν de w, l'automate A peut aboutir à l'état q à la fin de la lecture de $w'a'$ si et seulement s'il existe un état $q' \in Q$ tel que A peut aboutir à l'état q' à la fin de la lecture de w', et il y a une transition étiquetée par a' allant de q' à q.

Dans le second cas, si $a_{i+1} = ?$, on définit pour tout $q \in Q$:

$$\Phi_{i+1,q} := \Big(X_{i+1} \wedge \Big(\bigvee_{\substack{q'\in Q \\ (q',1,q)\in\delta}} \Phi_{i,q'} \Big)\Big) \vee \Big(\neg X_{i+1} \wedge \Big(\bigvee_{\substack{q'\in Q \\ (q',0,q)\in\delta}} \Phi_{i,q'} \Big)\Big)$$

En effet, pour toute valuation ν de w, il y a deux possibilités pour que l'automate A aboutisse à l'état q à la fin de la lecture de $w'a'$: soit $\nu(X_{i+1}) = 0$ et A peut aboutir à l'état q à la fin de la lecture de $w'0$, soit $\nu(X_{i+1}) = 1$ et A peut aboutir à l'état q à la fin de la lecture de $w'1$. Ainsi, pour chaque branche de cette alternative, on teste la valeur de X_{i+1}, et on prend la conjonction du littéral correspondant avec une expression similaire à ce que l'on a obtenu pour le premier cas ci-dessus.

Question 4. On observe tout d'abord qu'on peut écrire une formule de provenance Φ pour w sur A comme $\bigvee_{q \in F} \Phi_{|w|,q}$, où $F \subseteq Q$ désigne l'ensemble des états finaux de A et Q désigne l'ensemble des états de A. Ainsi, il suffit de savoir calculer les formules partielles de provenance. On procédera par récurrence finie sur $0 \leq i \leq n$, pour tout $q \in Q$:

Cas de base $i = 0$ **:** Pour tout $q \in Q$, on a $\Phi_{i,q} = 1$ si q est un état initial, et $\Phi_{i,q} = 0$ sinon.

Cas de récurrence : Si l'on suppose que l'on a calculé les formules partielles de provenance $\Phi_{i,q}$ pour un certain $0 \leq i < n$ et pour tout $q \in Q$, alors les formules partielles de provenance $\Phi_{i+1,q}$ pour $i + 1$ et pour tout q s'obtiennent simplement avec les équations de la question précédente.

Il est aisé de se convaincre que la représentation explicite de la formule calculée est exponentielle en général, donc cet algorithme sophistiqué n'est pas plus efficace que l'algorithme naïf de la question 2. En revanche, et contrairement à l'algorithme naïf, l'algorithme sophistiqué réutilise intuitivement un nombre limité de sous-formules (à savoir, les $\Phi_{i,q}$), dont la définition en fonction des précédentes est de petite taille. Ainsi, on peut exécuter plus efficacement l'algorithme si on s'autorise de définir des sous-formules et à les réutiliser plus tard sans les récrire entièrement.

Formellement, on peut définir la notion d'un *système de formules* $\Psi = \Phi_1, \ldots, \Phi_n$ sur $\mathcal{X}$ comme une séquence de formules où, pour $1 \leq i \leq n$, la formule Φ_i est sur l'ensemble de variables $\mathcal{X} \cup \{Y_1, \ldots, Y_{i-1}\}$. Le *déroulement* Ψ' de Ψ est la formule Φ'_n de la séquence $\Phi'_1, \ldots, \Phi'_n$ obtenue à partir de Ψ comme suit : on remplace successivement, pour chaque $1 \leq i \leq n$, la variable Y_i par la formule Φ_i dans chaque Φ_j pour $j > i$: ainsi, Ψ' est une formule sur $\mathcal{X}$. L'algorithme de cette question peut calculer, en temps $O(|w| \times |Q|)$, un système de formules (à savoir, pour une énumération quelconque $q_1, \ldots, q_k$ de Q, le système $\Phi_{0,q_1}, \ldots, \Phi_{0,q_k}, \Phi_{1,q_1}, \ldots, \Phi_{1,q_k}, \ldots, \Phi_{n,q_1}, \Phi_{n,q_k}$, les $\Phi_{1,q}$, etc.) dont le déroulement est une formule de provenance pour w sur A. Ce système est une représentation concise d'une formule de provenance de taille généralement exponentielle.

Une autre façon équivalente de présenter les systèmes de formules est de les voir comme un circuit, c'est-à-dire un graphe sans cycle où les nœuds sont étiquetés par les opérateurs propositionnels. Intuitivement, un circuit est défini comme une formule propositionnelle, mais a une forme de graphe sans cycle au lieu d'avoir une forme d'arbre ; il permet donc la réutilisation de sous-expressions communes.

Références

[ABS15] Antoine Amarilli, Pierre Bourhis, and Pierre Senellart. Provenance Circuits for Trees and Treelike Instances. In *ICALP*, 2015.

A6 – Clôture commutative de langages réguliers

On fixe l'alphabet $\Sigma = \{a, b\}$, et on note Σ^* l'ensemble des mots sur Σ. Pour $w \in \Sigma^*$, on note respectivement $|w|_a$ et $|w|_b$ le nombre d'occurrences de a et de b dans w. Un *langage* sur Σ^* est un sous-ensemble non nécessairement fini de Σ^*. La *clôture commutative* d'un langage L sur l'alphabet Σ est le langage $\mathrm{CCl}(L)$ des mots $w \in \Sigma^*$ tels qu'il existe $w' \in L$ avec $|w|_a = |w'|_a$ et $|w|_b = |w'|_b$.

On admettra le théorème de Kleene : un langage peut être décrit par une expression rationnelle si et seulement s'il est reconnu par un automate fini. De tels langages sont appelés *réguliers*. L'objet du problème est d'étudier la clôture commutative de langages réguliers.

Question 0. Calculer la clôture commutative du langage régulier L_0 défini par l'expression rationnelle $a^* + b^*$.

Question 1. Soit L_1 le langage régulier défini par l'expression rationnelle $b(aa)^*$. Proposer une expression rationnelle pour $\mathrm{CCl}(L_1)$. Existe-t-il un langage régulier L_1' tel que $\mathrm{CCl}(L_1') = L_1$?

Question 2. Exhiber un langage régulier L_2 tel que $\mathrm{CCl}(L_2)$ soit le langage $L_2' := \{w \in \Sigma^* \mid |w|_a = |w|_b\}$. En déduire que la clôture commutative d'un langage régulier n'est pas nécessairement un langage régulier.

Question 3. On dit qu'un langage L est *a-borné* s'il existe $k \in \mathbb{N}$ tel que, pour tout $w \in L$, on a $|w|_a \leq k$. Montrer que, pour tout langage régulier L, si L est a-borné, alors $\mathrm{CCl}(L)$ est régulier.

Étendre l'argument aux langages réguliers *ab-bornés*, c'est-à-dire les langages réguliers L où il existe $k \in \mathbb{N}$ tel que, pour tout $w \in L$, on a soit $|w|_a \leq k$, soit $|w|_b \leq k$.

Question 4. Proposer un algorithme qui décide, étant donné un langage régulier L sur Σ, si L est a-borné. Proposer également un algorithme qui décide si L est ab-borné.

Suite des questions

Question 5. Exhiber un langage régulier L qui n'est pas ab-borné mais où $\mathrm{CCl}(L)$ est régulier. Qu'en déduire par rapport à la question 3 ?

Question 6. On dit qu'un langage L est *ultimement périodique* s'il existe $t \in \mathbb{N}$ et $p \in \mathbb{N}^*$ tel que, pour tous $k \geq t$ et $l \geq t$, s'il existe un mot $w \in L$ avec $(|w|_a, |w|_b) = (k, l)$, alors il existe un mot $w' \in L$ avec $(|w'|_a, |w'|_b) = (k + p, l)$, et un mot $w'' \in L$ avec $(|w''|_a, |w''|_b) = (k, l + p)$.

Montrer que, pour tout langage régulier L, si L est ultimement périodique, alors $\mathrm{CCl}(L)$ est régulier. Comparer au résultat établi à la question 3.

Question 7. Montrer que, pour tout langage régulier L, si $\mathrm{CCl}(L)$ est régulier, alors L est ultimement périodique. Conclure.

Indication : On pourra utiliser la notion de *quotient à gauche*. Le *quotient à gauche* d'un langage L par un mot $w \in \Sigma^*$, noté $w^{-1}L$, est l'ensemble $\{w' \in \Sigma^* \mid ww' \in L\}$. On pourra utiliser, puis démontrer, l'affirmation suivante : pour tout langage régulier L', l'ensemble de langages $\{w^{-1}L' \mid w \in \Sigma^*\}$ est fini.

Corrigé

Question 0. La clôture commutative est $a^* + b^*$, c'est-à-dire le langage L_0 lui-même. Ceci est normal : le langage proposé est *commutatif*, c'est-à-dire que si un mot appartient au langage alors toutes ses permutations (ici, le mot lui-même) appartiennent au langage.

Question 1. Le langage $\mathrm{CCl}(L_1)$ est, intuitivement, le langage des mots comportant exactement un b, et un nombre pair de a. Une expression régulière pour ce langage est $(aa)^*(aba + b)(aa)^*$.

Le langage L_1 n'est pas commutatif, car on a $baa \in L_1$ mais $aab \notin L_1$. Comme la clôture commutative d'un langage est toujours un langage commutatif, il n'existe pas de langage L_1' tel que $\mathrm{CCl}(L_1') = L_1$.

Question 2. Notons L_2' le langage proposé. On considère le langage régulier $L_2 := (ab)^*$, dont il est clair que la clôture commutative est L_2'.

On peut aisément montrer à l'aide du lemme de pompage que L_2' n'est pas rationnel : si on prend $k \in \mathbb{N}$ la longueur de pompage, le mot $a^k b^k$ est dans L_2', mais le pompage conduirait à affirmer que $a^{k+d}b^k$ est dans L_2' pour un certain $d \in \mathbb{N}^*$, ce qui serait absurde.

Question 3. Pour tout langage régulier L, on appelle *a-projection* de L le langage $L_{\downarrow a} := \{a^i \mid \exists w \in L, |w|_a = i\}$. On observe d'abord que, pour tout langage régulier L, la a-projection de L est un langage régulier. En effet, si l'on considère une expression régulière pour L et que l'on remplace chaque occurrence de b par ϵ, il est clair que l'expression régulière obtenue décrit $L_{\downarrow a}$. (En d'autres termes, $L_{\downarrow a}$ est l'image de L par le morphisme $h : \Sigma \to \Sigma^*$ défini par $h(a) := a$ et $h(b) := \epsilon$, et les langages réguliers sont clos par morphisme, c'est-à-dire que l'image d'un langage régulier par un morphisme est toujours un langage régulier.)

Pour tout langage régulier L, on dit que L est un *langage sur a* si, pour tout $w \in L$, on a $|w|_b = 0$. Pour un langage L sur a, et pour $i \in \mathbb{N}$, on appelle *(b, i)-complétion* de L le langage $L_{\uparrow b,i} := \{w \in \Sigma^* \mid a^{|w|_a} \in L \wedge |w|_b = i\}$. On observe également que, pour tout langage régulier L sur a, pour tout $i \in \mathbb{N}$, le langage $L_{\uparrow b,i}$ est un langage régulier. (Le résultat est également vrai sans l'hypothèse que L est un langage sur a, mais celle-ci permet de simplifier légèrement la définition de la complétion.) En effet, si l'on considère un automate déterministe A qui reconnaît L, on peut supposer, quitte à retirer les états inutiles (non accessibles ou non coaccessibles), que A n'a pas de transitions étiquetées b. On peut ensuite construire un automate $A_{\uparrow b,i}$ qui reconnaît $L_{\uparrow b,i}$ en créant $i + 1$ copies de l'automate A étiquetées de 0 à i, en prenant comme état initial l'état initial dans la copie 0, en prenant comme états

finaux les états finaux dans la copie i, et en ajoutant, en plus des transitions originales dans chaque copie, une transition étiquetée b qui va de chaque état dans une copie $0 \leq j < n$ vers le même état dans la copie $j+1$. Il est facile de constater que $A_{\uparrow b,i}$ reconnaît bien $L_{\uparrow b,i}$, qui est donc régulier.

Prouvons à présent le résultat demandé. Soit L un langage régulier a-borné, et soit $k \in \mathbb{N}$ la borne. Comme L est a-borné, on peut l'écrire comme $\bigcup_{0 \leq i \leq k} L_i$, où $L_i := \{w \in L \mid |w|_b = i\}$. On a alors clairement $\mathrm{CCl}(L) = \bigcup_{0 \leq i \leq k} \mathrm{CCl}(L_i)$, et comme les langages réguliers sont clos par union, il suffit de montrer que, pour chaque $0 \leq i \leq k$, le langage $\mathrm{CCl}(L_i)$ est régulier.

Ainsi, fixons $0 \leq i \leq k$, et notons $L_{i\downarrow a}$ la a-projection de L_i. Comme la clôture commutative est commutative, il est facile de voir que $\mathrm{CCl}(L_i) = (\mathrm{CCl}(L_{i\downarrow a}))_{\uparrow b,i}$. Or, comme L_i est régulier, nous avons montré initialement que $L_{i\downarrow a}$ est également régulier, et c'est un langage sur a, donc commutatif. Ainsi, on a $\mathrm{CCl}(L_{i\downarrow a}) = L_{i\downarrow a}$, qui est donc un langage régulier sur a. Ainsi, $\mathrm{CCl}(L_i)$ est également régulier d'après ce que nous avons montré initialement, ce qui conclut la preuve.

Pour les langages ab-bornés, si l'on prend un tel langage L et qu'on pose k la borne, on peut clairement écrire $L = \bigcup_{0 \leq i \leq k} L_{a,i} \cup \bigcup_{0 \leq i \leq k} L_{b,i}$, où $L_{c,i} := \{w \in L \mid |w|_c = i\}$ pour $c \in \{a, b\}$. Ainsi, il suffit de montrer que, pour $c \in \{a, b\}$ et $0 \leq i \leq k$, le langage $L_{c,i}$ est régulier, ce que l'on peut faire comme précédemment.

Question 4. La question ne spécifie pas quelle représentation du langage L est fournie en entrée. On supposera que le langage L nous est fourni sous la forme d'un automate fini ; si le langage L était fourni sous la forme d'une expression rationnelle, on pourrait utiliser les méthodes vues en cours pour obtenir à partir de l'expression rationnelle un automate fini reconnaissant le même langage.

Étant donné l'automate A, on supprime d'abord les états inutiles, ce qui peut être fait en temps linéaire. À présent, il est clair que le langage $L(A)$ reconnu par A n'est *pas* a-borné si et seulement s'il existe un cycle dans A comportant une transition étiquetée a. En effet, s'il existe un tel cycle, $L(A)$ comprend des mots avec un nombre arbitraire de a, que l'on peut obtenir en parcourant le cycle un nombre arbitraire de fois. À l'inverse, s'il n'existe pas de tel cycle, le nombre de a dans un mot de $L(A)$ est borné par le nombre total de transitions étiquetées par a dans A, puisque chaque transition ne peut être franchie qu'une seule fois dans un chemin acceptant. Ainsi, il suffit de déterminer si A contient un tel cycle.

