Second Concours à l'ENS Paris-Saclay
Interrogation orale d'informatique
Déroulement de l'épreuve

Le sujet se compose de deux problèmes indépendants. Le jury recommande de traiter les deux sujets, au moins partiellement.

Le temps de préparation est de 2 heures. Ensuite le candidat exposera les résultats obtenus pendant une heure à l'oral. Bien entendu, le jury posera des questions et pourra revenir sur des questions omises par le candidat.

Il est autorisé d'admettre le résultat d'une question pour ne pas rester bloqué lors de la phase de préparation.

Il est enfin demandé au candidat de mentionner en début d'oral quelles questions il a traitées dans les deux problèmes. Cela permettra au jury de gérer au mieux le temps et de permettre au candidat de présenter l'intégralité des résultats obtenus lors de la préparation.

La rigueur du raisonnement scientifique est un point capital dans l'évaluation de l'épreuve.

Logique de Presburger

La **logique de Presburger** est l'ensemble des formules logiques générées par la grammaire suivante :

$$\varphi ::= x = 0 \mid x = 1 \mid z = x + y \mid \neg\varphi \mid (\varphi \wedge \varphi) \mid \exists x\, \varphi$$

où x, y, z sont des symboles de variables, choisis parmi un ensemble infini de symboles. Une variable de φ est dite **libre** si elle apparaît dans une formule sans être quantifiée préalablement. Par exemple, dans la formule $\exists y\ y = x + x$, la variable x est libre, mais pas la variable y. On note $\mathrm{Free}(\varphi)$ l'ensemble des variables libres de φ.

1. Donner une définition inductive de l'ensemble $\mathrm{Free}(\varphi)$ des variables libres d'une formule φ de la logique de Presburger.

Une **valuation** des variables libres de φ est une application $\sigma\colon \mathrm{Free}(\varphi) \to \mathbf{N}$ associant à chaque variable une valeur dans les entiers naturels. La **sémantique** d'une formule φ est un ensemble de valuations des variables libres de φ vers des entiers naturels, qu'on note $[\![\varphi]\!]$ et qui est défini récursivement par :

- $[\![x = 0]\!]$ est le singleton contenant l'unique valuation σ associant x à la valeur 0 ;
- $[\![x = 1]\!]$ est le singleton contenant l'unique valuation σ associant x à la valeur 1 ;
- $[\![z = x + y]\!]$ est l'ensemble contenant l'ensemble des valuations σ vérifiant $\sigma(z) = \sigma(x) + \sigma(y)$;
- $[\![\neg\varphi]\!]$ est le complémentaire de l'ensemble $[\![\varphi]\!]$;
- $[\![\varphi \wedge \varphi']\!]$ est l'ensembles des valuations $\sigma\colon \mathrm{Free}(\varphi \wedge \varphi') \to \mathbf{N}$ telles que la restriction de σ au domaine $\mathrm{Free}(\varphi)$ appartient à $[\![\varphi]\!]$, et la restriction de σ au domaine $\mathrm{Free}(\varphi')$ appartient à $[\![\varphi']\!]$;
- $[\![\exists x\, \varphi]\!]$ est l'ensemble des valuations $\sigma\colon \mathrm{Free}(\exists x\, \varphi) \to \mathbf{N}$ telle qu'il existe un entier n tel que la valuation $\sigma'\colon \mathrm{Free}(\varphi) \to \mathbf{N}$ définie par $\sigma'(x) = n$ et $\sigma'(y) = \sigma(y)$ pour tout $y \in \mathrm{Free}(\varphi) \setminus \{x\}$ appartient à $[\![\varphi]\!]$.

2. Décrire la sémantique de la formule $\exists z\ (z = 1 \wedge y = x + z)$.

3. Écrire une formule de la logique de Presburger dont la sémantique est l'ensemble des valuations $\sigma\colon \{x\} \to \mathbf{N}$ telles que $\sigma(x)$ est congru à 1 modulo 3.

Une formule φ est dite **satisfaisable** si sa sémantique $[\![\varphi]\!]$ est non vide. Elle dite **valide** si sa sémantique contient toutes les valuations de ses variables libres, autrement dit si $[\![\neg\varphi]\!]$ est l'ensemble vide. On cherche maintenant à prouver la décidabilité de la satisfaisabilité (ou de la validité) d'une formule de la logique de Presburger. Pour ce faire, on utilise des automates dont on rappelle d'abord les définitions.

Un **automate fini** $\mathcal{A}$ sur l'alphabet A est la donnée d'un quadruplet $\mathcal{A} = \langle Q, \delta, I, T\rangle$ où Q est un ensemble fini d'états, $\delta \subseteq Q \times A \times Q$ est l'ensemble des transitions, $I \subseteq Q$ est l'ensemble des états initiaux et $T \subseteq Q$ est l'ensemble des états terminaux. La transition $(p, a, q) \in \delta$ est notée par la flèche $p \xrightarrow{a} q$. Un **calcul** c de $\mathcal{A}$ est une séquence finie de transitions qui forment un chemin dans le graphe de $\mathcal{A}$, $p_0 \xrightarrow{a_1} p_1 \xrightarrow{a_2} p_2 \cdots \xrightarrow{a_n} p_n$: on note un tel calcul $c = p_0 \xrightarrow{a_1 a_2 \cdots a_n} p_n$. L'étiquette de c est le mot $a_1 a_2 \cdots a_n$ de A^*. Le calcul est acceptant si

$p_0 \in I$ et $p_n \in T$. Le **langage** de $\mathcal{A}$ est le sous-ensemble $\mathcal{L}(\mathcal{A})$ de A^* qui contient l'ensemble des étiquettes des calculs acceptants de $\mathcal{A}$. L'automate $\mathcal{A}$ est dit **déterministe** si I possède un unique élément et pour tout $p \in Q$ et $a \in A$, il existe au plus un état $q \in Q$ tel que $(p, a, q) \in T$.

