Fondements de l'informatique. Examen
Durée: 3h

(corrigé)

Les 4 parties sont indépendantes, et peuvent être traitées dans un ordre quelconque. On pourra admettre le résultat d'une question pour passer aux questions suivantes. On pourra utiliser tous les résultats et les théorèmes démontrés ou énoncés en cours ou en petite classe ou dans le polycopié sans chercher à les redémontrer.

Il est possible d'avoir la note maximale sans répondre à toutes les questions. La difficulté des questions n'est pas une fonction linéaire ni croissante de leur numérotation.

La ***qualité et clarté de votre argumentation et de votre rédaction*** *sera une partie importante de votre évaluation.*

1 Machines de Turing

Question 1.1. *Parmi les questions suivantes, lesquelles sont décidables ? Récursivement énumérable ? dans P ? Justifier votre réponse.*

- *Déterminer si une machine de Turing M est telle qu'elle accepte un langage qui contient des mots d'au moins* 10 *longueurs différentes.*
- *Déterminer si un programme Python contient au moins* 10 *lignes de codes.*

Solution :

- C'est indécidable : il s'agit d'une application directe du théorème de Rice puisque c'est une propriété de langage. Il y a une machine qui reconnaît le langage vide (qui n'a pas cette propriété), et une machine de Turing qui accepte tous les mots (et qui a cette propriété), et donc la propriété est bien non triviale. Il est récursivement énumérable car car il suffit de simuler la machine M sur de plus en plus d'entrées en parallèle et d'accepter si et seulement si la simulation accepte au moins 10 mots de longueurs différentes. Il n'est pas dans P car les langages de P sont décidables.
- C'est dans P (et donc décidable et récursivement énumérable) : il suffit de parcourir et le code et de compter les lignes de code.

□

On admettra dans la suite qu'il existe une machine de Turing universelle implémentable en moins de 10 000 lignes de Python.

Question 1.2. *Montrer que le langage suivant est indécidable mais récursivement énumérable : Déterminer si un programme Python exécute au moins* 100 000 *lignes de code différentes (c'est à dire à des positions différentes dans le programme) sur une entrée donnée (et peut s'arrêter ou non).*

Solution : C'est indécidable : on réduit depuis le problème universel. On prend une machine de Turing universelle U et on traduit son code en un programme Python P_U : le code obtenu a moins de 10 000 lignes de codes d'après notre hypothèse ci-dessus. On construit alors un programme python P qui lit l'entrée puis exécute P_U sur cette entrée puis, si P_U termine et accepte, exécute 100 000 lignes de codes que l'on ajoute à la fin du fichier. Par exemple :

- lire l'entrée,
- exécuter P_U,
- $x = 1$,

— $x = 2$,
— ... encore 1000 000 fois ...
— $x = 1000\,003$,
— fin.

Etant donné une machine M et un mot w, on construit le programme Python P ci-dessus et l'entrée $(\langle M \rangle, w)$ encodée pour la machine universelle U. On observe alors que :

— Si M accepte w, alors la simulation par U sur l'entrée (M, w) termine et accepte donc le programme Python P va exécuter les 1000 003 lignes de codes différentes à la fin (et s'arrêter).
— Si M rejette w ou ne s'arrête pas, la simulation par U sur l'entrée (M, w) va soit ne jamais terminer, soit rejeter. Dans tous les cas, le programme P n'exécutera pas les 1000 003 lignes de codes différentes à la fin. Or le code de P_U ne prend que 100 000 lignes de code donc le programme n'exécute pas plus de 100 003 lignes de codes différentes.

On a donc bien réduit le programme universal au problème de savoir si un programme Python exécute au moins 100 000 lignes de code différentes sur une entrée donnée (et peut s'arrêter ou non). Donc ce problème est indécidable. Par ailleurs, il est récursivement énumérable car on peut le simuler en mémorisant quelles lignes ont déjà été exécutée puis accepter dès qu'on a exécuté plus que 100 000 lignes de code différentes. □

Question 1.3. *Montrer que le langage suivant n'est ni décidable, ni récursivement énumérable : Déterminer si une machine de Turing M est telle qu'elle accepte un langage qui contient au moins un mot de chaque longueur.*

Solution : C'est indécidable : il s'agit d'une application directe du théorème de Rice puisque c'est une propriété de langage. Il y a une machine qui reconnaît le langage vide (qui n'a pas cette propriété), et une machine de Turing qui accepte tous les mots (et qui a cette propriété), et donc la propriété est bien non triviale.