Un algorithme naïf pour cette tâche, en temps quadratique, consiste à considérer toutes les transitions étiquetées a allant d'un état q à un état q', et, pour chaque transition, déterminer s'il existe un chemin dans A allant de l'état q' à l'état q : si oui, on a trouvé un cycle, et à l'inverse, s'il y a un cycle comportant une transition étiquetée a, il est clair que le test effectué pour cette transition va réussir. Cet algorithme est en temps quadratique. On peut également résoudre le problème en temps linéaire avec un algorithme qui considère le sous-graphe induit par les transitions étiquetées b, construit en temps linéaire le graphe acyclique orienté des composantes connexes, y ajoute les arêtes induites par les transitions étiquetées a (y compris les boucles allant d'une composante à elle-même), et teste en temps linéaire si le résultat est acyclique. Si c'est le cas, alors il ne peut y avoir de cycle impliquant une transition étiquetée a, puisqu'un tel cycle impliquerait l'existence d'un cycle dans le graphe résultat (quitte à rassembler les parties du chemin qui parcourent des arêtes étiquetées b et restent dans la même composante connexe du graphe de ces transitions). À l'inverse, tout cycle dans ce graphe donne lieu à un cycle dans le graphe original, en ajoutant des chemins au sein des composantes connexes du graphe des transitions étiquetées b.

Pour le problème de décider si un langage L est ab-borné, en posant A un automate tel que $L(A) = L$, il est facile d'observer que L n'est *pas* ab-borné si et seulement si A contient un cycle C_a incluant une transition a, un cycle C_b incluant une transition b, et l'un de C_a et C_b est accessible à partir de l'autre (ce qui est en particulier le cas s'ils sont égaux). En effet, il est clair que remplir cette condition suffit à ne pas être ab-borné. À l'inverse, si L n'est pas ab-borné, pour tout $k' \in \mathbb{N}$, il faut que L contienne un mot $w_{k'}$ qui s'écrit soit $w^a_{k'} w^b_{k'}$, où $w^a_{k'}$ contient au moins k' a et $w^b_{k'}$ contient au moins k' b, soit $w^b_{k'} w^a_{k'}$ avec les mêmes conditions. Pour k' suffisamment grand, le pompage dans un tel mot permet d'extraire

un cycle contenant une transition étiquetée par a et un cycle contenant une transition étiquetée par b, où l'un des cycles est accessible à partir de l'autre (suivant le cas).

Sans chercher cette fois à optimiser le degré du polynôme, un algorithme efficace pour cette tâche consiste à tester d'abord s'il y a un cycle contenant une transition a et une transition b, en essayant toutes les paires de telles transitions possibles, et en vérifiant l'accessibilité de l'extrémité de chaque transition depuis l'extrémité opposée de l'autre transition (via des transitions d'étiquette quelconque). À défaut, on teste s'il y a deux cycles différents, l'un contenant une transition a et l'autre une transition b, et l'un accessible à partir de l'autre, en essayant toutes les paires de telles transitions, en vérifiant l'accessibilité de l'extrémité de chaque transition depuis son autre extrémité (via des transitions d'étiquette quelconque), puis en vérifiant l'accessibilité d'une extrémité quelconque d'une des deux transitions depuis une extrémité quelconque de l'autre transition (via des transitions d'étiquette quelconque).

Question 5. Le langage régulier $(a+b)^*$ de tous les mots sur Σ^* est commutatif, donc égal à sa propre clôture commutative, qui est donc régulière ; pourtant $(a+b)^*$ n'est clairement pas ab-borné.

Si on veut un exemple moins trivial, on peut prendre $(aa+bb)^*$, ou encore $(aa)^*(bb)^*$. En tout cas, par rapport à la question 3, être ab-borné n'est manifestement pas une condition nécessaire pour avoir une clôture commutative régulière.

Question 6. Pour $i, j \in \mathbb{N}$, on appelle $S_{i,j}$ le sous-ensemble de $\{0, \dots, p-1\}^2$ défini par $(k,l) \in S_{i,j}$ s'il existe $w \in L$ tel que $|w|_a = t + ip + k$ et $|w|_b = t + jp + l$. Comme L est ultimement périodique, il est clair que $S_{i,j} \subseteq S_{i+1,j+1}$ pour tout $i \in \mathbb{N}$. Ainsi, la suite $S_{i,j}$ d'ensembles est croissante au sens de l'inclusion, et elle est bornée par $\{0, \dots, p-1\}^2$, donc il existe un rang r à partir duquel elle est constante, c'est-à-dire $S_{r',r'} = S_{r,r}$ pour tout $r' \geq r$. Par définition de l'ultime périodicité, il est clair que ceci implique en fait que $S_{r',r''} = S_{r,r}$ pour tous $r', r'' \geq r$, car si on avait $S_{r,r} \subsetneq S_{r',r''}$ pour $r', r'' \geq r$, on en déduirait par ultime périodicité que $S_{r,r} \subsetneq S_{\max\{r',r''\},\max\{r',r''\}}$, une contradiction. On appelle $t' := t + rp$.

Notons d'abord qu'on peut écrire $L = L_{<t'} \cup L_{\geq t'}$, où $L_{<t'}$ est $\{w \in L \mid |w|_a < t' \vee |w|_b < t'\}$ et $L_{\geq t'}$ est $\{w \in L \mid |w|_a \geq t' \wedge |w|_b \geq t'\}$. Il est clair que $L_{<t'}$ et $L_{\geq t'}$ sont réguliers : en effet, on peut les écrire comme l'intersection de L avec des langages réguliers. Du reste, $L_{<t'}$ est clairement ab-borné, donc $\mathrm{CCl}(L_{<t'})$ est régulier d'après la question 3. Ainsi, comme les langages réguliers sont clos par union, il suffit de montrer que $\mathrm{CCl}(L_{\geq t'})$ est régulier. Écrivons $L_{\geq t'}$ comme l'union des $L_{k,l}$ pour $(k,l) \in \{0, \dots, p-1\}^2$, chacun étant défini comme l'intersection de $L_{\geq t'}$ avec le langage régulier $\{w \in \Sigma^* \mid (|w|_a - t') \bmod p = k \wedge (|w|_b - t') \bmod p = l\}$. Il suffit de montrer que, pour tout $(k,l) \in \{0, \dots, p-1\}^2$, le langage $\mathrm{CCl}(L_{k,l})$ est régulier. Fixons $(k,l) \in \{0, \dots, p-1\}^2$. Si le langage $L_{k,l}$ est vide, alors $\mathrm{CCl}(L_{k,l})$ est clairement régulier. Sinon, s'il est non-vide, par définition de t', ceci permet de savoir que $(k,l) \in S_{r',r''}$ pour un certain choix de $r'' \geq r$ et $r' \geq r$. Ainsi, d'après la reformulation de l'ultime périodicité ci-dessus, on sait que $L_{k,l}$ contient, pour tous $i, j \in \mathbb{N}$, un mot w tel que $|w|_a = t' + ip + k$ et $|w|_b = t' + jp + l$. À l'inverse, par définition de $L_{k,l}$, pour tout $w \in L_{k,l}$, on a $(|w|_a - t') \bmod p = k$ et $(|w|_b - t') \bmod p = l$, et $|w|_a \geq t'$ et $|w|_b \geq t'$. On a ainsi montré que $\mathrm{CCl}(L_{k,l})$ est le langage des mots $w \in \Sigma^*$ contenant un nombre de a plus grand que t' et dont la différence avec t' est congrue à k modulo p, et un nombre de b plus grand que t' et dont la différence avec t' est congrue à l modulo p. Ce langage est manifestement régulier, ce qui conclut la preuve.

Notons que le résultat de cette question généralise celui de la question 3 : en effet, il est clair que tout langage ab-borné est ultimement périodique, car on peut prendre t strictement plus grand que la borne, et p arbitraire.

Question 7. On utilise d'abord l'affirmation proposée dans l'énoncé, sans la démontrer.

Fixons le langage régulier L. On définit la fonction f qui associe à $(i,j) \in \mathbb{N}$ le quotient à gauche $(a^i b^j)^{-1}\mathrm{CCl}(L)$ de $\mathrm{CCl}(L)$. Comme $\mathrm{CCl}(L)$ est régulier, d'après l'affirmation, l'image de f est un

ensemble fini.

On montre à présent un autre lemme : pour tout ensemble fini non-vide X, pour toute fonction g de $\mathbb{N}^2$ dans X, il existe $i, j \in \mathbb{N}$ et $d_i, d_j \in \mathbb{N}^*$ tels que $g(i,j) = g(i+d_i, j) = g(i, j+d_j)$. On montre ce résultat par récurrence sur le cardinal de X. Pour le cas de base, si $|X| = 1$, on peut prendre $i := 0$, $j := 0$, $d_i := 1$, $d_j := 1$. Montrons à présent le cas de récurrence. Soit $n \in \mathbb{N}$, supposons le résultat acquis pour n, et démontrons-le pour $n+1$. Soit X un ensemble tel que $|X| = n+1$, et soit g une fonction de $\mathbb{N}^2$ dans X. Considérons la suite $(g(i,0))_{i\geq 0}$ d'éléments de X. Comme cette suite est infinie et que X est fini, il doit y avoir un élément $x \in X$ qui apparaît infiniment souvent dans cette suite, et on peut ainsi choisir une suite strictement croissante $p_0 < \cdots < p_n < \cdots$ de $\mathbb{N}$ de sorte que la suite extraite $(g(p_i, 0))_{i\geq 0}$ soit la suite constante dont tous les termes valent x. Considérons à présent la fonction g' de $\mathbb{N}^2$ dans X définie par $g'(i,j) := g(p_i, j+1)$ pour tout $(i,j) \in \mathbb{N}^2$. Soit il existe $i', j' \in \mathbb{N}$ tel que $g'(i', j') = x$, soit il n'existe pas de tels $i', j' \in \mathbb{N}$.

Dans le premier cas, s'il existe $i', j' \in \mathbb{N}$ tels que $g'(i', j') = x$, alors on a $g(p_{i'}, j'+1) = x$, et on peut ainsi prendre $i := p_{i'}$, $j := 0$, $d_i := p_{i'+1} - p_{i'}$, et $d_j := j'+1$, et obtenir le résultat, car on a $g(i,j) = g(p_{i'}, 0) = x$, on a $g(i+d_i, j) = g(p_{i'} + (p_{i'+1} - p_{i'}), 0) = g(p_{i'+1}, 0) = x$, et on a $g(i, j+d_j) = g(p_{i'}, j'+1) = x$.

Dans le second cas, s'il n'existe pas de tels $i', j' \in \mathbb{N}$, alors g' est en fait une fonction de $\mathbb{N}^2$ dans $X \setminus \{x\}$. Comme cet ensemble a cardinalité n, par application de l'hypothèse de récurrence, il existe $i', j' \in \mathbb{N}$, $d'_i, d'_j \in \mathbb{N}^*$, tels que $g'(i', j') = g'(i'+d'_i, j') = g'(i', j'+d'_j)$. Par définition de g', on peut ainsi déduire i, j, d_i, d_j satisfaisant les mêmes égalités pour g. Ceci établit le cas d'induction, et conclut la preuve du lemme.

Ainsi, en utilisant le lemme pour la fonction f définie plus haut, et pour l'ensemble fini non-vide des quotients à gauche de $\mathrm{CCl}(L)$, on en conclut qu'il existe $i, j \in \mathbb{N}$, $d_i, d_j \in \mathbb{N}^*$, tels que $f(i,j) = f(i+d_i, j) = f(i, j+d_j)$. Posons $t := \max\{i, j\}$, et $p := d_i \times d_j$. Montrons que L est ultimement périodique pour t et p. Soit $k, l \geq t$, et supposons qu'il existe un mot $w \in \mathbb{N}$ avec $(|w|_a, |w|_b) = (k, l)$. Comme la clôture commutative est commutative, ceci implique que $a^i b^j a^{k-i} b^{l-j} \in \mathrm{CCl}(L)$ (ceci est bien défini car $k \geq i$ et $l \geq j$). Ainsi, $a^{k-i} b^{l-j} \in f(i,j)$. À présent, comme $f(i,j) = f(i+d_i, j)$, on a également $a^{k-i} b^{l-j} \in f(i+d_i, j)$, d'où $a^{k-i+d_i} b^{l-j} \in f(i,j)$. En appliquant d_j fois ce procédé, on déduit que $a^{k-i+p} b^{l-j} \in f(i,j)$, de sorte que $a^i b^j a^{k-i+p} b^{l-j} \in \mathrm{CCl}(L)$. Ainsi, il existe un mot $w' \in L$ avec $(|w'|_a, |w'|_b) = (k+p, l)$. On peut montrer d'une manière analogue qu'il existe un mot $w' \in L$ avec $(|w'|_a, |w'|_b) = (k, l+p)$. Ainsi, L est bien ultimement périodique.

Il ne reste plus qu'à montrer l'affirmation supplémentaire donnée dans l'énoncé. Soit L un langage régulier, et montrons que L a un nombre fini de quotients à gauche. Soit A un automate déterministe fini *complet* qui reconnaît L, et soit Q son ensemble d'états. Pour tout mot $w \in \Sigma^*$, on note $\phi(w)$ l'état de Q auquel on aboutit dans l'automate A en lisant w. Il est facile de voir que, pour tout mot $w \in \Sigma^*$, le quotient à gauche $w^{-1}L$ est entièrement déterminé par $\phi(w)$: ce quotient $w^{-1}L$ est l'ensemble des mots w' tels qu'il existe un chemin dans A étiqueté par w' qui va de $\phi(w)$ à un état final. Ainsi, il y a au plus $|Q|$ quotients à gauche, ce qui conclut la preuve.

A7 – Allocation de candidats à des écoles

Dans ce problème, on considère un ensemble fini d'*écoles d'ingénieur* $\mathcal{E} = E_1, \ldots, E_n$, et un ensemble fini de *candidats* $\mathcal{C}$. Pour $1 \leq i \leq n$, chaque école a un ensemble $A_i \subseteq \mathcal{C}$ de candidats *sur liste principale*, un ordre total $<_i$ sur A_i (qui trie les candidats du meilleur au moins bon pour cet école), un ensemble $A'_i \subseteq \mathcal{C}$ disjoint de A_i de candidats *sur liste complémentaire*, et un ordre total $<'_i$ sur A_i. On s'intéresse à un candidat $c \in \mathcal{C}$, qui accepterait d'intégrer n'importe laquelle des écoles de $\mathcal{E}$. On cherche à déterminer s'il est certain que c pourra être admis dans une école, quels que soient les choix des autres candidats, sachant que :

— chaque candidat ne peut accepter qu'une seule école ;
— chaque école ne peut admettre que des candidats figurant sur l'une de ses listes ;
— chaque école E_i peut admettre un nombre de candidats égal au plus à la longueur $|A_i|$ de sa liste principale ;
— chaque école admet en priorité les candidats les mieux classés dans ses listes principale et complémentaire.

Question 0. Donner un exemple de situation où c ne figure sur aucune liste principale mais où il est certain que c pourra être admis dans une école.

Question 1. En simplifiant les données introduites dans l'énoncé, proposer une formalisation de ce problème.

Question 2. On suppose qu'il n'y a que deux écoles, c'est-à-dire $n = 2$. Proposer un algorithme simple pour résoudre efficacement le problème.