Pour coder une valuation σ des variables $\{x_1, \ldots, x_k\}$, on introduit l'alphabet $A_k = \{0,1\}^k$. Un mot de A_k^* peut donc s'écrire $(u_1, \ldots, u_k)$ avec $u_i \in \{0,1\}^*$: chaque mot $u \in \{0,1\}^*$ représente alors l'entier dont il est l'écriture binaire (avec le bit de poids fort le plus à gauche). Ainsi, le mot $(1,0)(1,1)(0,0)(0,1)(0,0)$ de A_2^* peut s'écrire $(11000, 01010)$ et représente le couple d'entiers $(24, 10)$.

4. Construire des automates dont le langage est l'ensemble des tuples représentant les valuations apparaissant dans la sémantique des formules $x_1 = 0$, $x_1 = 1$ et $x_1 = x_2 + x_3$.

5. En supposant connus des automates $\mathcal{A}_1$ et $\mathcal{A}_2$ représentant des formules φ_1 et φ_2, expliquer comment obtenir des automates représentant les formules $\neg\varphi_1$ et $(\varphi_1 \wedge \varphi_2)$. Illustrer la construction pour obtenir l'automate représentant la formule $x_1 = x_2 + x_3 \wedge \neg(x_3 = 0)$.

6. En supposant connu un automate $\mathcal{A}$ représentant une formule φ avec $\{x_1, \ldots, x_k\}$ comme variables libres, expliquer comment obtenir un automate représentant la formule $\exists x_k\ \varphi$. Illustrer la construction pour obtenir l'automate représentant la formule $\exists x_3\ (x_1 = x_2 + x_3 \wedge \neg(x_3 = 0))$.

8. Proposer un algorithme permettant de décider la satisfaisabilité et la validité d'une formule de la logique de Presburger. Évaluer sa complexité. Comment se déroule votre algorithme dans le cas particulier d'une formule close, c'est-à-dire n'ayant aucune variable libre.

Numérotation de sommets dans les graphes

Question 1. Soit T un tableau de p entiers relatifs.

Décrire un algorithme qui retourne le plus petit élément, ainsi que le second plus petit élément d'un tableau en $\frac{3p}{2} + O(1)$ comparaisons. *Indication : diviser le tableau en deux.*

Terminologie Un *graphe non orienté* G est un couple (V, E) où V est un ensemble de sommets (cercles) et E est un ensemble d'arêtes (trait entre deux cercles). Le graphe G est dit *connexe* si pour toute paire de sommets $(s,\ t)$, il existe un chemin reliant s à t. Rappellons, qu'un *chemin* C allant de s à t est une suite $u_0, u_1, \cdots, u_\ell$ de sommets telle que $u_0 = s$, $u_\ell = t$, $(u_{i-1}, u_i) \in E$ pour tout $1 \leq i \leq \ell$.

Nous considérons les notations suivantes.

1. $n = |V|$, $m = |E|$.
2. Les sommets u et v sont dits *voisins* s'il y a une arête entre u et v. Le *degré* de v dans G que l'on notera $d_G(v)$ est le nombre de voisins de v.
3. $\Gamma_G(v)$ est l'ensemble des sommets voisin de v dans G.
4. $G\backslash\{v\}$ est le graphe G privé du sommet de v : son ensemble de sommets est $V\backslash\{v\}$ et son ensemble d'arêtes est $E\backslash\{(u, v) : u \in \Gamma_G(v)\}$.

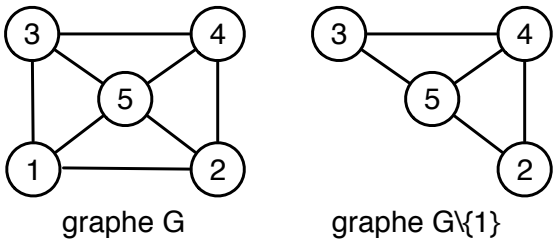


FIGURE 1 – Le graphe planaire G possède une numérotation $\mathcal{L}$ qui est 3-*dégénérée*.

Une *numérotation* des sommets $\mathcal{L} : V \to [1, \ldots, n]$ est une fonction injective qui donne des numéros distincts (des entiers) de 1 à n à chaque sommet. Nous allons nous concentrer sur une numérotation particulière des sommets. Une numérotation $\mathcal{L}$ est dite t-*dégénérée* si pour tout sommet v de V, au plus t des voisins de v ont des numéros plus grand que v, plus formellement,

$$\text{pour tout sommet } v \text{ de } G, \text{ on a } |\{x \in \Gamma_G(v) \text{ tel que } \mathcal{L}(x) > \mathcal{L}(v)\}| \leq t.$$

Nous allons chercher la plus petite valeur $t^*(G)$ telle que G possède une numérotation $t^*(G)$-*dégénérée*.

Question 2. Montrer que $t^*(G) = max(d_G(u_0), t^*(G\backslash\{u_0\}))$ avec u_0 un sommet de G de plus petit degré dans G.