On va montrer qu'il n'est pas récursivement énumérable, en réduisant depuis le complémentaire du problème de l'arrêt. On rappelle qu'il s'agit du langage des $(\langle M \rangle, w)$ tels que M ne termine pas sur w. Sur l'entrée $(\langle M \rangle, w)$, on construit le programme M' qui sur l'entrée u, simule $M(w)$ pendant $|u|$ étapes et accepte si et seulement si la simulation n'a pas terminé après $|u|$ étapes. On vérifie que :

— Si M ne termine pas sur w alors pour tout u, $M(u)$ accepte car la simulation n'est jamais terminée après $|u|$ étapes. En particulier, M accepte tous les mots donc au moins un mot de chaque longueur.
— Si M termine sur w après N étapes, $M(u)$ rejette pour $|u| \geqslant N$ car la simulation termine après $|u|$ étapes. En particulier, M n'accepte pas des mots de toutes les longueurs possibles.

□

2 Problèmes de tuiles

On considère un ensemble fini $\mathcal{T}$, dont les élements sont appelés des *tuiles*. On fixe des relations de compatibilité entre tuiles : on fixe $H \subseteq \mathcal{T}^2$ qui indique les compatibilités horizontales, et $V \subseteq \mathcal{T}^2$ les compatibilités verticales.

On dit qu'une partie $\mathcal{E}$ du plan $\mathbb{Z}^2$ peut être *résolue* par $\mathcal{T}$ s'il existe une façon de placer les tuiles sur le plan qui respecte ces relations de compatibilité : Formellement, s'il existe une fonction $f : \mathcal{E} \to \mathcal{T}$ telle que, pour tout $(m, n) \in \mathcal{E}$,

1. si $(m+1, n) \in \mathcal{E}$ alors $(f(m, n), f(m+1, n)) \in H$ (compatibilité horizontale),
2. si $(m, n+1) \in \mathcal{E}$ alors $(f(m, n), f(m, n+1)) \in V$ (compatibilité verticale).

Question 2.1. *Montrer que $\mathbb{Z}^2$ peut être résolu par $\mathcal{T}$ si et seulement si tous les carrés $[\![-n,n]\!]^2$, pour $n \in \mathbb{N}$, peuvent être résolus par $\mathcal{T}$, où $[\![-n,n]\!]^2 = \{(x,y) \in \mathbb{Z}^2 : -n \leqslant x \leqslant n \wedge -n \leqslant y \leqslant n\}$.*

Solution : Pour un point $(i,j) \in \mathbb{Z}^2$, on note $h(i,j) := (i,j+1)$ et $d(i,j) := (i+1,j)$ respectivement ses voisins de haut et de droite. Pour tout $\mathcal{E} \subseteq \mathbb{Z}^2$, soit l'ensemble $S_{\mathcal{E}}$ de formules sur le domaine $\mathcal{P} := \mathbb{Z}^2 \times \mathcal{T}$ défini par :

$$S_{\mathcal{E}} := \left\{ \left(\bigvee_{t \in \mathcal{T}} p(c,t) \right) \wedge \neg \left(\bigvee_{\{t \neq t'\} \subseteq \mathcal{T}} \left[p(c,t) \wedge p(c,t') \right] \right) \mid c \in \mathcal{E} \right\}$$
$$\bigcup \left\{ \bigvee_{(t,t') \in H} \left[p(c,t) \wedge p(c',t') \right] \mid c \in \mathcal{E}, c' = d(c) \right\} \bigcup \left\{ \bigvee_{(t,t') \in V} \left[p(c,t) \wedge p(c',t') \right] \mid c \in \mathcal{E}, c' = h(c) \right\}.$$

Par construction, $\mathcal{E}$ peut être résolue par $\mathcal{T}$ si et seulement si $S_{\mathcal{E}}$ est satisfiable.

Par le théorème de compacité, $\mathbb{Z}^2$ peut donc être résolu par $\mathcal{T}$ si et seulement si tout sous-ensemble fini de $S_{\mathbb{Z}^2}$ est satisfiable.

Comme chacun de ces sous-ensembles est inclus dans un certain $S_{[\![-n,n]\!]^2}, n \in \mathbb{N}$, on conclut que $\mathbb{Z}^2$ peut être résolu par $\mathcal{T}$ si et seulement si tous les carrés $[\![-n,n]\!]^2, n \in \mathbb{N}$, peuvent être résolus par $\mathcal{T}$. □

Remarques sur la complexité

Dans la suite de cet examen, R sera toujours un anneau (généralement $\mathbb{Z}$ ou $\mathbb{Q}$). On notera $R^{n \times m}$ l'ensemble des matrices à n lignes et m colonnes et à coefficients dans R. On notera $I_n \in R^{n \times n}$ la matrice identité de taille n et 0_n la matrice nulle. On posera $\log_k^+(x) = \log_k \max(1,x)$ qui sera utile pour exprimer des complexité sans traiter le cas du mot vide à part.