Question 3. On suppose à présent que chaque candidat excepté c apparaît sur la liste de deux écoles au plus, et que que chaque école peut admettre exactement un candidat.

Expliquer comment on peut construire efficacement une formule de la logique propositionnelle qui est satisfiable si et seulement s'il est possible que c ne soit admis dans aucune école. On construira une formule en *forme normale conjonctive* (CNF), c'est-à-dire une conjonction de *clauses*, où chaque clause est une disjonction de littéraux.

Question 4. Soit $F = C_1 \wedge \cdots \wedge C_n \wedge (x \vee C) \wedge (\neg x \vee C')$ une formule en CNF, où les occurrences de la variable x sont exactement comme indiqué. Montrer que F est satisfiable si et seulement si la formule $C_1 \wedge \cdots \wedge C_n \wedge (C \vee C')$ est satisfiable.

Question 5. En observant la forme de la CNF obtenue en question 3, déduire un algorithme pour résoudre le problème sous les hypothèses de la question 3. Discuter de l'efficacité de cet algorithme.

Suite des questions

Question 6. On considère un graphe non-orienté $G = (U, E)$, où U est un ensemble de sommets et E est un ensemble de paires d'éléments de U. Une *orientation* de G est un graphe orienté $G' = (U, E')$ tel que, pour toute paire $\{u, v\} \in E$, un et un seul des couples (u, v) et (v, u) est dans E'. Une orientation G' de G est *répartie* si, pour tout $u \in U$, il existe $v \in U$ tel que $(v, u) \in E'$.

Proposer une caractérisation des graphes non-orientés pour lesquels il existe une orientation répartie, et un algorithme pour les reconnaître.

En déduire un autre algorithme pour résoudre le problème sous les hypothèses de la question 3. Discuter de l'efficacité de cet algorithme.

Question 7. Un *graphe biparti* est un graphe non-orienté $G = (U, E)$ où U est partitionné en deux ensembles disjoints $U = V \sqcup W$ de sommets, et où les arêtes de E sont des paires formées d'un sommet de V et d'un sommet de W.

Deux arêtes $e, e' \in E$ sont *indépendantes* si ce sont des paires disjointes. Un *couplage* de G est un sous-ensemble $E' \subseteq E$ d'arêtes qui sont deux à deux indépendantes.

Expliquer comment on peut construire, à partir des données du problème, un graphe biparti G et un entier $\ell \in \mathbb{N}$ tel que G admet un couplage de cardinalité $\geq \ell$ si et seulement s'il est possible que c ne soit admis dans aucune école. (On ne fait plus ici les hypothèses de la question 3.)

Corrigé

Question 0. Prendre $n := 2$, $\mathcal{C} := \{1, 2\}$, $c := 2$, $A_1 = A_2 = \{1\}$, $A'_1 = A'_2 = \{2\}$. Il est clair que, comme le candidat 1 ne peut intégrer qu'une seule école de $\mathcal{E}$, le candidat 2 pourra intégrer l'autre.

Question 1. On peut d'abord supposer que c n'est sur la liste principale d'aucune école ; en effet, si c est sur liste principale dans une école, alors il est certain de pouvoir avoir cette école, et le problème est trivial.

On peut ensuite éliminer de $\mathcal{E}$ les écoles où c n'apparaît pas sur la liste complémentaire : en effet, c ne pourra jamais être admis dans cette école, et le fait que d'autres candidats puissent y être admis (parmi ceux qui apparaissent sur les mêmes listes que c) n'est pas pertinent si on s'intéresse au pire cas.

On peut ensuite tronquer les listes complémentaires pour en retirer tous les candidats qui y sont classés après c. En effet, le fait que ces candidats apparaissent après c signifie qu'ils ne pourront pas l'empêcher d'être admis dans cette école, et là encore la possibilité qu'ils le soient n'est pas pertinente pour le pire cas.

On peut à présent modéliser le processus d'affectation de la manière suivante : tous les candidats restants font ensemble un choix d'école parmi les écoles où ils sont sur l'une des listes. Leur objectif, pour empêcher que c puisse être admis dans une des écoles, est d'assurer que, pour chaque école $1 \leq i \leq n$, si on considère l'ensemble des candidats autres que c (qui sont classés devant lui, sur liste principale ou sur liste complémentaire), alors le nombre de candidats qui a choisi E_i est égal au moins à la taille de la liste principale de l'école E_i.

On s'aperçoit alors que, pour chaque école, l'ordre relatif des candidats devant c est en fait importance, et on arrive à la modélisation simplifiée du problème que voici : pour chaque école E_i pour $1 \leq i \leq n$, on a un nombre $a_i \in \mathbb{N}$ de places (défini comme $a_i := |A_i|$) et un ensemble $S_i \subseteq \mathcal{C} \setminus \{c\}$ de candidats adverses classés devant c pour l'école i. Une *affectation* est une fonction f de $\mathcal{C} \setminus \{c\}$ vers $\{1, \ldots, n\}$ telle que, pour tout $c' \in \mathcal{C} \setminus \{c\}$, on a $c' \in S_{f(s)}$. Une affectation est *bloquante* si, pour tout $1 \leq i \leq n$, on a $|\{s \in \mathcal{C} \setminus \{c\} \mid f(s) = i\}| \geq a_i$.

On cherche donc à déterminer, étant donné $S_1, \ldots, S_n$ et $a_1, \ldots, a_n$, s'il existe une affectation bloquante.

Question 2. Si $n = 2$, alors fixons $(S_1, a_1), (S_2, a_2)$. On commence par retirer de S_1 les éléments qui n'apparaissent pas dans S_2, car on peut supposer que ces candidats vont dans l'école 1 ; on diminue a_1 en conséquence. On fait de même pour S_2. À la fin de ce processus, on a $S_1 = S_2$, et il est clair que les autres candidats peuvent empêcher c d'être admis si et seulement si $|S_1| \geq a_1 + a_2$.

Question 3. On observe tout d'abord que, si un candidat adverse apparaît dans un seul S_i, alors la solution la plus défavorable est toujours que ce candidat accepte cette école, et on peut éliminer l'école correspondante puisqu'elle a une seule place. Ainsi, on peut supposer sans perte de généralité que chaque candidat adverse apparaît dans exactement deux S_i.

Pour exprimer que chaque candidat ne peut choisir qu'un seul S_i parmi les deux, on construit une variable x_i pour chaque candidat adverse $i \in \mathcal{C} \setminus \{c\}$. L'intuition est que cette variable est vraie si i accepte la première école où il apparaît devant c (dans l'ordre sur les écoles $E_1 < \cdots < E_n$), et fausse sinon.

On va construire une formule en CNF à n clauses, où la clause $1 \leq i \leq n$ signifiera intuitivement "c ne peut pas être admis dans l'école E_i".

Comme chaque école admet un seul candidat, la clause i sera une disjonction exprimant qu'un des candidats de S_i choisit l'école i. Spécifiquement, pour chaque candidat dans S_i, on ajoute à la clause le littéral x_i si S_i est la première école où i apparaît, et $\neg x_i$ sinon.

Montrons que la formule obtenue satisfait les conditions. S'il existe une allocation bloquante, on considère la valuation qui rend x_i vraie ou fausse selon que le candidat i a accepté la première ou la deuxième école ; dans l'éventualité sans intérêt où le candidat i n'aurait accepté aucune école, on choisit une valeur arbitraire pour x_i. Si c n'est admis nulle part, ceci signifie que, pour chaque école, il y a un des candidats devant S_i qui a accepté, donc l'un des littéraux correspondants est vrai, ainsi la formule est vraie.

À l'inverse, à partir d'une valuation qui satisfait la formule, on construit une allocation bloquante comme suit : pour chaque école, on choisit un candidat de S_i dont le littéral est vrai dans la i-ème clause. Il faut vérifier que ceci ne choisit pas deux fois le même candidat, mais c'est le cas car on aurait alors choisi les deux littéraux correspondant au candidat, mais ils ne peuvent pas être vrai simultanément par définition d'une valuation puisqu'ils sont de polarités opposées.

Question 4. On va montrer que, pour toute variable x, clauses disjonctives C, C' où x n'apparaît pas, et formule Φ où x n'apparaît pas, alors $\Phi \wedge (x \vee C) \wedge (\neg x \vee C')$ est satisfiable si et seulement si $\Phi \wedge (C \vee C')$ est satisfiable.

Supposons que $\Phi \wedge (x \vee C) \wedge (\neg x \vee C')$ soit satisfiable, et soit ν la valuation correspondante. Si x est vrai dans ν, alors comme ν satisfait $\neg x \vee C'$, on sait que ν satisfait C', donc ν satisfait $C \vee C'$, et ν satisfait $\Phi \wedge (C \vee C')$ car ν satisfait Φ. Si x est faux, alors ν satisfait $x \vee C$, donc ν satisfait C, donc ν satisfait $C \vee C'$ et également $\Phi \wedge (C \vee C')$. Ainsi, dans tous les cas, ν satisfait $\Phi \wedge (C \vee C')$, qui est donc satisfiable.

À l'inverse, supposons que $\Phi \wedge (C \vee C')$ soit satisfiable, et soit ν la valuation correspondante. Soit ν satisfait C, soit ν satisfait C'. Si ν satisfait C, alors soit ν' la valuation définie comme ν mais qui rend également x faux ; ν' satisfait donc $x \vee C$ car ν satisfait C, et ν' satisfait $\neg x$ donc elle satisfait $\neg x \vee C'$; ainsi, comme au demeurant ν satisfait Φ, on sait que ν' satisfait $\Phi \wedge (x \vee C) \wedge (\neg x \vee C')$, ce qui montre que cette formule est satisfiable. Si ν satisfait C', on procède exactement de la même manière, mais on rend x vrai dans ν'.

[La procédure présentée dans cette question s'appelle la résolution *; voir par exemple [Wik17b].]*

Question 5. On va déterminer s'il existe une allocation bloquante en déterminant si la formule de la question 3 est satisfiable. À cette fin, on observe que la formule construite à la question 3 a la propriété que, pour chaque variable x, chacun des littéraux x et $\neg x$ apparaissent exactement une fois dans la formule, et dans deux clauses différentes.

On observe également que, si la formule $C_1 \wedge \cdots \wedge C_n \wedge (x \vee C) \wedge (\neg x \vee C')$ a cette propriété, alors c'est clairement toujours le cas de $C_1 \wedge \cdots \wedge C_n \wedge (C \vee C')$; à moins que $C \vee C'$ contienne les deux littéraux d'une même variable. Mais dans ce cas, la clause est tautologique et on peut récrire de façon équivalente en $C_1 \wedge \cdots \wedge C_n$ qui satisfait la condition.

On peut ainsi décider si la formule de la question 3 est satisfiable par applications répétées de la procédure de la question 4 : cette procédure est toujours applicable parce qu'on peut choisir n'importe quelle variable x et, comme x et $\neg x$ apparaissent exactement une fois dans une seule clause, on se ramène à la forme de la question 3 en utilisant la commutativité de $\wedge$ et $\vee$, et on applique la procédure de la question 4, en éliminant la clause résultante si elle est tautologique. À chaque étape du processus, le nombre de clauses décroît au moins de 1, donc on peut appliquer le processus et progresser tant que les clauses ne sont pas vides. Le résultat de la récriture est équivalent et ne contient plus de variables, et il suffit alors de vérifier si c'est la conjonction vide (tautologique donc satisfiable), ou une conjonction non-vide de disjonctions vides (contradictoire donc insatisfiable), ce qui permet de conclure.

Une implantation naïve de cette procédure, à chaque étape, choisit une variable et trouve le littéral opposé en temps linéaire, effectue ensuite la récriture et teste si la clause résultante est tautologique en temps linéaire. Le nombre d'étapes est linéaire, ainsi la complexité est quadratique en la formule initiale, c'est-à-dire en $O(n^2)$. On peut rendre cette complexité linéaire avec des structures de données adéquates, mais on peut aussi utiliser l'algorithme linéaire de la question suivante.

Question 6. Il est clair qu'un graphe non-orienté G a une orientation répartie si et seulement si c'est le cas de toutes ses composantes connexes, dont on suppose que le graphe G est connexe. On va montrer qu'un graphe connexe non-orienté G n'admet pas d'orientation répartie si et seulement si G est acyclique, c'est-à-dire si et seulement si c'est un arbre non-orienté.

Si G est acyclique, alors on constate facilement que G n'admet pas d'orientation répartie. En effet, si on enracine G en un sommet arbitraire, on constate qu'une orientation répartie doit nécessairement orienter vers le bas l'unique arête incidente aux feuilles, et de même pour les nœuds internes (par récurrence sur la profondeur, des plus profonds aux moins profonds), de sorte que toutes les arêtes adjacentes à la racine sont nécessairement orientées vers le bas, et celle-ci n'a donc pas d'arête entrante et ne respecte pas la condition.

À l'inverse, si G a un cycle, alors on montre qu'il admet une orientation répartie. On choisit un cycle simple arbitraire de G, et on oriente les arêtes de ce cycle suivant le cycle dans un sens arbitraire ; ceci assure que toutes les arêtes du cycle satisfont la condition. On enlève les arêtes du cycle, et on considère un parcours en largeur du graphe non-orienté G' résultant, qui part des sommets du cycle. On observe que ce parcours en largeur va pouvoir visiter tous les sommets de G'. En effet, le graphe G initial était connexe, donc pour tout sommet de G il existait un chemin à partir d'un sommet du cycle qui n'empruntait pas d'arête du cycle, comme on peut s'en rendre compte en prenant un chemin témoin simple arbitraire à partir d'un sommet quelconque du cycle, et en considérant son suffixe qui n'emprunte plus d'arête du cycle, et part donc d'un sommet du cycle. Au cours de ce parcours en largeur, on oriente toutes les arêtes empruntées dans le sens du parcours ; ceci permet de garantir que tous les sommets de G' qui ne sont pas dans le cycle ont une arête entrante, et satisfont la condition. Ainsi, en orientant les éventuelles arêtes restantes d'une manière arbitraire, on obtient une orientation répartie de G.

Ainsi, pour un graphe arbitraire G, on a montré que G admettait une orientation répartie si et seulement si chacune de ses composantes connexes avait un cycle. On peut donc reconnaître en temps linéaire les graphes admettant une orientation répartie : il suffit de calculer les composantes connexes, et de vérifier si elles sont acycliques (ce qui peut se faire simplement en comptant les arêtes, car un graphe connexe à n sommets est acyclique si et seulement s'il a $n-1$ arêtes). On peut également calculer cette orientation répartie en temps linéaire si elle existe, comme expliqué dans le paragraphe précédent.

On explique à présent comment on peut réduire notre problème en temps linéaire à un test d'existence d'une orientation répartie d'un graphe non-orienté G. On simplifie au préalable pour éliminer les

candidats qui n'apparaissent que dans une seule liste, comme à la question 3 : ceci peut être fait en temps linéaire en comptant le nombre d'occurrences de chaque candidat en une première passe et en éliminant les écoles contenant un candidat avec une seule occurrence en une seconde passe. Le graphe G est ensuite obtenu en mettant un sommet par école, et une arête par candidat qui relie les deux écoles où le candidat apparaît. On peut construire ce graphe en temps linéaire en parcourant les listes (après simplification) et en mémorisant, pour chaque candidat, dans un tableau associatif, la première école où on l'a vu ; et en construisant l'arête pour le candidat lorsque l'on voit la deuxième occurrence.