Question 3. Un *arbre* $G = (V, E)$ est un graphe sans cycle (un cycle est une suite d'arêtes consécutives (chaine) dont les deux sommets extrémités sont identiques). Donner la valeur de $t^*(G)$ lorsque G est un arbre.

Question 4. Décrire un algorithme qui, étant donné un graphe G, retourne $t^*(G)$ et la numérotation $\mathcal{L}$ qui est $t^*(G)$-*dégénérée*. Evaluer la complexité de cet algorithme.

Question 5. Soit k un entier. Donner le nombre d'arêtes maximum d'un graphe G de n sommets tel que $t^*(G) = k$ et $n \geq k$.

Colorer un graphe G signifie attribuer une couleur à chacun des sommets de G telle qu'aucune arête de G a deux extrémités de la même couleur.

Question 6. Montrer que si G possède une numérotation $\mathcal{L}$ qui est t-*dégénérée*, alors G peut être coloré en $t + 1$ couleurs. En déduire un algorithme et donner sa complexité.

Un graphe $G = (V, E)$ est *régulier* si tous ses sommets ont le même nombre de voisins.

Notons par la suite $\Delta(G)$ le degré maximum du graphe ($\Delta(G) = \max\{d_G(v) : v \in V\}$).

Question 7. Soit G un graphe connexe. Montrer que G est *régulier* si et seulement si $t^*(G) = \Delta(G)$.

Question 8. Montrer que si G n'est pas régulier, alors G peut être coloré en $\Delta(G)$ couleurs.

Numérotation de sommets dans les graphes

Terminologie Un *graphe non orienté* G est un couple (V, E) où V est un ensemble de sommets (cercles) et E est un ensemble d'arêtes (trait entre deux cercles). Le graphe G est dit *connexe* si pour toute paire de sommets $(s,\ t)$, il existe un chemin reliant s à t. Rappellons, qu'un *chemin* C allant de s à t est une suite $u_0, u_1, \cdots, u_\ell$ de sommets telle que $u_0 = s$, $u_\ell = t$, $(u_{i-1}, u_i) \in E$ pour tout $1 \leq i \leq \ell$.

Nous considérons les notations suivantes.

1. $n = |V|$, $m = |E|$.
2. Les sommets u et v sont dits *voisins* s'il y a une arête entre u et v. Le *degré* de v dans G que l'on notera $d_G(v)$ est le nombre de voisins de v.
3. $\Gamma_G(v)$ est l'ensemble des sommets voisin de v dans G.
4. $G\backslash\{v\}$ est le graphe G privé du sommet de v : son ensemble de sommets est $V\backslash\{v\}$ et son ensemble d'arêtes est $E\backslash\{(u, v) : u \in \Gamma_G(v)\}$.

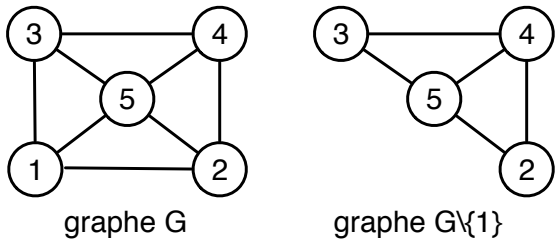


FIGURE 1 – Le graphe planaire G possède une numérotation $\mathcal{L}$ qui est 3-*dégénérée*.

Une *numérotation* des sommets $\mathcal{L} : V \to [1, \ldots, n]$ est une fonction injective qui donne des numéros distincts (des entiers) de 1 à n à chaque sommet. Nous allons nous concentrer sur une numérotation particulière des sommets. Une numérotation $\mathcal{L}$ est dite t-*dégénérée* si pour tout sommet v de V, au plus t des voisins de v ont des numéros plus grand que v, plus formellement,

$$\text{pour tout sommet } v \text{ de } G, \text{ on a } |\{x \in \Gamma_G(v) \text{ tel que } \mathcal{L}(x) > \mathcal{L}(v)\}| \leq t.$$

Nous allons chercher la plus petite valeur $t^*(G)$ telle que G possède une numérotation $t^*(G)$-*dégénérée*.

Question 1. Montrer que $t^*(G) = max(d_G(u_0), t^*(G\backslash\{u_0\}))$ avec u_0 un sommet de G de plus petit degré dans G.

Soit u_0 un sommet de G de plus petit degré dans G.
Prouvons que $t^*(G) \geq max(d_G(u_0), t^*(G\backslash\{u_0\}))$.
Soit $\mathcal{L}^*$ une numérotation telle qu'elle est $t^*(G)$-*dégénérée*. Il suffit de considérer le sommet u numéroté 1. Tous ses voisins ont un numéro plus grand que 1 : pour tout sommet v voisin de u, on a $\mathcal{L}^*(v) > \mathcal{L}^*(u)$. Donc $t^*(G) \geq d_G(u) \geq d_G(u_0)$.

Nous allons transformer $\mathcal{L}^*$ en $\mathcal{L}^{'}$ de telle façon que l'ordre soit conservé à l'exception de u_0 qui prend la valeur 1. Plus formellement,

$$\mathcal{L}^{'}(u) = \begin{cases} 1 \ si \ u = u_0 \\ \mathcal{L}^*(u) + 1 \ si \ \mathcal{L}^*(u) < \mathcal{L}^*(u_0) \\ \mathcal{L}^*(u) \ sinon \end{cases}$$

Il suffit de remarquer que la numérotation $\mathcal{L}^{'}$ est une numérotation du graphe $G\backslash\{u_0\}$. Ainsi elle est aussi t-*dégénérée* pour le graphe $G\backslash\{u_0\}$ avec $t^*(G) \geq t$. Donc $t^*(G) \geq t \geq t^*(G\backslash\{u_0\})$.
Donc, on peut conclure que

$$t^*(G) \geq max(d_G(u_0), t^*(G\backslash\{u_0\}))$$

Prouvons que $t^*(G) \leq max(d_G(u_0), t^*(G\backslash\{u_0\}))$. Pour cela il suffit de transformer *de la bonne façon* une numérotation $\mathcal{L}^{'}$ du graphe $G\backslash\{u_0\}$ qui est $t^*(G\backslash\{u_0\})$-*dégénérée*.
Soit $\mathcal{L}^{'}$ une numérotation de sommets de $G\backslash\{u_0\}$ telle qu'elle est $t^*(G\backslash\{u_0\})$-*dégénérée*. Nous allons construire une numérotation $\mathcal{L}$ de sommets de G

$$\mathcal{L}(u) = \begin{cases} 1 \ si \ u = u_0 \\ \mathcal{L}^{'}(u) + 1 \ sinon \end{cases}$$

Il suffit de voir que la numérotation $\mathcal{L}$ est aussi x-*dégénérée* pour le graphe G avec $x = max(d_G(u_0), t^*(G\backslash\{u_0\}))$.

$$t^*(G) \leq max(d_G(u_0), t^*(G\backslash\{u_0\}))$$

Question 2. Un *arbre* $G = (V, E)$ est un graphe sans cycle (un cycle est une suite d'arêtes consécutives (chaine) dont les deux sommets extrémités sont identiques). Donner la valeur de $t^*(G)$ lorsque G est un arbre.

Il suffit de voir qu'un arbre possède une numérotation 1-*dégénérée*. Pour cela, il suffit d'observer qu'un arbre privé d'une feuille (sommet de degré 1) reste toujours un arbre. Pour construire une numérotation $\mathcal{L}$ 1-*dégénérée* sur un arbre G, il suffit de trouver une feuille v et de définir $\mathcal{L}(v)$ à 1. Ensuite, nous supprimons le sommet v, et nous itérons le processus.

Question 3. Décrire un algorithme qui, étant donné un graphe G, retourne $t^*(G)$ et une numérotation $\mathcal{L}$ qui est $t^*(G)$-*dégénérée*. Évaluer la complexité de cet algorithme.

Notons n le nombre de sommets du graphe G.

Entrée : un graphe G
Sortie : une numérotation $\mathcal{L}$ qui est $t^*(G)$-dégénérée

1. $\mathcal{L} \leftarrow \emptyset$

2. pour i allant de 1 à n faire
   (a) calculer $\mathcal{D}(G)$ où l'ensemble est l'ensemble de sommets de plus petit degré dans G ;
   (b) extraire un sommet v_i de $\mathcal{D}(G)$;
   (c) $\mathcal{L}(v_i) = i$
   (d) $G = G \backslash \{v\}$
3. Retourner $\mathcal{L}$

Complexité : Au total $O(n^2)$ opérations.

La boucle for est exécutés n fois. Maintenant focalisons nous sur les instructions 2.(a), 2.(b), 2.(c) et 2.(d). Remarquons que les instructions 2.(b), 2.(c) peuvent se faire en $O(1)$ opérations. L'instruction 2.(d) peut être couteuse si le graphe est recopié. Sinon, il suffit de supprimer toutes les arêtes incidentes à v ($O(n)$ opérations si le graphe est codé en utilisant une matrice d'adjacence).

Focalisons nous sur l'instruction 2.(a). Calculer les degrés de tous sommets d'un graphe peuvent se faire en $O(n^2)$ opérations si le graphe est codé en utilisant une matrice d'adjacence ou en $O(|E|)$ opérations au total si le graphe est codé en utilisant sa liste d'adjacente. Si pour chaque itération de la boucle nous devons calculer les degrés de tous les sommets, exécuter n fois l'instruction 2.(a) nécessite $O(n^3)$ opérations (ou $O(|E|n)$ opérations en fonction du codage). Afin de réduire la complexité, on peut réaliser l'astuce suivante :

1. Tout d'abord, il suffit de précalculer le degré de tous les sommets et de les stocker dans un tableau.
2. Trouver le sommet de degré minimum revient à trouver la valeur minimum dans un tableau ($O(p)$ opérations avec p le nombre d'éléments dans le tableau).
3. Ensuite lorsqu'on supprime un sommet v du graphe courant on réactualise le degré de tous les sommets voisin de v en décrémentant de 1 sa valeur. ($O($ degré de $v)$ opérations). Au total, pour toutes les itérations de la boucle for, ces instructions nécessitent $O(|E|)$ opérations) Remarquons que cette opération peut remplacer l'instruction 2.(d)

Question 4. Soit k un entier. Donner le nombre d'arêtes maximum d'un graphe G de n sommets tel que $t^*(G) = k$ et $n \geq k + 1$.

Rappellons qu'une *clique* est un graphe dans lequel chaque paire de sommets est liée par une arête.

— Cas $n = k + 1$: c'est le cas où G possède moins de k sommets. G est une clique ayant $\frac{k(k+1)}{2}$ arêtes.

— Cas $n > k + 1$. Nous allons construire G de la façon suivante :

1. tous les sommets sont numérotés de 1 à n ;

2. le sommet de numéro i avec $1 \leq i \leq n-(k+1)$, a les sommets $i+1, \ldots, i+k$ comme voisin ;
3. tous les sommets de $j = n-k+1$ à n forment une clique dans G.