On voit à présent qu'un choix d'affectation pour chaque candidat revient à orienter l'arête du candidat, la direction indiquant l'école choisie. Une école est inaccessible au candidat c si elle a une arête entrante. Ainsi, on peut s'assurer que c n'est admis dans aucune école si et seulement le graphe G a une orientation répartie. On sait à présent que l'on peut effectuer ce test en temps linéaire.

Question 7. On construit un graphe biparti $(V \sqcup W, E)$ en créant un sommet dans V par candidat adverse $j \in \mathcal{C} \setminus \{c\}$, en créant un sommet dans W par *place* dans une école $\{(i,p) \mid 1 \leq i \leq n, 1 \leq p \leq a_i\}$, et en mettant pour chaque $1 \leq i \leq n$, pour chaque $j \in S_i$, une arête de $j \in V$ à (i,p) pour chaque $1 \leq p \leq a_i$. On choisit $\ell := \sum_i a_i$, c'est-à-dire $\ell := |W|$. On observe de ce fait qu'un couplage biparti ne peut donc pas être de taille strictement supérieure à ℓ.

Montrons que, s'il existe une allocation bloquante, alors il existe un couplage de taille $\geq \ell$. Étant donné une affectation, on numérote de façon arbitraire les a_i places occupées par les candidats adverses admis dans chaque école E_i pour $1 \leq i \leq n$, et on choisit pour chaque candidat $j \in \mathcal{C} \setminus \{c\}$ l'arête entre j et (i,p), où i est l'école choisie par j, et p est la place choisie par j dans cette numérotation ; si le candidat j ne prend aucune école, alors on ne prend aucune arête incidente à j. Le résultat est un couplage, parce que l'on choisit au plus une arête par sommet de V, et qu'on ne peut pas prendre deux arêtes différentes incidentes à (i,p) pour $1 \leq i \leq n$ et $1 \leq p \leq a_i$: ceci signifierait que la même place est occupée par deux candidats différents. De plus, c'est un couplage de taille ℓ, car la définition d'une affectation garantit que toutes les places sont occupées, donc tous les sommets de W ont une arête incidente.

À l'inverse, à partir d'un couplage de taille $\geq \ell$ de G, on déduit à partir des arêtes retenues un choix de ℓ candidats qui assurent qu'il y ait a_i places occupées dans chaque école $1 \leq i \leq n$, de sorte que chaque place de l'école $1 \leq i \leq n$ soit occupée par un candidat adverse qui apparaît dans S_i, et de sorte qu'on n'utilise pas deux fois le même candidat. Ceci conclut la preuve de l'équivalence.

[À partir de la représentation de ce problème comme un graphe biparti, on peut calculer en temps polynomial s'il existe une solution, voir par exemple [Wik17a]. Cette formulation du problème est inspirée de [Ama14].]

Références

[Ama14] Antoine Amarilli. A cute allocation problem, 2014. `https://a3nm.net/blog/allocation.html`.

[Wik17a] Wikipedia. Matching (graph theory)#in_unweighted_bipartite_graphs, 2017. `https://en.wikipedia.org/wiki/Matching_(graph_theory)#In_unweighted_bipartite_graphs`.

[Wik17b] Wikipedia. Resolution (logic), 2017. `https://en.wikipedia.org/wiki/Resolution_%28logic%29`.

A8 – Automates à pile et lemme de pompage

Soit Σ un alphabet fini, et Γ un autre alphabet fini appelé *alphabet de pile*. Un *automate à pile* sur Σ est un quintuplet $A = (Q, q_0, \gamma_0, \delta, F)$, où Q est un ensemble fini d'*états*, $q_0 \in Q$ est l'*état initial*, $\gamma_0 \in \Gamma$ est le *symbole de pile initial*, δ est une *fonction de transition* de $Q \times \Sigma \times \Gamma$ vers l'ensemble des parties de $Q \times \Gamma^*$, et $F \subseteq Q$ est un ensemble d'états *finaux*.

Une *configuration* de A est un couple (q, z) où $q \in Q$ est l'*état* et où z est un mot non-vide sur Γ appelé *pile*. La *configuration initiale* est (q_0, γ_0), et une configuration (q, z) est *acceptante* si $q \in F$. Lorsque l'automate est dans une configuration (q, z) et lit une lettre $a \in \Sigma$, il décompose $z = z'\gamma$ avec $\gamma \in \Gamma$ le *sommet de pile*, il choisit un $(q', g) \in \delta(q, a_{i+1}, \gamma)$ tel que $z'g$ soit non-vide, et il aboutit à la configuration $(q', z'g)$. L'automate à pile A *accepte* un mot de Σ^* s'il peut lire ses lettres dans l'ordre à partir de la configuration initiale pour parvenir à une configuration acceptante suivant ces règles.

Question 0. Étant donné un automate A sur Σ sans pile, expliquer comment construire un automate à pile A' sur Σ qui reconnaît le même langage que A.

Question 1. On prend dans cette question $\Sigma = \{a, b\}$. Proposer un automate à pile qui reconnaît le langage $\{a^n b^n \mid n \geq 2\}$. Qu'en déduire ?

Question 2. On prend toujours $\Sigma = \{a, b\}$. Proposer un automate à pile qui reconnaît le langage $\{w \in \Sigma^* \mid |w|_a = |w|_b\}$, où $|w|_a$ et $|w|_b$ désignent respectivement le nombre de a et de b de w.

Suite des questions

Question 3. Pour $\eta \in \mathbb{N}^*$, étant donné un mot w de longueur n et un automate à pile A, les *configurations η-tronquées* sont les triplets (q, z) où $q \in Q$ et z est un mot non-vide sur Γ de longueur $\leq \eta$. Un *calcul η-tronqué* de A sur un mot est une séquence de configurations η-tronquées : la définition est comme auparavant sauf que, si $|z'g| \geq \eta$, on le remplace par son suffixe de longueur η. Le *langage η-tronqué* $L_{\leq\eta}(A)$ de A est le langage des mots sur lesquels A a un calcul η-tronqué acceptant.

Quelle est la relation entre $L_{\leq\eta}(A)$ et $L(A)$? Montrer que, pour tout automate à pile A et $\eta \in \mathbb{N}^*$, le langage η-tronqué de A est un langage régulier, et expliquer comment construire un automate sans pile qui le reconnaît.

Question 4. Soit w un mot de longueur n et A un automate à pile. Soit $\chi = (q_0, z_0), \ldots, (q_p, z_p)$ un calcul de A sur w, et notons $h_i := |z_i|$ pour tout $0 \leq i \leq p$. Pour $\eta \in \mathbb{N}^*$, une *η-montagne* de χ est un triplet d'indices $0 \leq l < m < r \leq p$ tels qu'on ait $h_l = h_r$, on ait $h_m - h_l \geq \eta$, et on ait $h_j > h_l$ pour tout $l < j < r$. Une *montagne* est une η-montagne pour un certain $\eta \in \mathbb{N}^*$.

On appelle G la plus grande longueur d'un mot de Γ^* dans une image de la fonction de transition δ. Montrer que, pour tout $\eta > G$, si $w \in L(A) \setminus L_{\leq\eta}(A)$, alors tout calcul acceptant de A sur w a une $(\eta - G)$-montagne.

Question 5. L'*état de base* d'une η-montagne $l < r$ dans un calcul est le triplet $\beta(l, r) := (q_l, q_r, \gamma)$ où q_l, q_r sont les états des étapes l et r du calcul respectivement, et γ est le dernier symbole de la pile z_l.

Montrer que, pour tout automate à pile A, il existe $\eta_0 \in \mathbb{N}^*$ avec la propriété suivante : pour tout mot w, pour tout calcul χ de A sur w, si χ a une η_0-montagne (l, m, r), alors il existe une montagne (l', m, r') et une montagne (l'', m, r'') avec $l < l' < l'' < m < r'' < r' < r$ telles que $\beta(l', r') = \beta(l'', r'')$.

Question 6. Déduire des trois questions précédentes une forme affaiblie du *lemme de pompage pour les automates à pile* : pour tout langage L reconnu par un automate à pile, il existe un entier $p \in \mathbb{N}$ tel que, pour tout mot $w \in L$ avec $|w| > p$, on peut écrire $w = uvxyz$ où les mots u, v, x, y, z satisfont :

— $|vy| \geq 1$

— Pour tout $n \in \mathbb{N}$, on a $uv^n xy^n z \in L$.

Corrigé

Question 0. On choisit un alphabet de pile Γ arbitraire, et $\gamma_0 \in \Gamma$ arbitraire. Soit $A = (Q, q_0, F, \delta)$ un automate sans pile sur Σ, non nécessairement complet ou déterministe, où $\delta : Q \times \Sigma \to 2^Q$. On définit l'automate $A' = (Q, q_0, \gamma_0, \delta', F)$, où on définit $\delta(q, a, \gamma) := \{(q', \gamma_0) \mid q' \in \delta(q)\}$ pour tous $q \in Q$, $a \in \Sigma$, et $\gamma \in \Gamma$.

Observons d'abord que, dans toutes les configurations accessibles pour l'automate A', la *pile* est exactement γ_0. En effet, à chaque transition, on la laisse à γ_0. De plus, la valeur du sommet de pile n'a jamais d'impact sur les transitions. Ainsi, les séquences de configurations possibles de A' sur n'importe quel mot $w \in \Sigma^*$ correspondent exactement aux séquences d'états possibles de A sur w en ajoutant une pile qui vaut toujours γ_0. En particulier, la configuration initiale correspond à l'état initial, les configurations acceptantes correspondent aux états finaux, et les transitions entre configurations correspondent aux transitions de A. Ainsi, A et A' reconnaissent le même langage.

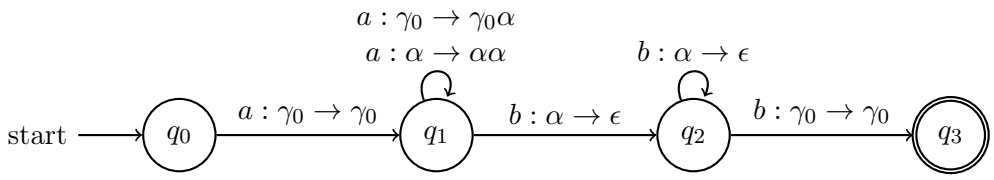
Question 1. On choisit l'alphabet de pile $\Gamma = \{\alpha, \gamma_0\}$. On définit l'automate $A = (\{q_0, q_1, q_2, q_3\}, q_0, \gamma_0, \delta, \{q_3\})$ avec δ comme suit (c'est $\emptyset$ pour les cas non spécifiés) :

— $\delta(q_0, a, \gamma_0) := \{(q_1, \gamma_0)\}$

— $\delta(q_1, a, \gamma) := \{(q_1, \gamma\alpha)\}$ pour $\gamma \in \{\gamma_0, \alpha\}$;

— $\delta(q_1, b, \alpha) := \{(q_2, \epsilon)\}$

— $\delta(q_2, b, \alpha) := \{(q_2, \epsilon)\}$;
— $\delta(q_2, b, \gamma_0) := \{(q_3, \gamma_0)\}$.
Graphiquement :



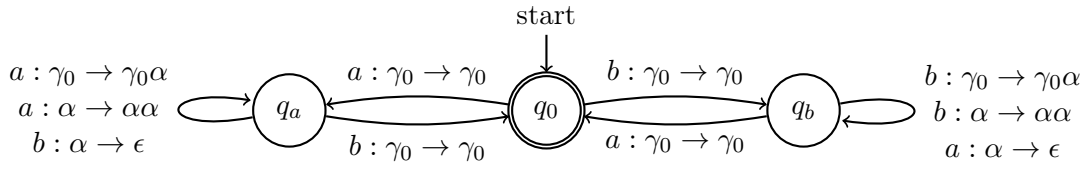
Intuitivement, l'automate passe dans l'état 1 quand il voit un a ; il reste dans l'état 1 tant qu'il lit des a et empile un α par a lu. Lorsqu'il lit des b, il passe dans l'état 1 et dépile un α par b lu. Il atteint l'état final lorsque le fond de la pile γ_0 est atteint en lisant le dernier b, auquel cas il accepte précisément si c'est la fin du mot.

Formellement, montrons que, pour tout mot $w = aa^nbb^n$ avec $n \in \mathbb{N}^*$, l'automate accepte w. L'automate suit la séquence de configurations $(q_0, \gamma_0), (q_1, \gamma_0), (q_1, \gamma_0\alpha), \ldots, (q_1, \gamma_0\alpha^n), (q_2, \gamma_0\alpha^{n-1}), \ldots, (q_2, \gamma_0\alpha), (q_2, \gamma_0), (q_3, \gamma_0)$, et il accepte car q_3 est final.

À l'inverse, considérons un chemin de configurations, et montrons que le mot reconnu est de la forme aa^nbb^n avec $n \in \mathbb{N}^*$. On commence à la configuration (q_0, γ_0), et on lit nécessairement un a pour parvenir à la configuration (q_1, γ_0). On lit alors un certain nombre de a, notons n ce nombre, et on parvient à la configuration $(q_1, \gamma_0\alpha^n)$. Il faut ensuite lire un b pour parvenir à la configuration $(q_1, \gamma_0\alpha^{n-1})$ ce qui impose que $n > 0$. Ensuite, pour parvenir à l'état final q_3, il faut que le sommet de pile soit γ_0, ce qui impose de lire $n - 1$ fois un b pour parvenir à la configuration (q_2, γ_0). Ensuite, il faut lire un autre b pour effectuer la transition et parvenir à (q_3, γ_0). On a alors lu $n + 1$ fois a, suivi de $1 + (n - 1) + 1$ fois b, avec $n > 0$, donc un mot de la forme attendue. Le mot se termine forcément à ce moment, car il n'y a plus de transition sortante. Ainsi, tout mot accepté par l'automate est de la bonne forme, ce qui conclut la preuve.

On en déduit qu'il existe des langages non rationnels reconnus par un automate à pile, car le langage $\{a^nb^n \mid n \geq 2\}$ n'est pas rationnel : ceci se montre sans difficulté par le lemme de l'étoile.

Question 2. On construit l'automate suivant sur l'alphabet de pile $\Gamma = \{\alpha, \gamma_0\}$:



Il est clair que tout mot avec autant de a que de b est accepté. En effet, considérons la fonction δ qui à toute position $0 \leq i \leq n$ d'un mot d'entrée w de longueur n associe $|w'|_a - |w'|_b$ pour w' le préfixe de longueur i du mot. On montre facilement par induction sur la position que, pour tout $0 \leq i \leq n$, si l'on considère la configuration où on se trouve après avoir lu le préfixe w' de longueur i de w :
— on a $\delta(i) = 0$ ssi la configuration est (q_0, γ_0)
— on a $\delta(i) > 0$ ssi la configuration est $(q_a, \gamma_0\alpha^{|\delta(i)|-1})$
— on a $\delta(i) < 0$ ssi la configuration est $(q_b, \gamma_0\alpha^{|\delta(i)|-1})$
Ainsi, le mot est accepté si et seulement si $\delta(|w|) = 0$, c'est-à-dire si et seulement s'il est dans le langage.