Donc le nombre d'arêtes est $(n-(k+1))k + \frac{k(k+1)}{2} = (n-k)k + \frac{k(k-1)}{2}$.

Colorer un graphe G signifie attribuer une couleur à chacun des sommets de G telle qu'aucune arête de G a deux extrémités de la même couleur.

Question 5. Montrer que si G possède une numérotation $\mathcal{L}$ qui est t-*dégénérée*, alors G peut être coloré en $t+1$ couleurs. En déduire un algorithme et donner sa complexité.

Nous allons introduire les notations suivantes :

1. $[t+1]$: l'ensemble de $t+1$ premiers entiers
2. $\Gamma_G(u)$: l'ensemble des voisins de u dans G.
3. u_i : le sommet de numéro i ;

Entrée : un graphe G et une numérotation $\mathcal{L}$ qui est t-dégénérée
Sortie : un tableau C tel que $C[u]$ représente la couleur du sommet u

1. $C[u_n] = 1$ avec $\mathcal{L}(u_n) = 1$
2. pour i allant de $n-1$ à 1 faire
   (a) notons u_i le sommet tel que $\mathcal{L}(u_i) = i$
   (b) Calculons $X = [t+1] \backslash \{C[u] : j > i \wedge \mathcal{L}(u) = j \wedge u \in \Gamma_G(u_i)\}$
   (c) $C[u_i] = c$ tel que $c \in X$
3. Retourner C

Complexité : On fait n fois la boucle. En fonction de la façon de coder le graphe, l'instruction (2.b) peut se faire en t opérations (au pire des cas). En effet, il suffit que le graphe soit codé en utilisant la liste d'adjacente et la numérotation $\mathcal{L}$ pour les sommets. Si ce n'est pas le cas, l'instruction (2.b) peut se faire en $O(n-i)$ opérations à la $i$ème itération. En conclusion, la complexité est soit $O(tn)$ ou $O(n^2)$ selon la représentation du graphe.

Un graphe $G = (V, E)$ est *régulier* si tous ses sommets ont le même nombre de voisins. Notons par la suite $\Delta(G)$ le degré maximum du graphe ($\Delta(G) = \max\{d_G(v) : v \in V\}$).

Question 6. Soit $G = (V, E)$ un graphe connexe. Montrer que G est *régulier* si et seulement si $t^*(G) = \Delta(G)$.

Si G est k-régulier, alors le graphe G possède une numérotation k-dégénéré (car tous les sommets sont de même degré).

Tout d'abord nous allons calculer la valeur $t^*(G)$ et une numérotation $\mathcal{L}$ qui est $t^*(G)$-dégénérée en utilisant la question **3**. Cela signifie que le sommet u avec $\mathcal{L}(u) = i$ est un sommet de plus petit degré dans $G_i = (V_i, E_i)$ tel que $V_i = \{v \in V : \mathcal{L}(u) > i\}$ et $E_i = E \cap (V_i \times V_i)$.

Rappellons que $t^*(G) = \Delta(G)$. Cela signifie qu'il existe un sommet v_i tel qu'il a $\Delta(G)$ voisins de numéro plus grand dans G. Considérons le sommet v qui

satisfait cette propriété de numéro le plus petit. nous avons que pour tout sommet w tel que $\mathcal{L}(w) > \mathcal{L}(v)$, le degré de w dans $G_{\mathcal{L}(v)}$ est $\Delta(G)$.

Si $\mathcal{L}(v) > 1$, la connexité de G implique l'existence d'une arête (x, y) avec $\mathcal{L}(x) \geq \mathcal{L}(v) > \mathcal{L}(y)$. Cela signifie que x est de degré au moins $\Delta(G) + 1$ dans G. Ceci est en contradiction avec la définition de $\Delta(G)$. Donc $\mathcal{L}(v) = 1$.

$\mathcal{L}(v) = 1$ implique que le degré de tous les autres sommets est $\Delta(G)$. Donc G est régulier.

Question 7. Montrer que si G n'est pas régulier, alors G peut être coloré en $\Delta(G)$ couleurs.

On en déduit des deux précédentes questions.

Logique de Presburger - Correction

La **logique de Presburger** est l'ensemble des formules logiques générées par la grammaire suivante :

$$\varphi ::= x = 0 \mid x = 1 \mid z = x + y \mid \neg\varphi \mid (\varphi \wedge \varphi) \mid \exists x\, \varphi$$

où x, y, z sont des symboles de variables, choisis parmi un ensemble infini de symboles. Une variable de φ est dite **libre** si elle apparaît dans une formule sans être quantifiée préalablement. Par exemple, dans la formule $\exists y\ y = x + x$, la variable x est libre, mais pas la variable y. On note $\mathrm{Free}(\varphi)$ l'ensemble des variables libres de φ.

1. Donner une définition inductive de l'ensemble $\mathrm{Free}(\varphi)$ des variables libres d'une formule φ de la logique de Presburger.

- $\mathrm{Free}(x = 0) = \mathrm{Free}(x = 1) = \{x\}$;
- $\mathrm{Free}(z = x + y) = \{x, y, z\}$;
- $\mathrm{Free}(\neg\varphi) = \mathrm{Free}(\varphi)$;
- $\mathrm{Free}(\varphi \wedge \varphi') = \mathrm{Free}(\varphi) \cup \mathrm{Free}(\varphi')$;
- $\mathrm{Free}(\exists x\, \varphi) = \mathrm{Free}(\varphi) \setminus \{x\}$.