Question 3. Il est clair que $L_{\leq\eta}(A) \subseteq L(A)$ pour tout η, car tout calcul η-tronqué permet d'obtenir un calcul : la seule différence est qu'un calcul η-tronqué peut se bloquer parce que la pile est devenue vide, alors qu'elle n'aurait pas été vide dans un vrai calcul (i.e., on a dépilé plus profond que η).

Pour la seconde partie de la question, on construit simplement un automate fini sur les configurations η-tronquées : chaque configuration η-tronquée correspond à un état. Le point clé est que les configurations η-tronquées sont en nombre fini, donc on obtient bien un automate fini.

Question 4. Fixons w un mot de longueur n, l'automate à pile A, et $\eta \in \mathbb{N}^*$. Supposons que $w \in L(A) \setminus L_{\leq\eta}(A)$, et considérons un calcul acceptant $\chi = (q_0, z_0), \ldots, (q_p, z_p)$ de A sur w.

On observe d'abord que, clairement, χ est un calcul acceptant η-tronqué à moins qu'il existe un point $0 \leq m \leq p$ et un point $m < r' \leq p$ où $h_m - h_{r'} \geq \eta$. En effet, la seule façon que χ échoue en tant que calcul η-tronqué est qu'on atteigne la pile vide à un moment où la pile n'est pas véritablement vide, ce qui veut dire qu'on avait la pile à une hauteur h_m et qu'on atteint ensuite une hauteur de $h_m - \eta$ ou moins. En fait, vu que l'on ne peut dépiler les symboles que un par un, quitte à prendre $r' > m$ aussi petit que possible, on peut supposer que $h_m - h_{r'} = \eta$, et que r' est la plus petite valeur $> m$ pour laquelle c'est le cas. Ainsi, comme $w \notin L_{\leq\eta}(A)$, on sait qu'il existe un tel témoin $0 \leq m < r \leq p$ tels que $h_m - h_{r'} = \eta$ pour le calcul χ, et pour tout $m \leq j < m$, on a $h_j > h_{r'}$ par minimalité de r'.

Étant donné ce témoin, on cherche à construire une $(\eta - G)$-montagne. Comme la pile est initialement de hauteur 1, et qu'elle croît de G à chaque étape, il est clair qu'il existe $0 \leq l < m$ tel que $h_{r'} \leq h_l \leq h_{r'} + G$. On prend le plus grand $l < m$ tel que h_l ait cette propriété. On diminue ensuite r' en r de sorte à garantir que $h_r = h_l$ (ce qui assure par l'encadrement de h_l que $r' - r \leq G$) : ceci est possible car, vu que la taille de pile décroît de h_m à $h_{r'}$ et qu'on dépile au plus un symbole à chaque fois, toutes les valeurs intermédiaires sont forcément atteintes au cours de la descente (au contraire de ce qu'il se passait précédemment pour la montée). On prend le plus petit r avec cette propriété. On a maintenant construit $0 \leq l < m < r \leq p$ tels que $h_l = h_r$ et $h_r \geq \eta - G$. Il reste juste à observer que $h_j > h_l$ pour tout $l < j < r$, mais pour $j \leq m$ c'est une conséquence de la maximalité de l, et pour $m \leq j$ c'est une conséquence de la minimalité de r.

On a donc construit une $(\eta - G)$-montagne (l, m, r).

Question 5. On observe tout d'abord que, par les propriétés d'une montagne, le dernier symbole de la pile z_r est également γ, car ce symbole n'a pu être dépilé entre l et r.

Le nombre d'états de base est de $|Q|^2 \times |\Gamma|$, et on définit η_0 comme $|Q|^2 \times |\Gamma| \times (G+1)$, où G est défini à partir de A comme à la question précédente. Considérons un mot w et un calcul χ de A sur w qui a une η_0-montagne (l, m, r). De z_l à z_m, la pile passe d'une hauteur h_l à h_m ; comme on empile au plus G symboles à chaque étape de calcul, par définition de η_0, il y a au moins $v := 1 + |Q|^2 \times |\Gamma|$ hauteurs entre h_l et h_m qui sont atteintes à un indice entre l et m. Pour chacune de ces hauteurs, comme à la question précédente, il existe une position entre m et r où la même hauteur de pile est atteinte (toutes les hauteurs de pile sont atteintes pendant la descente). Ainsi, à partir de la η_0-montagne (l, m, r), on peut définir v montagnes imbriquées : spécifiquement, on a la séquence $h_l < h'_1 < \cdots < h'_v < h_m$ des hauteurs atteintes, et les indices $l < l'_1 < \cdots < l'_v < m$ où elles le sont, que l'on choisit chacun maximaux parmi les indices $< m$ qui atteignent la hauteur respective. On a ensuite la séquence des indices $m < r'_1 < \cdots < r'_v < r$ tels que $h_{l'_i} = h_{r'_i}$ pour tout $1 \leq i \leq v$, que l'on choisit chacun minimaux parmi les indices $> m$ qui atteignent la hauteur respective (il est clair qu'ils satisfont bien la relation d'ordre indiquée). Pour tout $1 \leq i \leq v$, il est ainsi clair que (l'_i, m, r'_i) est une η_i-montagne pour un certain $\eta_i > 0$: la seule condition à vérifier est celle sur les h_j, qui est vraie comme précédemment par maximalité des l'_i et minimalité des r'_i.

À présent, par définition de v et par le principe des tiroirs, il y a nécessairement deux de ces montagnes qui ont le même état de base, ce qui permet de conclure.

Question 6. On pose $\eta := \eta_0 + G$, pour η_0 comme défini à la question précédente. On sait par la question 3 que $L_{\leq\eta}(A)$ est un langage régulier : on pose p la longueur de pompage classique pour un automate A (sans pile) qui le reconnaît. Prenons un mot $w \in L$ avec $|w| > p$. Si $w \in L_{\leq\eta}(A)$, comme

$|w| > p$, on peut appliquer le pompage pour A et décomposer $w = uvz$ avec $|v| \geq 1$ (correspondant au cycle dans l'automate) et $uv^n z \in L$ pour tout $n \in \mathbb{N}$, et on conclut en posant $x = y = \epsilon$.

Si $w \in L(A) \setminus L_{\leq \eta}(A)$, alors on considère un calcul acceptant de A sur w, on note p sa longueur. D'après la question 4, ce calcul a une η_0-montagne. D'après la question 5, il existe deux montagnes (l', m, r') et (l'', m, r'') avec $l' < l'' < m < r'' < r'$ telles que $\beta(l', r') = \beta(l'', r'')$. On décompose $w = uvxyz$ suivant $0 \leq l' < l'' < r'' < r' \leq p$, ce qui garantit la première condition. Pour la seconde, on observe qu'on peut construire à partir de χ un calcul acceptant pour $uv^n xy^n z$ pour $n \in \mathbb{N}$. L'idée est que l'on peut reproduire les étapes du calcul correspondant à v et y : pour v, on part et on arrive au même état q_l avec le même sommet de pile qu'on peut lire mais qu'on ne peut jamais dépiler, et ce faisant on empile un certain mot z ; et pour y, on part et on arrive au même état q_r avec le même sommet de pile γ, en dépilant z sans descendre au-delà.

[La classe des langages reconnus par les automates à pile s'appelle également classe des langages algébriques*, et le résultat démontré à cette question est le pendant pour les automates à pile du théorème de Bar-Hillel, Perles et Shamir, cf [Wik17], avec un petit affaiblissement (on ne montre pas la condition $|vxz| \leq p$). La preuve proposée suit [AJ13].]*

Références

[AJ13] Antoine Amarilli and Marc Jeanmougin. A Proof of the Pumping Lemma for Context-Free Languages Through Pushdown Automata, 2013.

[Wik17] Wikipedia. Pumping lemma for context-free languages, 2017. `https://en.wikipedia.org/wiki/Pumping_lemma_for_context-free_languages`.

A9 – Automates pour les valuations de formules booléennes

Soit $\mathcal{X}$ un ensemble fini de variables. Une *valuation* de $\mathcal{X}$ est une fonction de $\mathcal{X}$ dans $\{0,1\}$. Soit Φ une formule de la logique propositionnelle sur $\mathcal{X}$. On dit que la valuation ν *satisfait* Φ si la formule Φ s'évalue à 1 lorsque l'on remplace chaque variable dans Φ par son image par ν.

On fixe l'alphabet $\Sigma = \{0,1\}$. Soit $<$ un ordre total sur $\mathcal{X}$: on écrira en conséquence $\mathcal{X} = x_1, \ldots, x_n$, avec $x_i < x_j$ pour tout $1 \leq i < j \leq n$. Le *mot suivant* $<$ d'une valuation ν de $\mathcal{X}$ est le mot $\nu(x_1) \cdots \nu(x_n)$ de longueur n sur l'alphabet Σ. Un *automate de valuations* pour Φ et $<$ est un automate A sur l'alphabet Σ tel que, pour tout mot $w \in \Sigma^*$, l'automate A accepte w si et seulement si $|w| = n$ et w est le mot suivant $<$ d'une valuation ν qui satisfait Φ.

Question 0. Construire un automate de valuations pour la formule $(x_1 \wedge x_3) \vee x_2$.

Question 1. Proposer un algorithme naïf qui, étant donné une formule Φ de la logique propositionnelle, construit un automate de valuations pour Φ. Discuter de la complexité de cet algorithme.

Dans les deux prochaines questions, on cherche à améliorer l'efficacité de cet algorithme.

Question 2. Pour tout $n \in \mathbb{N}$, on appelle L_n le langage sur Σ défini par $L_n := \{ww \mid w \in \Sigma^n\}$. Montrer que, pour tout $n \in \mathbb{N}$, pour tout automate A qui reconnaît L_n, l'automate A a au moins 2^n états.

Question 3. En utilisant la question précédente, montrer que, pour tout ensemble de variables totalement ordonné de taille paire $\mathcal{X} = x_1, \ldots, x_{2n}$, on peut construire une formule Φ_{2n} de taille $O(n)$ telle que tout automate de valuations pour Φ_{2n} et $<$ ait au moins 2^n états.

Qu'en déduire quant à l'algorithme de la question 1 ?

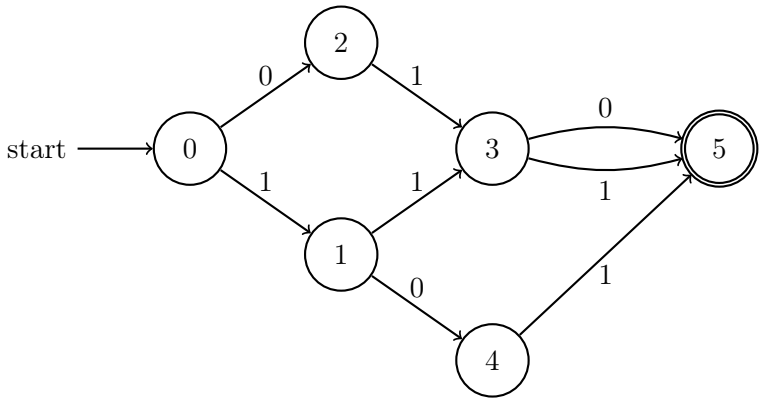
Suite des questions

Question 4. Montrer le même résultat qu'à la question précédente pour une famille de formules Φ'_{2n} qui utilise seulement les opérateurs $\vee$ et $\wedge$ (et pas $\neg$).

Question 5. Soit $\mathcal{X}$ un ensemble de variables arbitraire de taille paire totalement ordonné par $<$, soit $2n$ la taille de cet ensemble, et soit Φ_{2n} la formule définie à la question 3. Existe-t-il un ordre différent $<'$ sur $\mathcal{X}$ pour lequel il y ait un automate de valuations pour Φ_{2n} et $<'$ de taille plus faible que 2^n ?

Corrigé

Question 0. On construit par exemple l'automate suivant :



Question 1. On peut énumérer l'ensemble des valuations de $\mathcal{X}$. Pour chaque valuation ν, on teste si ν satisfait Φ, et on construit ainsi l'ensemble des valuations qui satisfont Φ. On construit explicitement l'ensemble L des mots suivant $<$ de ces valuations, et, comme ce langage est fini, on peut construire efficacement un automate qui reconnaît exactement ce langage : pour chaque mot de L, il y a un chemin d'états depuis l'état initial qui mène à un état final avec des transitions dont la concaténation des étiquettes forme w.

Si l'on suppose ici que chaque variable de $\mathcal{X}$ apparaît dans Φ, la complexité de cet algorithme est en $O(|\Phi| \times 2^n)$, car on fait pour chaque valuation une opération linéaire en la formule, et (dans l'automate) linéaire en n, or $n \leq |\Phi|$. En général, la complexité est en $O(\max(|\Phi|, n) \times 2^n)$.

Question 2. *Indication possible : proposer de traiter d'abord le cas d'un automate déterministe, pour lequel la preuve est un peu plus simple : il suffit de considérer l'image de tous les mots de Σ^n et de montrer que c'est une fonction totale et injective.*

Soit A un automate qui reconnaît L_n. On appelle Q l'ensemble d'états de A. Montrons que $|Q| \geq 2^n$.

Soit f la fonction de Σ^n dans 2^Q qui, a un mot $w \in \Sigma^n$, associe un état q_w tel qu'il existe un calcul de l'automate pour ww (c'est-à-dire un chemin de q_0 à un état final dont les transitions sont étiquetées par des lettres dont la concaténation forme ww) tel que q_w est l'état auquel on aboutit après avoir suivi n transitions. Notons qu'un tel calcul existe nécessairement, car ww est dans L_n et accepté par l'automate ; ainsi, la fonction f est une fonction totale.

Montrons à présent que f est injective. Procédons par l'absurde, et supposons qu'il existe $w_1 \neq w_2$ dans Σ^n tels que $f(w_1) = f(w_2)$. Soit $q := f(w_1)$. Ainsi, il existe un chemin π_1 dans l'automate étiqueté par w_1 qui va de q_0 à q, et un chemin π'_1 dans l'automate étiqueté par w_1 qui va de q à un état final q_1 ; et un chemin π_2 étiqueté par w_2 qui va de q_0 à q, et un chemin π'_2 étiqueté par w_2 qui va de q à un état final q_2. Considérons à présent le chemin formé en concaténant π_1 et π'_2 : ce chemin est étiqueté par

w_1w_2, et va de l'état initial q_0 à l'état final q_2, donc il montre que l'automate accepte w_1w_2. Or on a $w_1w_2 \in \Sigma^n$ mais $w_1 \neq w_2$, donc $w_1w_2 \notin L_n$, ce qui contredit le fait que A reconnaisse L_n. Ainsi, f est injective.

Comme $|\Sigma^n| = 2^n$, on déduit de l'existence de f que $|Q| \geq 2^n$, ce qu'il fallait démontrer.

Question 3. Fixons $n \in \mathbb{N}$. Le cas $n = 0$ est trivial avec la formule tautologique, donc on suppose $n > 0$.

Pour $1 \leq i \leq n$, soit Ψ_i la formule $(x_i \wedge x_{n+i}) \vee (\neg x_i \wedge \neg x_{n+i})$. Intuitivement, une valuation ν satisfait Ψ_i si et seulement si $\nu(x_i) = \nu(x_{n+i})$. Construisons la formule suivante :

$$\Phi_{2n} := \bigwedge_{1 \leq i \leq n} \Psi_i$$

Ainsi, une valuation ν de $\mathcal{X} := x_1, \ldots, x_{2n}$ satisfait Φ_{2n} si et seulement si, pour tout $1 \leq i \leq n$, on a $\nu(x_i) = \nu(x_{n+i})$. Ceci est le cas si et seulement si le mot de ν suivant $<$ est dans le langage L_n de la question précédente. Ainsi, d'après la question précédente, tout automate de valuations pour Φ_{2n} et $<$ a au moins 2^n états.

Or, la taille de Φ_{2n} est bien en $O(n)$, car chaque Ψ_i est de taille constante. Ceci conclut la preuve.

Ainsi, on ne peut pas espérer obtenir un algorithme efficace pour le problème de la question 1, parce que, dans le cas de la famille Φ_{2n}, un algorithme qui répond à ce problème doit matérialiser un automate de taille exponentielle en la formule d'entrée.

Question 4. On fixe à nouveau $n \in \mathbb{N}^*$. On définit la formule $\Psi'_i := x_i \vee x_{n+i}$, et on définit :

$$\Phi'_{2n} := \bigwedge_{1 \leq i \leq n} \Psi'_i$$

Cette formule utilise les bons opérateurs, et est toujours de taille $O(n)$. Montrons que tout automate de valuations pour Φ_{2n} et $<$ a 2^n états. Il est clair qu'une valuation ν de $\mathcal{X}$ satisfait Φ_{2n} si et seulement si le mot suivant $<$ de ν est dans le langage $L'_n := \{ww' \mid w, w' \in \Sigma^n \wedge \forall i \in \{1, \ldots, n\}, w_i = 1 \vee w'_i = 1\}$. Ainsi, montrons le pendant de la question 2 pour ce langage.

Soit un automate A qui reconnaisse L'_n, et soit Q son ensemble d'états. Montrons que $|Q| \geq 2^n$. Soit g la fonction de Σ^n dans Q qui, pour tout $w \in \Sigma^n$, considère un calcul de l'automate étiqueté par $w\overline{w}$, où $\overline{w}$ est le complément de w (c'est-à-dire le mot sur Σ tel que $w_i \neq \overline{w}_i$ pour tout $1 \leq i \leq n$). Ce mot est nécessairement dans L'_n, donc un tel calcul existe, et $g(w)$ choisit et renvoie un état q auquel on peut arriver après n transitions dans un tel calcul. Montrons que g est injective.

Procédons par l'absurde et supposons qu'il existe $w \neq w'$ dans Σ^n tel que $g(w) = g(w')$; soit $q := g(w)$. Comme $w \neq w'$, il existe $1 \leq i \leq n$ tel que $w_i \neq w'_i$; comme $\Sigma = \{0, 1\}$, quitte à échanger w et w', on peut supposer que $w_i = 1$ et $w'_i = 0$. Ainsi, $\overline{w}_i = 0$ mais $\overline{w'}_i = 1$. On considère à présent le chemin de l'état initial q_0 à q étiqueté par w', et le chemin de q à un état final étiqueté par $\overline{w}$. Ce chemin montre que $w'' := w'\overline{w}$ est accepté par l'automate. Pourtant, on sait que $w''_i = 0$ et $w''_{n+i} = \overline{w}_i = 0$. Ainsi $w'' \notin L'_n$. On est donc parvenu à une contradiction.

Question 5. Fixons $n \in \mathbb{N}$, et $\mathcal{X}$. Considérons l'ordre $<'$ défini comme suit :

$$x_1 <' x_{n+1} <' x_2 <' x_{n+2} <' \cdots <' x_{n-1} <' x_{2n-1} <' x_n <' x_{2n}.$$

Ainsi, l'ordre suivant $<'$ des valuations qui satisfont Φ_{2n} est à présent $(00|11)^n$.

Construisons un automate de valuations pour Φ_{2n} et $<'$ de taille $O(n)$. On définit l'ensemble d'états comme $Q := \{q_0, q_1, \ldots, q_n\} \cup \{q_i^b \mid 1 \leq i \leq n, b \in \{0, 1\}\}$, l'état initial est q_0, l'état final est q_n, et les transitions sont comme suit : pour $1 \leq i \leq n$, pour $b \in \{0, 1\}$, une transition étiquetée b de q_{i-1} à q_i^b et de q_i^b à q_i.

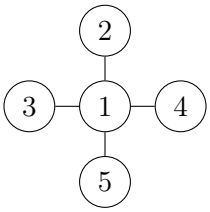
Il est clair que l'automate est bien de taille linéaire. On montre par récurrence descendante pour i de n à 0 que le langage reconnu à partir de l'état q_i est $(00|11)^{n-i}$. Le cas de base est celui de l'état q_n, qui est final et n'a pas de transition sortante, donc reconnaît le langage $\{\epsilon\}$. Pour la récurrence, si l'on suppose le résultat pour q_i pour un certain $0 < i \leq n$, on montre le résultat pour q_{i-1} car les mots acceptés depuis q_{i-1} sont les mots de q_i précédés de 00 ou de 11. Ceci prouve que l'automate obtenu reconnaît effectivement le bon langage.

A10 – Ampleur de tris sur des graphes

Dans ce problème, on appelle *graphe* un graphe non-orienté et non-vide. Soit $G = (V, E)$ un graphe, et posons $n := |V|$ son nombre de sommets. Pour toute partition de V en deux ensembles disjoints X et Y tels que $X \cup Y = V$, la *frontière* de (X, Y) est le sous-ensemble de E défini par $\mathcal{F}_G(X, Y) := \{\{u, v\} \in E \mid (u, v) \in (X \times Y) \cup (Y \times X)\}$. Pour toute bijection $\sigma : \{1, \ldots, n\} \to V$, on appelle *tri* de G la séquence $S = \sigma(1), \ldots, \sigma(n)$, et pour tout $1 \leq i < n$ on définit $S_{\leq i} := \{\sigma(1), \ldots, \sigma(i)\}$ et $S_{>i} := \{\sigma(i+1), \ldots, \sigma(n)\}$. L'*ampleur* du tri S est :

$$\mathcal{A}_G(S) := \max_{1 \leq i < n} |\mathcal{F}_G(S_{\leq i}, S_{>i})|$$

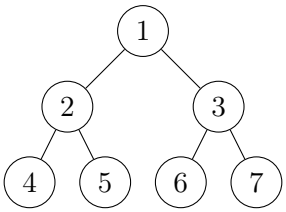
Question 0. Montrer que, pour le graphe G suivant, tout tri de G est d'ampleur au moins 2 :



Question 1. Montrer que, pour tout $n \in \mathbb{N}^*$, il existe un graphe B_n à $n+1$ sommets et n arêtes tel que tout tri de B_n est d'ampleur au moins $\lceil n/2 \rceil$.

Question 2. Montrer que, pour tout $n \in \mathbb{N}^*$, il existe un graphe C_n à n sommets tel que tout tri de C_n est d'ampleur $\lfloor n/2 \rfloor \times \lceil n/2 \rceil$.

Question 3. Pour tout $n \in \mathbb{N}^*$, on appelle A_n l'arbre binaire complet de hauteur n, vu comme un graphe. Par exemple, A_2 est comme suit :



Pour tout $n \in \mathbb{N}^*$, construire un tri de A_n d'ampleur $2^{n+1} - 2$.

Question 4. Construire un autre tri de A_n d'ampleur n.

Suite des questions

Question 5. Un chemin $\pi = v_1 \dots v_n$ dans un graphe G est dit *simple* s'il n'existe pas $1 \leq i < j < n$ tels que $\{v_i, v_{i+1}\} = \{v_j, v_{j+1}\}$. Un *cycle* dans un graphe G est un chemin simple $\pi = v_1 \dots v_n$ de longueur $n > 1$ tel que $v_1 = v_n$. On dit que G est *acyclique* si G n'a pas de cycle.

Montrer que, pour tout graphe G, si G est acyclique et si tous les sommets de G ont degré au plus 3, alors il existe un tri de G d'ampleur au plus $2 + \lfloor \log_2 n \rfloor$, où n est le nombre de sommets de G.

Question 6. Pour tout $n \in \mathbb{N}^*$, on définit le graphe $G_n = (V_n, E_n)$ par $V_n = \{(i,j) \mid 1 \leq i \leq n, 1 \leq j \leq n\}$, et $E_n = \{\{(i,j),(i',j')\} \mid (i = i' \wedge |j - j'| = 1) \vee (j = j' \wedge |i - i'| = 1)\}$. Construire un tri de G_n d'ampleur $2n(n-1)$. Pour $n \geq 3$, construire un tri de G_n d'ampleur $n+1$.

Question 7. Pour tout $n \geq 3$, montrer que tout tri de G_n est d'ampleur au moins $n+1$.

Corrigé

[La notion d'ampleur d'un tri définie dans ce problème est liée à celle de largeur linéaire *d'un graphe, voir [Wik17].]*

Question 0. On montre directement le résultat général en question 1.

Question 1. Pour tout $n \in \mathbb{N}^*$, on définit $B_n := (\{0, \dots, n\}, \{\{0, i\} \mid 1 \leq i \leq n\})$. Ce graphe a $n+1$ sommets et n arêtes. Soit $S = s_1 \cdots s_{n+1}$ un tri de G : montrons que $\mathcal{A}_{B_n}(S)$ est d'ampleur au moins $\lceil n/2 \rceil$. Soit $1 \leq i \leq n+1$ tel que $s_i = 0$. Soit $i \leq 1 + n/2$, soit $i \geq 1 + n/2$; plus précisément, comme i est entier, on a soit $i \leq 1 + \lfloor n/2 \rfloor$, soit $i \geq 1 + \lceil n/2 \rceil$.

Si $i \geq 1 + \lceil n/2 \rceil$, alors considérons la partition $(X, Y) := (s_1 \dots s_{i-1}, s_i \dots s_{n+1})$. Les sommets $\{s_1, \dots, s_{i-1}\}$ sont différents de 0, donc ils sont dans l'ensemble $\{1, \dots, n\}$, et comme 0 n'est pas dans $\{s_1, \dots, s_{i-1}\}$, les arêtes qui relient ces sommets à 0 sont dans $\mathcal{F}_{B_n}(X, Y)$. Ainsi, $|\mathcal{F}_{B_n}(X, Y)| = |\{s_1, \dots, s_{i-1}\}|$, et cette quantité vaut $i - 1$, qui est $\geq \lceil n/2 \rceil$. Ainsi, $\mathcal{A}_{B_n}(S) \geq \lceil n/2 \rceil$.

Si $i \leq 1 + \lfloor n/2 \rfloor$, alors considérons la partition $(X, Y) := (s_1 \dots s_i, s_{i+1} \dots s_{n+1})$. En considérant les arêtes adjacentes à $\{s_{i+1}, \dots, s_{n+1}\}$, on déduit comme précédemment que $|\mathcal{F}_{B_n}(X, Y)| = (n+1) - i = n - i + 1$, et comme $i \leq 1 + \lfloor n/2 \rfloor$, on a $n - i + 1 \geq n - \lfloor n/2 \rfloor$, et le membre de droite vaut $\lceil n/2 \rceil$, d'où l'inégalité souhaitée. Ceci conclut la preuve.

Question 2. Pour tout $n \in \mathbb{N}^*$, on définit $C_n := (\{1, \dots, n\}, \{\{i, j\} \mid 1 \leq i < j \leq n\})$. Ce graphe a n sommets. Il est clair que, par symétrie, tous les tris de C_n ont la même ampleur, et le maximum de l'ampleur est clairement atteint pour $i = \lfloor n/2 \rfloor$ ou $i = \lceil n/2 \rceil$, auquel cas elle vaut le produit du nombre de sommets de chaque côté de la partition, à savoir $\lfloor n/2 \rfloor \times \lceil n/2 \rceil$.

Question 3. On appelle *niveau* de A_n l'ensemble des sommets situés à la même profondeur. On considère la partition des sommets de A_n en niveaux pairs et niveaux impairs, et on la réalise comme la partition (X, Y) d'un tri obtenu en choisissant un ordre arbitraire sur les sommets des niveaux pairs et sur ceux des niveaux impairs. Il est clair que toutes les arêtes de A_n sont dans $\mathcal{F}_{A_n}(X, Y)$, et il y en a autant que de sommets dans l'arbre moins 1 (la racine, qui n'a pas d'arête parente), c'est-à-dire $2^{n+1} - 2$.

Question 4. On construit le second tri par un parcours de l'arbre en ordre infixe. Montrons d'abord que l'ampleur de ce tri est d'au moins n. On suppose d'abord que n est pair. Soit l une feuille atteinte à partir de la racine en suivant l'enfant gauche, puis droit, puis gauche, etc. Juste avant que le tri énumère l, on montre que la frontière est de taille n en démontrant par récurrence sur $0 \leq i \leq n$ que la frontière dans le sous-arbre enraciné au j-ème ancêtre a_j de l est de taille j. Le cas de base est que le sous-arbre enraciné en l est un singleton donc la frontière a taille 0. Pour la récurrence, si a_j est l'enfant gauche de a_{j+1}, vu qu'on s'apprête à énumérer l qui est un descendant de l'enfant droit de a_j, alors on n'a pas encore énuméré a_{j+1} mais on a énuméré a_j, ainsi l'arête $\{a_j, a_{j+1}\}$ est dans la frontière, et on n'a rien énuméré dans le sous-arbre enraciné à l'enfant droit de a_{j+1}, donc la frontière dans le sous-arbre enraciné en a_{j+1} contient une arête de plus que celle dans le sous-arbre enraciné en a_j. Si a_j est l'enfant droit de a_{j+1}, vu qu'on s'apprête à énumérer a_j ou un descendant de son enfant gauche, alors on a déjà énuméré a_{j+1} mais pas encore a_j, donc à nouveau $\{a_j, a_{j+1}\}$ est dans la frontière, et on a tout énuméré dans le sous-arbre enraciné à l'enfant gauche de a_{j+1}, donc encore une fois la frontière contient une arête de plus.

Si n est impair, on définit l de la même manière, et on raisonne sur le moment juste après que le tri a énuméré l. La preuve est analogue.

On montre ensuite que l'ampleur de ce tri est de n au plus. On montre à cet effet que, pour toute partition, il y a au plus une arête par niveau dans la frontière. Supposons par l'absurde qu'il y a deux arêtes distinctes $\{u, v\}$ et $\{u', v'\}$ dans la frontière, où u et u', et v et v', sont au même niveau. Soit $u \neq u'$, soit $u = u'$. Dans le premier cas, soit w le plus petit ancêtre commun de u et de u' ; on sait alors que u est descendant d'un enfant z de w et u' descendant de l'autre enfant z', on suppose sans perte de généralité que z est l'enfant gauche et z' l'enfant droit. Dans ce cas, comme $\{u, v\}$ contient un sommet non énuméré, on sait que l'énumération n'a pas fini de traiter le sous-arbre enraciné en z, ainsi w n'est pas encore énuméré. Mais, comme $\{u', v'\}$ contient un sommet énuméré, on sait que l'énumération a commencé à traiter le sous-arbre enraciné en z', donc w est énuméré. On a une contradiction.