Une **valuation** des variables libres de φ est une application $\sigma \colon \mathrm{Free}(\varphi) \to \mathbf{N}$ associant à chaque variable une valeur dans les entiers naturels. La **sémantique** d'une formule φ est un ensemble de valuations des variables libres de φ vers des entiers naturels, qu'on note $[\![\varphi]\!]$ et qui est défini récursivement par :

- $[\![x = 0]\!]$ est le singleton contenant l'unique valuation σ associant x à la valeur 0 ;
- $[\![x = 1]\!]$ est le singleton contenant l'unique valuation σ associant x à la valeur 1 ;
- $[\![z = x + y]\!]$ est l'ensemble contenant l'ensemble des valuations σ vérifiant $\sigma(z) = \sigma(x) + \sigma(y)$;
- $[\![\neg\varphi]\!]$ est le complémentaire de l'ensemble $[\![\varphi]\!]$;
- $[\![\varphi \wedge \varphi']\!]$ est l'ensembles des valuations $\sigma \colon \mathrm{Free}(\varphi \wedge \varphi') \to \mathbf{N}$ telles que la restriction de σ au domaine $\mathrm{Free}(\varphi)$ appartient à $[\![\varphi]\!]$, et la restriction de σ au domaine $\mathrm{Free}(\varphi')$ appartient à $[\![\varphi']\!]$;
- $[\![\exists x\, \varphi]\!]$ est l'ensemble des valuations $\sigma \colon \mathrm{Free}(\exists x\, \varphi) \to \mathbf{N}$ telle qu'il existe un entier n tel que la valuation $\sigma' \colon \mathrm{Free}(\varphi) \to \mathbf{N}$ définie par $\sigma'(x) = n$ et $\sigma'(y) = \sigma(y)$ pour tout $y \in \mathrm{Free}(\varphi) \setminus \{x\}$ appartient à $[\![\varphi]\!]$.

2. Décrire la sémantique de la formule $\exists z\ (z = 1 \wedge y = x + z)$.

$[\![\exists z\ (z = 1 \wedge y = x + z)]\!]$ est l'ensemble des valuations $\sigma \colon \{x, y\} \to \mathbf{N}$ telles que $\sigma(y) = \sigma(x) + 1$.

3. Écrire une formule de la logique de Presburger dont la sémantique est l'ensemble des valuations $\sigma \colon \{x\} \to \mathbf{N}$ telles que $\sigma(x)$ est congru à 1 modulo 3.

On aurait envie d'écrire la formule $\exists y\ x = 3y + 1$. Une fois écrite dans la logique de Presburger cela donne :

$$\exists y \exists x_1 \exists y_2 \exists y_3\ (x = y_3 + x_1 \wedge x_1 = 1 \wedge y_2 = y + y \wedge y_3 = y_2 + y)$$

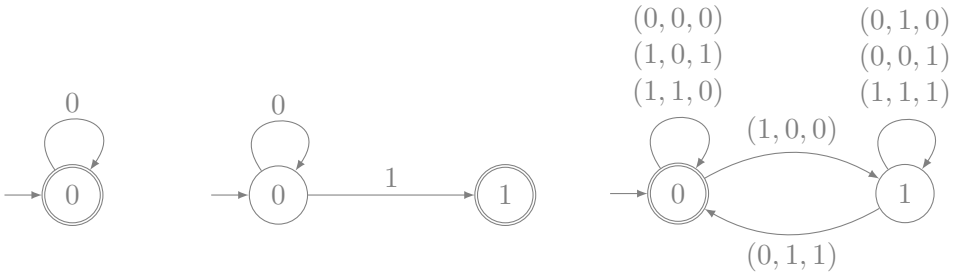
Une formule φ est dite **satisfaisable** si sa sémantique $[\![\varphi]\!]$ est non vide. Elle dite **valide** si sa sémantique contient toutes les valuations de ses variables libres, autrement dit si $[\![\neg\varphi]\!]$ est l'ensemble vide. On cherche maintenant à prouver la décidabilité de la satisfaisabilité (ou de la validité) d'une formule de la logique de Presburger. Pour ce faire, on utilise des automates dont on rappelle d'abord les définitions.

Un **automate fini** $\mathcal{A}$ sur l'alphabet A est la donnée d'un quadruplet $\mathcal{A} = \langle Q, \delta, I, T\rangle$ où Q est un ensemble fini d'états, $\delta \subseteq Q \times A \times Q$ est l'ensemble des transitions, $I \subseteq Q$ est l'ensemble des états initiaux et $T \subseteq Q$ est l'ensemble des états terminaux. La transition $(p, a, q) \in \delta$ est notée par la flèche $p \xrightarrow{a} q$. Un **calcul** c de $\mathcal{A}$ est une séquence finie de transitions qui forment un chemin dans le graphe de $\mathcal{A}$, $p_0 \xrightarrow{a_1} p_1 \xrightarrow{a_2} p_2 \cdots \xrightarrow{a_n} p_n$: on note un tel calcul $c = p_0 \xrightarrow{a_1 a_2 \cdots a_n} p_n$. L'étiquette de c est le mot $a_1 a_2 \cdots a_n$ de A^*. Le calcul est acceptant si $p_0 \in I$ et $p_n \in T$. Le **langage** de $\mathcal{A}$ est le sous-ensemble $\mathcal{L}(\mathcal{A})$ de A^* qui contient l'ensemble des étiquettes des calculs acceptants de $\mathcal{A}$. L'automate $\mathcal{A}$ est dit **déterministe** si I possède un unique élément et pour tout $p \in Q$ et $a \in A$, il existe au plus un état $q \in Q$ tel que $(p, a, q) \in T$.