Dans le second cas, si $u = u'$ (et donc $v \neq v'$), il suffit de montrer qu'à aucun stage de l'algorithme il n'est possible que les deux arêtes enfant d'un nœud soit dans la frontière. Si on considère un nœud u et ses deux enfants v et v', il y a quatre situations possibles au cours de l'exécution :

- Aucun des trois n'est énuméré, aucune des deux arêtes n'est dans la frontière.
- v est énuméré et u n'est pas énuméré, mais alors v' n'est pas énuméré non plus, donc seule $\{u, v\}$ est dans la frontière.
- v est énuméré et u est énuméré également, et v' n'est pas encore énuméré : seule $\{u, v'\}$ est dans la frontière.
- Les trois sommets sont énumérés, aucune des deux arêtes n'est dans la frontière.

Ainsi, l'ampleur de ce tri est de n exactement.

Remarque : Une façon graphique de voir l'ampleur de ce tri est d'agencer l'arbre de sorte que l'énumération des sommets se fasse de gauche à droite : la frontière contient toutes les arêtes traversées par une droite verticale.

Remarque : On peut obtenir un tri d'ampleur $n+1$ en effectuant un parcours préfixe, ou un parcours postfixe.

[On peut en fait construire pour tout $n \in \mathbb{N}^*$ un tri d'ampleur $\min\left(n, \lceil \frac{n+2}{2} \rceil\right)$, et on peut montrer qu'il n'existe pas de tri d'ampleur plus petite, mais c'est plus délicat.]

Question 5. Un graphe acyclique peut clairement être vu comme un arbre. On prend un sommet de degré 1 et on le choisit comme racine, et on obtient un arbre binaire non nécessairement complet (chaque nœud a 0, 1, ou 2 enfants).

En d'autres termes, on veut montrer que si on a un arbre binaire non nécessairement complet alors il existe un tri d'ampleur au plus $2 + \lfloor \log_2 n \rfloor$.

On commence par précalculer, pour chaque nœud de l'arbre, son nombre de descendants. Ce précalcul peut s'effectuer en temps linéaire en allant de bas en haut. Pour chaque nœud, on définit ensuite

un ordre entre ces deux enfants de la façon suivante : le premier enfant est celui qui a le moins de descendants, et le second est celui qui en a le plus. (S'il y a égalité, on fait un choix arbitraire.)

Ensuite, on définit le tri suivant un parcours de l'arbre en ordre préfixe, où quand on traite un nœud, on énumère le nœud, puis on traite d'abord le premier enfant, puis le second enfant.

À chaque fois qu'on énumère un nœud n, il est clair que tous ses ancêtres ont été énumérés. Par ailleurs, pour tout ancêtre strict n' de n, il y a exactement un enfant de n' qui est un ancêtre de n : on dit que n' est un ancêtre *unaire* si c'est le seul enfant, *lourd* si c'est le premier enfant, et *léger* si c'est le deuxième. Pour tout ancêtre léger n' de n, il est clair que le premier enfant de n', et en fait tout le sous-arbre enraciné en n', est énuméré. Ainsi, quand on énumère le nœud n, on peut voir en parcourant de bas en haut le chemin de n à la racine que la frontière contient les arêtes suivantes :

- Pour le sous-arbre enraciné en n, au plus deux arêtes, à savoir les filles de n ; le reste du sous-arbre n'est pas énuméré donc ne contient aucune arête de la frontière.
- Pour chaque ancêtre strict n' de n qui est unaire, comme l'unique enfant n'' de n' est un ancêtre, il est énuméré aussi, donc l'arête entre n' et n'' n'est pas dans la frontière, et les autres arêtes dans la frontière dans le sous-arbre enraciné en n' sont celles dans le sous-arbre enraciné en n'' qu'on a compté en examinant n''.
- Pour chaque ancêtre strict n' de n qui est léger, dans le sous-arbre enraciné en n' :
  - Comme n' et son premier enfant sont énumérés, la première arête fille de n' n'est pas dans la frontière ; par ailleurs dans le sous-arbre enraciné au premier enfant tous les sommets sont énumérés donc la frontière est vide.
  - Comme le second enfant n'' de n' est un ancêtre, il est énuméré, donc l'arête entre n' et n'' n'est pas dans la frontière ; et les autres arêtes dans la frontière dans le sous-arbre enraciné en n' sont celles dans le sous-arbre enraciné en n'', qu'on a compté en examinant n''.
- Pour chaque ancêtre strict n' de n qui est lourd, dans le sous-arbre enraciné en n' :
  - Le second enfant de n' n'est pas énuméré alors que n' l'est, donc l'arête entre n' et son second enfant est dans la frontière ; en revanche, dans le sous-arbre enraciné en ce second enfant, aucun nœud n'est énuméré, donc il n'y a pas d'autres arêtes de la frontière.
  - Comme précédemment, comme le premier enfant de n' est un ancêtre, il est énuméré, donc il n'y a pas d'autres arêtes dans la frontière que celles considérées en traitant le premier enfant.

On a ainsi montré qu'à tout stade de l'énumération où on vient d'énumérer un nœud, le nombre d'arêtes de la frontière est au plus de deux plus le nombre d'ancêtres lourds du nœud que l'on vient d'énumérer. Ainsi, il suffit de montrer qu'il y a au plus $\lfloor \log_2 n \rfloor$ ancêtres lourds à chaque stade de l'énumération pour justifier que l'ampleur du tri satisfait la borne demandée. On va montrer pour ce faire que, pour tout $k \in \mathbb{N}$, si on suppose qu'il y a k ancêtres lourds alors l'arbre est de taille au moins 2^k : ainsi ceci établit que si on suppose qu'il y a strictement plus que $\lfloor \log_2 n \rfloor$ ancêtres lourds alors on a une contradiction.

On considère un stade quelconque de l'énumération où on vient d'énumérer le nœud n, et on considère le chemin allant de n à la racine, de bas en haut, et passant par tous ses ancêtres. On montre l'affirmation par récurrence sur le nombre k d'ancêtres lourds traversés dans ce chemin. Le cas de base $k = 0$ est clair : si on a énuméré n alors l'arbre est non-vide. Pour le cas de récurrence, supposons qu'on a déjà traversé k ancêtres lourds, et que le sous-arbre $T(a_k)$ enraciné en le k-ième ancêtre lourd a_k est de taille au moins 2^k. Supposons qu'on traverse un $(k+1)$-ième ancêtre lourd a_{k+1} ; ainsi, le premier enfant a' de a_{k+1} est un ancêtre (non nécessairement strict) de a_k. Ainsi, la taille du sous-arbre $T(a')$ enraciné en a' est au moins celle du sous-arbre $T(a_k)$ enraciné en son descendant a_k, donc au moins 2^k. À présent, par l'ordre choisi sur les enfants, la taille du sous-arbre $T(a'')$ enraciné en le second enfant a'' de a_{k+1} est au moins celle de $T(a')$, c'est-à-dire au moins 2^k. Ainsi, la taille du sous-arbre $T(a_{k+1})$ enraciné en a_{k+1} est au moins la somme des tailles de $T(a')$ et $T(a'')$, donc au moins 2^{k+1}, ce qui conclut le cas de récurrence. Ainsi, on a prouvé l'affirmation par récurrence sur le nombre maximal d'ancêtres lourds, ce qui conclut la preuve.

[Cette preuve s'inspire de l'argument donné à la fin de l'appendice C de [ABJM17].]

Question 6. Pour construire un tri d'ampleur $2n(n-1)$, on considère la partition de G_n en les sommets (i,j) où $i+j$ est pair, et en ceux où $i+j$ est impair. Il est clair que toutes les arêtes sont dans la frontière de cette partition, et celle-ci peut être réalisée par un tri quelconque en choisissant un ordre arbitraire sur chaque classe de la partition, ainsi l'ampleur de ce tri est d'au moins $2n(n-1)$, et c'est exactement cette quantité vu que c'est le nombre total d'arêtes. *[En fait, il est facile de voir plus généralement que, pour tout graphe biparti, on peut construire un tri dont l'ampleur est le nombre d'arêtes du graphe.]*

Pour construire un tri d'ampleur n, on prend l'ordre $(1,1),\dots,(1,n)$, puis $(2,1),\dots,(2,n)$, etc., puis $(n,1),\dots,(n,n)$. La frontière est initialement vide, et lorsque l'on vient d'énumérer (i,j) avec $1 \leq i,j \leq n$, la frontière contient les arêtes suivantes :

- L'arête entre (i,j) et $(i,j+1)$ si $j<n$, et c'est la seule arête horizontale de la frontière, parce que pour une arête entre (i',j') et $(i',j'+1)$ avec $1 \leq i' \leq n$, $1 \leq j' < n$:
  - si $i' < i$ alors les deux sommets sont déjà énumérés ;
  - si $i' > i$ alors les deux sommets ne sont pas encore énumérés ;
  - si $i' = i$ et $j' < j$ alors les deux sommets sont déjà énumérés ;
  - si $i' = i$ et $j' > j$ alors les deux sommets ne sont pas encore énumérés.
- Si $i<n$, pour $j' \leq j$, les arêtes entre (i,j') et $(i+1,j')$ sont dans la frontière, soit $j+1$ arêtes ; mais pas celles pour $j' > j$ (les deux sommets ne sont pas encore énumérés) ; au demeurant, pour tout $i < i' < n$, pour tout $1 \leq j' \leq j$, aucune des arêtes entre (i',j') et $(i'+1,j')$ n'est dans la frontière (les sommets ne sont pas encore énumérés).
- Si $i>1$, pour $j' > j$, les arêtes entre $(i-1,j')$ et (i,j') sont dans la frontière, soit $n-j-1$ arêtes ; mais pas celles pour $j' < j$ (les deux sommets sont déjà énumérés) ; au demeurant, pour tout $1 < i' < i$, pour tout $1 \leq j' \leq j$, aucune des arêtes entre $(i'-1,j')$, et (i',j') n'est dans la frontière (les sommets sont déjà énumérés).

On a considéré toutes les arêtes, et on comptabilise $1+j+1+(n-j-1) = n+1$ arêtes dans la frontière au plus, donc l'ampleur de ce tri est d'au plus $n+1$. Il est clair que, si $n \geq 3$, alors l'ampleur est effectivement de $n+1$, lorsque l'on énumère le sommet $(2,1)$: les arêtes entre $(1,j)$ et $(2,j)$ pour $1 < j \leq n$ sont dans la frontière, ainsi que l'arête entre $(2,1)$ et $(3,1)$, et entre $(2,1)$ et $(2,2)$.

Question 7. Pour $1 \leq i \leq n$, on appelle *i-ème ligne* de G_n le sous-ensemble $\{(i,j) \mid 1 \leq j \leq n\}$, et pour $1 \leq j \leq n$ on appelle *j-ème colonne* de G_n le sous-ensemble de la forme $\{(i,j) \mid 1 \leq i \leq n\}$.

Considérons un tri arbitraire S de G_n et montrons que son ampleur est au moins de n. Pour une partition (X,Y) de S, on dit qu'une ligne ou une colonne est *complète* si tous ses sommets sont dans X, qu'elle est *vide* si tous ses sommets sont dans Y, et qu'elle est *entamée* sinon.

Il est clair que chaque sommet que l'on énumère rend complète au plus une ligne (ou colonne). De plus, le nombre de lignes et colonnes complètes est initialement de 0 et finalement de n. Ainsi, on peut considérer le plus petit $1 \leq i' \leq n^2$ où $S_{\leq i'}$ contient exactement une ligne complète et aucune colonne complète, ou $S_{\leq i'}$ contient exactement une colonne complète et aucune ligne complète. Intuitivement, i' est le premier moment où une ligne ou une colonne est complétée. On définit à présent i comme suit : si $S_{\leq i'}$ contient exactement une ligne complète et que les autres sont vides, ou si $S_{\leq i'}$ contient exactement une colonne complète et que les autres sont vides, on pose $i' := i+1$; sinon on pose $i' := i$. Comme $n \geq 3$, il est clair que $S_{\leq i}$ contient toujours exactement une ligne complète et aucune colonne complète, ou exactement une colonne complète et aucune ligne complète : en effet, seul le cas où $i = i'+1$ doit être considéré, et le sommet supplémentaire que l'on a pris ne peut pas avoir complété une ligne ou une colonne. Ceci assure au demeurant que $S_{\leq i}$ contient exactement une ligne complète, au moins une (autre) ligne entamée, et aucune colonne complète ; ou contient exactement une colonne complète, au moins une (autre) colonne entamée, et aucune ligne complète.

On va montrer que la frontière est alors de taille au moins n. On suppose le premier cas, parce que le second est symétrique. On sait qu'aucune colonne n'est complète, mais vu qu'il y a une ligne complète, on sait que chaque colonne intersecte la ligne complète. Ainsi, chaque colonne est entamée. On observe

alors que chaque colonne entamée contient une arête verticale dans la frontière entre deux sommets de la colonne : en effet, la colonne entamée contient des sommets énumérés et des sommets non énumérés, donc il y en a forcément deux adjacents. On a donc au moins n arêtes verticales dans la frontière. De plus, comme il y a au moins une ligne entamée, on sait que la ligne entamée contient une arête horizontale, et il y a donc au moins $n + 1$ arêtes dans la frontière au total.

[Cette preuve s'inspire de la preuve du Lemme 6.4.14 de [Ama16].]

Références

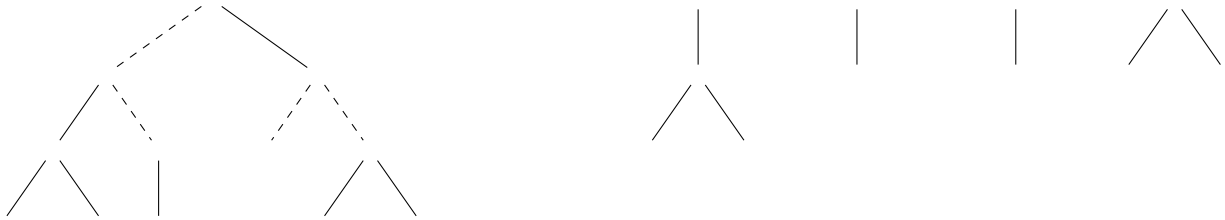
[ABJM17] Antoine Amarilli, Pierre Bourhis, Louis Jachiet, and Stefan Mengel. A Circuit-Based Approach to Efficient Enumeration. In *ICALP*, 2017.

[Ama16] Antoine Amarilli. *Leveraging the Structure of Uncertain Data.* PhD thesis, Télécom ParisTech, 2016.

[Wik17] Wikipedia. Pathwidth, 2017. `https://en.wikipedia.org/wiki/Pathwidth`.

A11 – Comptage de sous-arbres

On considère dans ce sujet des arbres enracinés binaires (chaque nœud a au plus 2 enfants). Étant donné un arbre T, tout sous-ensemble S des arêtes de T définit une forêt, notée $S(T)$, obtenue en conservant les arêtes de T présentes dans S et en retirant les autres (on retire également les nœuds qui n'ont plus d'arêtes incidentes). Par exemple, pour l'arbre représenté ci-dessous à gauche, pour S contenant toutes les arêtes sauf celles en pointillés, on obtient la forêt représentée à droite :



On s'intéresse dans ce problème, étant donné un arbre T et une certaine condition Φ, à compter le nombre de sous-ensembles d'arêtes S de T tels que la forêt $S(T)$ satisfasse la condition Φ.