Pour coder une valuation σ des variables $\{x_1, \ldots, x_k\}$, on introduit l'alphabet $A_k = \{0, 1\}^k$. Un mot de A_k^* peut donc s'écrire $(u_1, \ldots, u_k)$ avec $u_i \in \{0, 1\}^*$: chaque mot $u \in \{0, 1\}^*$ représente alors l'entier dont il est l'écriture binaire (avec le bit de poids fort le plus à gauche). Ainsi, le mot $(1, 0)(1, 1)(0, 0)(0, 1)(0, 0)$ de A_2^* peut s'écrire $(11000, 01010)$ et représente le couple d'entiers $(24, 10)$.

4. Construire des automates dont le langage est l'ensemble des tuples représentant les valuations apparaissant dans la sémantique des formules $x_1 = 0$, $x_1 = 1$ et $x_1 = x_2 + x_3$.

Les trois automates sont :

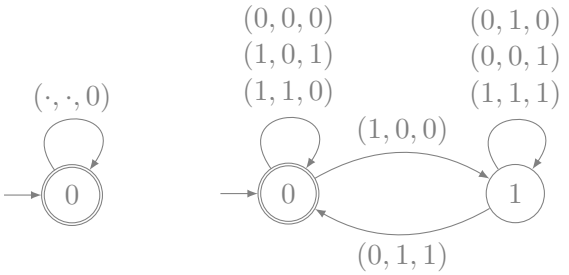


5. En supposant connus des automates $\mathcal{A}_1$ et $\mathcal{A}_2$ représentant des formules φ_1 et φ_2, expliquer comment obtenir des automates représentant les formules $\neg\varphi_1$ et $(\varphi_1 \wedge \varphi_2)$. Illustrer la construction pour obtenir l'automate représentant la formule $x_1 = x_2 + x_3 \wedge \neg(x_3 = 0)$.

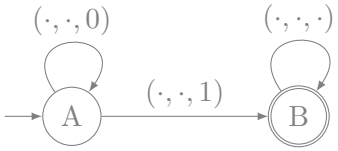
On obtient l'automate pour $\neg\varphi_1$ en complémentant l'automate $\mathcal{A}_1$: pour cela, on peut compléter et déterminiser l'automate $\mathcal{A}_1$, puis inverser les états terminaux et non-terminaux.

Considérons ensuite le cas de la conjonction $(\varphi_1 \wedge \varphi_2)$. Soient $\mathcal{A}_1 = \langle Q_1, \delta_1, I_1, T_1\rangle$ et $\mathcal{A}_2 = \langle Q_2, \delta_2, I_2, T_2\rangle$ des automates respectifs pour φ_1 et φ_2. Chacun fonctionne sur les variables libres de φ_1 et φ_2 respectivement. On peut commencer par étendre les alphabets pour qu'ils soient tous deux sur le même ensemble $\mathrm{Free}(\varphi_1 \wedge \varphi_2) = \mathrm{Free}(\varphi_1) \cup \mathrm{Free}(\varphi_2)$ de variables. Pour cela, remarquons qu'ajouter une variable x revient à ajouter dans le tuple de chaque lettre une coordonnée supplémentaire, en dupliquant chaque transition pour qu'elle puisse se faire avec la valeur 0 et avec la valeur 1 pour la nouvelle variable x. On suppose donc désormais que les automates

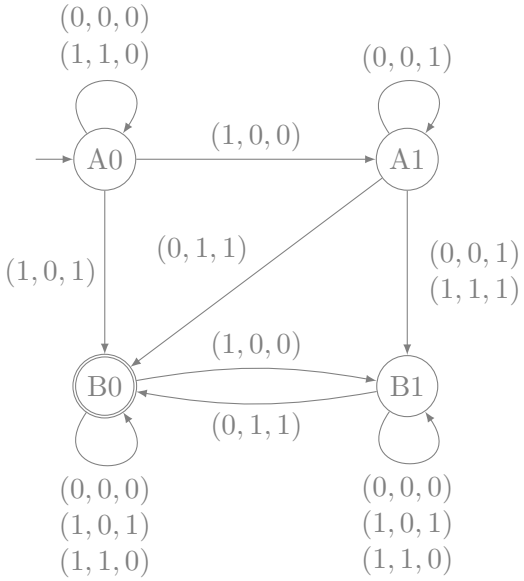
$\mathcal{A}_1$ et $\mathcal{A}_2$ ci-dessus sont sur le même alphabet. L'automate pour $(\varphi_1 \wedge \varphi_2)$ est alors obtenu par un produit synchrone des deux automates : $\mathcal{A} = \langle Q_1 \times Q_2, \delta, I_1 \times I_2, T_1 \times T_2 \rangle$ avec $\delta = \{((p_1, p_2), a, (q_1, q_2)) \mid (p_1, a, q_1) \in \delta_1, (p_2, a, q_2) \in \delta_2\}$.
Pour l'exemple, on utilise les automates de la question précédente pour $x_3 = 0$ et pour $x_1 = x_2 + x_3$, en étendant les deux sur l'alphabet $\{0,1\}^3$ (en utilisant un $\cdot$ dans une composante lorsque la valeur peut être à la fois 0 et 1) :



On complète et complémente le premier pour obtenir l'automate pour $\neg(x_3 = 0)$:

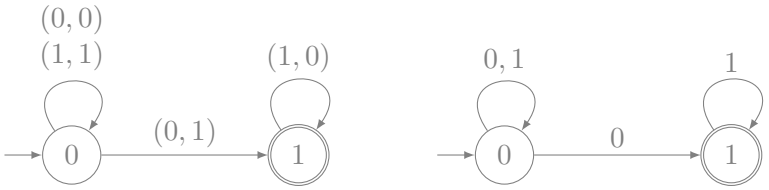


Finalement, on réalise le produit des deux automates pour obtenir celui de $\neg(x_3 = 0) \wedge x_1 = x_2 + x_3$:

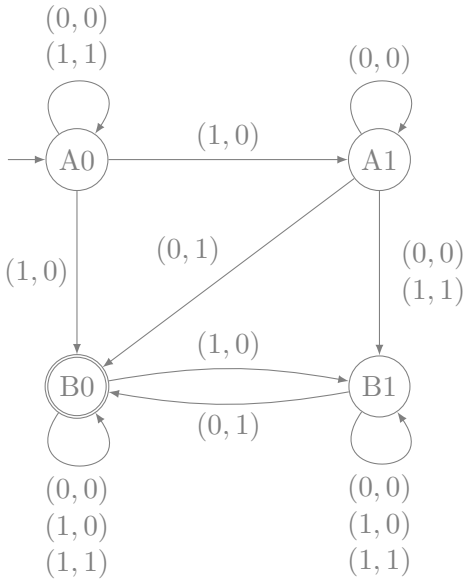


6. En supposant connu un automate $\mathcal{A}$ représentant une formule φ avec $\{x_1, \ldots, x_k\}$ comme variables libres, expliquer comment obtenir un automate représentant la formule $\exists x_k\ \varphi$. Illustrer la construction pour obtenir l'automate représentant la formule $\exists x_3\ (x_1 = x_2 + x_3 \wedge \neg(x_3 = 0))$.

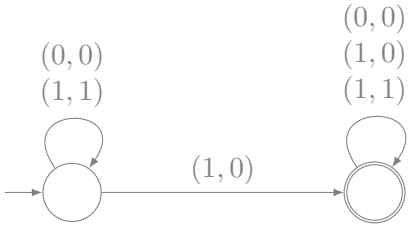
Notons $\mathcal{A} = \langle Q, \delta, I, T\rangle$ l'automate sur l'alphabet A_k^* représentant la formula φ. Soit $\mathcal{A}' = \langle Q, \delta', I', T\rangle$ l'automate sur l'alphabet A_{k-1}^* obtenu à partir de $\mathcal{A}$ en supprimant la dernière composante des lettres lues dans δ : $\delta' = \{(p, (a_1, \ldots, a_{k-1}), q) \mid (p, (a_1, \ldots, a_{k-1}, 0), q) \in \delta$ ou $(p, (a_1, \ldots, a_{k-1}, 1), q) \in \delta\}$ et I' est l'ensemble des états de Q qui peuvent être atteints à partir de I en lisant uniquement des lettres $(0, \ldots, 0, 0)$ ou $(0, \ldots, 0, 1)$. Le changement d'états initiaux permet de considérer le cas où les valuations reconnues ont une $k$ème coordonnée supérieure aux autres coordonnées. Par exemple, l'automate à gauche ci-dessous reconnaît l'ensemble des paires (x_1, x_2) avec $x_2 = x_1 + 1$. En projetant seulement x_2 des lettres (sans modifier les états initiaux), on obtient l'automate de droite qui doit donc reconnaitre l'ensemble des entiers naturels, mais qui est incorrect puisqu'il ne reconnait que des mots contenant au moins une lettre 0.



Pour obtenir l'automate pour la formule $\exists x_3\ (x_1 = x_2 + x_3 \wedge \neg(x_3 = 0))$, on applique la projection à l'automate obtenu dans la question précédente :



Cet automate peut être éventuellement déterminisé et minimisé pour obtenir l'automate équivalent suivant, qui représente bien le fait que les variables $x_1 > x_2$ sont différentes :



8. Proposer un algorithme permettant de décider la satisfaisabilité et la validité d'une formule de la logique de Presburger. Évaluer sa complexité. Comment se déroule votre algorithme dans le cas particulier d'une formule close, c'est-à-dire n'ayant aucune variable libre.

À partir d'une formule, on construit par induction un automate représentant la formule. Chaque négation nécessite la déterminisation de l'automate, de sorte que la complexité est non-élémentaire en fonction du nombre de négation n dans la formule, c'est-à-dire de l'ordre de $2^{O(2^{\cdot^{\cdot^{\cdot^{2^{O(1)}}}}})}$ avec une tour de hauteur n. Tester la satisfaisabilité revient à vérifier si un état terminal est accessible depuis un état initial : il suffit de réaliser un parcours du graphe de l'automate. Tester la validité peut se faire en testant la non satisfaisabilité de l'automate pour la négation de la formule. Dans les deux cas, la complexité est donc non-élémentaire.

La sémantique d'une formule close est soit l'ensemble vide (formule non satisfaisable), soit le singleton contenant l'unique valuation sur un ensemble vide de variables (formule valide, et donc satisfaisable). Les automates obtenus pour une telle formule close sont des automates avec ε-transitions : une fois déterminisés, on peut tester si la formule est satisfaisable (et valide, puisque c'est équivalent dans ce cas-là) en vérifiant si l'unique état est à la fois initial et final.

Inspiré par le théorème 3.63 du livre *Langages formels : Calculabilité et complexité* d'Olivier Carton.