Question 0. Pour l'arbre d'exemple T et la forêt d'exemple F ci-dessus, combien de sous-ensembles d'arêtes de T permettent d'obtenir la même forêt que F ? Combien y a-t-il de sous-ensembles S d'arêtes de T telles que $S(T)$ consiste exactement en deux arêtes isolées ?

Question 1. On cherche à déterminer, étant donné un arbre T et une hauteur $h \in \mathbb{N}^*$, le nombre de sous-ensembles S de T tels que $S(T)$ contienne un arbre de hauteur au moins h. Proposer un algorithme naïf pour cette tâche, et discuter de sa complexité en fonction de T et de h.

Question 2. Proposer un algorithme plus efficace pour cette tâche à base de programmation dynamique, et discuter de sa complexité en fonction de T et de h.

Question 3. Comment peut-on compter le nombre de sous-ensembles S tels que $S(T)$ contienne un arbre de hauteur *exactement* h ?

Question 4. Le *diamètre* d'une forêt F est le plus grand entier $d \in \mathbb{N}^*$ tel qu'il existe un chemin simple non-orienté de longueur d dans F. Proposer un algorithme qui, étant donné T et d, calcule le nombre de sous-ensembles S de T tels que $S(T)$ soit de diamètre au moins d.

Suite des questions

Question 5. Pour un alphabet fini Σ, on appelle Σ-*arbre* un arbre dont chaque arête est étiquetée par un élément de Σ. Étant donné un Σ-arbre T et un sous-ensemble S d'arêtes de T, on définit $S(T)$ comme précédemment ; il s'agit d'une forêt de Σ-arbres.

Étant donné un mot w de Σ^*, on dit qu'une forêt F de Σ-arbres *contient* w s'il existe un chemin simple non-orienté dans F dont la séquence d'étiquettes soit w.

Proposer un algorithme qui, étant donné un Σ-arbre T et un mot $w \in \Sigma^*$, calcule le nombre de sous-ensembles d'arêtes S de T tels que $S(T)$ contienne w.

Question 6. Proposer un algorithme qui, étant donné un Σ-arbre T et un langage régulier L, calcule le nombre de sous-ensembles d'arêtes S de T tels que $S(T)$ contienne un mot de L.

Indication : On supposera L donné par un automate.

Question 7. On ne fait plus à présent l'hypothèse que les arbres sont binaires. Comment peut-on adapter les algorithmes précédents dans ce cas ? Comment la complexité évolue-t-elle en fonction du degré maximal des arbres ? Peut-on faire mieux ?

Corrigé

[Ce sujet s'inspire de certains résultats de [AMS17].]

Question 0. *[La première partie de la question a une ambiguïté ; parle-t-on de "la même forêt" en conservant l'identité des nœuds, ou d'une forêt isomorphe ? Dans la première interprétation, bien sûr, la réponse est trivialement* 1*, donc c'est la seconde interprétation qu'il faut retenir.]*

Pour obtenir la forêt F, il faut obtenir le premier motif de la forêt, et il y a exactement 4 occurrences de ce motif dans T : deux sont enracinées à la racine, à gauche et à droite ; une est enracinée à l'enfant gauche de la racine ; la dernière est enracinée à l'enfant droit de la racine. Pour les deux premières occurrences, il est clair que, si on en conserve une et qu'aucune arête supplémentaire ne s'y rattache, alors ne peut garder aucune autre arête de ce sous-arbre de la racine, ni la première arête de l'autre sous-arbre de la racine, et il n'y a pas la place dans le reste de l'autre sous-arbre pour obtenir le reste de F, donc on ne peut pas conserver ces motifs. Ainsi, on doit conserver l'un des deux derniers motifs dans T.

Si on conserve le premier, il y a deux possibilités : le choix d'arêtes indiqué, et le choix d'arêtes où on remplace l'arête fille droite de la racine par l'arête fille gauche du fils droit de la racine.

Si on conserve le second, l'état du sous-arbre droit de la racine est fixé, et on se convainc facilement qu'il y a exactement une manière d'obtenir le reste de F. Ainsi, il y a 3 sous-ensembles au total qui permettent d'obtenir la forêt F.

Pour la seconde partie de la question, il suffit de compter le nombre de paires d'arêtes non-adjacentes de T, mais il faut faire attention aux doubles comptes. Si on considère chaque arête dans l'ordre de parcours préfixe, et qu'on compte le nombre de choix possibles pour la seconde arête parmi les arêtes non encore parcourues, on obtient : $8+8+6+6+5+4+2+2+0+0+0$ soit 41 possibilités. On peut aussi procéder en comptant le nombre de paires d'arêtes adjacentes, et en soustrayant ce nombre de $\binom{2}{11}$.

Question 1. L'algorithme naïf consiste à énumérer tous les sous-ensembles de A, à calculer la forêt correspondante, et à vérifier si elle respecte la condition sur la hauteur (ce qui se calcule de bas en haut en temps linéaire). Si on note n le nombre d'arêtes de A, la complexité est en $O(2^n \times n)$.

[Évidemment, cet algorithme naïf fonctionne en temps $2^n(n + f(n))$ pour tester n'importe quelle autre condition qui se teste en temps $f(n)$ sur des forêts avec au plus n arêtes.]

Question 2. Pour chaque nœud n de l'arbre, on considère le sous-arbre T_n enraciné en n, et on partitionne ses forêts possibles de la manière suivante :

— Les forêts possibles qui contiennent un arbre de hauteur au moins h (enraciné n'importe où) ; on dit qu'elles sont de type $\top$

— Pour $0 \leq i < h$, les forêts possibles qui ne contiennent pas de tel arbre, mais où l'arbre ayant n pour racine a hauteur exactement i ; on dit qu'elles sont de type i.

On calcule de bas en haut pour chaque nœud n de T un $(h+1)$-uplet $n_\top, n_0, \ldots, n_{h-1}$ qui indique le nombre de forêts de chaque type pour T_n. Pour une feuille n, on a $n_0 = 1$ et toutes les autres valeurs valent 0. Pour un nœud interne n, on initialise tous les $n_\bullet$ à 0, puis, pour chaque type possible dans $\{\top, 0, \ldots, h-1\}$ pour les nœuds fils de n (au plus 2), pour chaque état possible (conservé ou supprimé) pour les arêtes filles de n (au plus 4 possibilités), on ajoute à n_x le produit des n_j^i pour chaque enfant i et type j pour l'enfant i, où x est un nouveau type calculé à partir du type des enfants et de l'état des arêtes filles. Les règles pour le calcul de x sont comme suit (ceci correspond en réalité aux règles de transition d'un automate d'arbres) :

— S'il y a un enfant de n dont le type est $\top$, alors le nouveau type x est $\top$;

— S'il n'y a aucun tel enfant n, alors le nouveau type est le max, pour les arêtes filles que l'on a conservé, de 1 plus le type du nœud fils pour cette arête.

À la fin, $n_\top$ pour la racine n contient le nombre demandé. On peut en effet montrer par induction que les $n_\bullet$ stockent bien les bons nombres pour le sous-arbre T_n.

La complexité en temps est donc en $O(|T| \times h^2)$, et la complexité en espace est $O(|T| \times h)$; on peut faire un peu mieux en espace, par exemple $O(H \times h)$ où H est la hauteur de T, en observant qu'en calculant suivant un parcours préfixe on a besoin de conserver H tableaux au plus à tout point de l'algorithme, mais on peut effacer les autres.

Question 3. On appelle *hauteur* d'une forêt la hauteur maximale d'un de ses arbres. On peut partitionner les forêts selon leur hauteur maximale, et une forêt est de hauteur exactement h si et seulement si elle est de hauteur au moins h et n'est pas de hauteur au moins $h+1$. Ainsi, le nombre de forêts possibles de hauteur exactement h s'obtient comme la différence du nombre de forêts possibles de hauteur au moins $h+1$ et du nombre de forêts possibles de hauteur au moins h : les deux quantités peuvent être calculées avec l'algorithme de la question précédente (et donc avec la même complexité).

Question 4. On procède comme à la question 2, mais on adapte les règles de transition :

— Si un type enfant est $\top$, alors le nouveau type est $\top$;

— Si aucun type enfant n'est $\top$, alors :

  — S'il y a deux arêtes filles distinctes conservées alors, pour i et j le type des enfants correspondants, si $i + j + 2$ est $\geq d$, alors le nouveau type est $\top$ (intuitivement, on atteint au moins longueur de chemin demandée en montant jusqu'à ce nœud et en descendant depuis ce nœud) ;

  — Sinon, intuitivement ça ne sert à rien de monter et de descendre, et on définit le nouveau type comme le max, sur les arêtes filles conservées, de 1 plus le type du nœud fils de l'arête, exactement comme avant.

On conclut comme précédemment, et la complexité est inchangée.

[Cette question s'inspire de [AMS17], Proposition 5.4.]

Question 5. On procède comme aux questions précédentes, mais il faut adapter les règles de transition : au lieu de mémoriser simplement la longueur maximale (plus l'information $\top$ indiquant si on a déjà réussi), il faut mémoriser, lorsqu'on n'a pas réussi, l'*ensemble* des types possibles de chemin, pas juste la longueur maximale.

Plus précisément, on appelle $w^\leftrightarrow$ l'ensemble des préfixes et suffixes stricts de w, et pour un sous-arbre T_n, on partitionne les forêts possibles de T_n comme :

— Celles qui contiennent le chemin w entier (n'importe où), qui sont de type $\top$ comme auparavant.

— Celles qui ne le contiennent pas, que l'on partitionne suivant le type $S \subseteq w^{\leftrightarrow}$ de leur chemins enracinés. Une forêt est de type S si, lorsque l'on considère l'arbre ayant n pour racine et l'ensemble des chemins enracinés dans la forêt qui partent de n et vont vers le bas, alors l'ensemble des étiquettes de ces chemins intersecté avec $w^{\leftrightarrow}$ est exactement S.

Les règles de transition sont inspirées de la section 4 et sont comme suit :

— Si un type enfant est $\top$, alors le nouveau type est $\top$;
— Si aucun type enfant n'est $\top$, alors :
  — S'il y a deux arêtes filles distinctes conservées avec étiquettes l_1 et l_2, et si le type des enfants correspondants est S_1 et S_2, et si l'un des cas suivants est rempli, alors le nouveau type est $\top$:
    — il existe un préfixe $w^{\rightarrow}$ dans S_1, un suffixe $w^{\leftarrow}$ dans S_2, de sorte que $w^{\rightarrow} l_1 l_2 w^{\leftarrow}$ soit égal à w (intuitivement, on monte par l'enfant 1 et on descend par l'enfant 2) ;
    — la condition analogue en inversant S_1, l_1 et S_2, l_2.
  — Si le cas précédent n'est pas rempli, alors le nouveau type est défini comme suit :
    — Pour les préfixes réalisables, c'est l'union, sur les arêtes filles conservées, des $w^{\rightarrow} l$ (à condition que ce soit toujours un préfixe de w), où $w^{\rightarrow}$ est un préfixe réalisable dans le type du nœud correspondant et l est l'étiquette de l'arête fille ;
    — Pour les suffixes, définition analogue.

    Le type est $\top$ si on vient d'atteindre w (comme un préfixe ou suffixe non-strict).

On peut représenter les préfixes et suffixes comme des entiers (avec annotation de si c'est un préfixe ou un suffixe), donc la complexité en temps est à présent $O(|T| \times (2^{2\times|w|})^2)$, et celle en espace $O(|T| \times 2^{2\times|w|})$; on peut remplacer $|T|$ par la hauteur H de T comme précédemment.

Question 6. C'est une généralisation de la question 5. Le type d'une forêt est à présent défini comme suit :

— Celles qui contiennent un chemin accepté ; de type $\top$.
— Pour les autres, le type consiste en deux sous-ensembles d'états de l'automate : le sous-ensemble d'états où on peut parvenir depuis un état initial en lisant de bas en haut un mot aboutissant à la racine (appelés *états accessibles* du type) ; et le sous-ensemble d'états à partir desquels on peut parvenir à un état final en lisant de haut en bas un mot à partir de la racine (les *états coaccessibles* du type).

Pour les règles de transition :

— Si un enfant a type $\top$ alors on a type $\top$.
— On a type $\top$ s'il y a deux arêtes filles distinctes conservées de sorte qu'on puisse former un chemin acceptant à partir d'un état accessible du type d'un des deux nœuds enfant et d'un état coaccessible du type d'un autre nœud enfant en suivant les transitions des deux arêtes conservées.
— Sinon, le type est défini comme :
  — Pour les états accessibles, l'union, sur toutes les arêtes filles conservées, des états où l'on peut parvenir en prenant un état accessible de l'enfant correspondant à cet arête, et en effectuant une transition étiquetée par l'étiquette de cette arête.
  — Définition analogue pour les états coaccessibles.

  On a type $\top$ si la règle précédente rend accessible un état final ou coaccessible un état initial, c'est-à-dire, s'il y a une arête fille conservée de sorte qu'on puisse former un chemin acceptant à partir d'un état initial en lisant l'arête pour aboutir à un état coaccessible du type de cet enfant, ou en lisant l'arête à partir d'un état accessible du type de cet enfant pour aboutir à un état final.

Le résultat final est obtenu comme auparavant. La complexité est comme à la question précédente en remplaçant $|w|$ par $|Q|$, en négligeant le coût de la lecture des transitions dans l'automate (sinon il faut rajouter un facteur $|A|^2$ dans la complexité en temps).

Question 7. Tous les algorithmes précédents sont en fait adaptables au cas d'arbres d'arité non-bornée, avec la même complexité en espace. En revanche, la complexité en temps est beaucoup plus mauvaise,

parce que l'on énumère tous les états possibles pour les arêtes filles et pour les types des nœuds fils : par exemple, à la question 2, on a à présent $O(|T| \times 2^d \times h^d)$ où d est le degré maximal, et de même aux autres questions.

Pour régler le problème, on peut créer une nouvelle étiquette d'arête ϵ, et rendre l'arbre binaire en remplaçant chaque nœud de degré > 2 par une hiérarchie d'arêtes ϵ. Il suffit ensuite de modifier les algorithmes précédents pour gérer correctement les ϵ, ce qui nécessite deux changements. Premièrement, les arêtes ϵ sont toujours conservées (il faut prendre ceci en compte lorsque l'on énumère tous les choix pour les arêtes filles, pour toujours conserver les arêtes ϵ et ne pas pondérer par erreur avec des choix sur ces arêtes). Deuxièmement, ces arêtes sont gérées différemment pour le calcul du type à partir des types enfants et des configurations des arêtes :

— Pour les longueurs de chemins (questions 2 et 4), au lieu d'ajouter 1 à la longueur pour chaque arête fille traversée, on ajoute 0 ou 1 selon que l'arête fille est une arête ϵ ou non ;
— Pour les étiquettes (question 5), on traite l'étiquette des arêtes ϵ comme le mot vide ϵ dans les concaténations.
— Pour les automates (question 6), suivre des arêtes ϵ revient à n'effectuer aucune transition dans l'automate.

Avec cette astuce, les complexités restent les mêmes que précédemment, à ceci près que les remarques sur la hauteur H de l'arbre doivent être modifiées pour parler du H après récriture de l'arbre pour le rendre binaire.

Références

[AMS17] Antoine Amarilli, Mikaël Monet, and Pierre Senellart. Conjunctive Queries on Probabilistic Graphs: Combined Complexity. In *PODS*, 2017.