On pourra utiliser sans le mentionner qu'ajouter, soustraire, diviser et multiplier des nombres rationnels se fait en temps polynomial. De même, ajouter, soustraire et multiplier des matrices à coefficients dans R ne nécessite qu'un nombre polynomial d'opérations arithmétiques dans R. Lorsqu'une matrice est inversible, son inverse est aussi calculable en nombre polynomial d'opérations arithmétiques.

3 Programmation linéaire en nombre entiers

On s'intéresse au problème suivant, qui joue un rôle important en optimisation combinatoire.

`PROG–LIN–ENTIER` :
- **Donnée**: une matrice $C \in \mathbb{Z}^{n \times m}$ et un vecteur $b \in \mathbb{Z}^n$
- **Réponse**: existe-t-il $x \in \mathbb{Z}^m$ tel que † $Cx \leqslant b$?

On admettra que ce problème est dans NP, ce qui n'est pas complètement évident. On rappelle/admettra que le problème `3-SAT` suivant est NP-complet :
- **Donnée**: un ensemble de variables $x_1, \ldots, x_m$ et une formule $F = C_1 \wedge C_2 \cdots \wedge C_n$ avec $C_i = y_{i,1} \vee y_{i,2} \vee y_{i,3}$ où pour tout i et j, $y_{i,j}$ est soit x_k , soit $\neg x_k$ pour l'un des x_k.
- **Réponse**: Décider si F est satisfiable : c'est-à-dire décider s'il existe $x_1, \ldots, x_m \in \{0,1\}$ tels que F s'évalue en vraie pour cette valeur de ses variables $x_1, \ldots, x_m$.

Question 3.1. *Montrer que le problème `3-SAT` se réduit en temps polynomial au même problème avec la contrainte en plus que chaque clause ne contient pas à la fois une variable et sa négation.*

†. Ici, $Cx \leqslant b$ veut dire que $(Cx)_i \leqslant b_i$ pour tout $i = 1, \ldots, n$.

Solution : Soit F une formule comme ci-dessus. Si une clause contient une variable et sa négation alors cette clause est toujours vraie et n'affecte pas la satisfaisabilité de la formule. On peut donc construire, en temps polynomial, la formule F' qui contient uniquement les clauses de F qui ne contiennent pas une variable et sa négation. La formule F est alors satisfiable si et seulement si F' l'est. □

Pour une formule F de `3-SAT` comme ci-dessus, on considère la matrice $C^F \in \mathbb{Z}^{n\times m}$ définie par

$$C^F_{i,j} = \begin{cases} -1 & \text{si la variable } x_j \text{ apparaît sans négation dans } C_i, \\ 1 & \text{si la variable } x_j \text{ apparaît avec négation dans } C_i, \\ 0 & \text{si la variable } x_j \text{ n'apparaît pas } C_i. \end{cases}$$

Notons que cela est bien défini grâce à la question 3.1. On pose le vecteur $b^F \in \mathbb{Z}^n$ défini par

$$b^F_i = -1 + \sum_{j=1}^{m} \max(0, C^F_{i,j}).$$

Exemple. Prenons $m = 4$ variables et $F = (x_1 \vee \neg x_2 \vee x_3) \wedge (x_1 \vee \neg x_3 \vee x_4)$ avec $n = 2$ clauses. La matrice C^F et le vecteur b^F sont alors

$$C^F = \begin{pmatrix} -1 & 1 & -1 & 0 \\ -1 & 0 & 1 & -1 \end{pmatrix}, \qquad b^F = \begin{pmatrix} 0 \\ 0 \end{pmatrix}.$$

En effet, la variable x_1 apparaît sans négation dans les deux clauses (d'où les -1 de la première colonne), x_2 apparait négativement dans C_1 et pas du tout dans C_2, x_3 apparaît positivement dans C_1 et négativement dans C_2, et enfin x_4 n'apparaît pas dans C_1 et positivement dans C_2.

Question 3.2. *Montrer que F est satisfiable si et seulement s'il existe $x \in \{0,1\}^m$ tel que $C^F x \leqslant b^F$.*

Solution : Soit $x_1, \dots, x_m \in \{0,1\}$ un assignement quelconque des variables. Soit i une clause. On vérifie tout d'abord que

$$\begin{aligned}(C^F x)_i \leqslant (b^F)_i &\Leftrightarrow \sum_{j=1}^{m} C^F_{i,j} x_j \leqslant -1 + \sum_{j=1}^{m} \max(0, C^F_{i,j}) \\ &\Leftrightarrow \sum_{j=1}^{m} \left(C^F_{i,j} x_j - \max(0, C^F_{i,j})\right) \leqslant -1.\end{aligned}$$

Or on observe que
- si $C_{i,j} > 0$ alors $C^F_{i,j} x_j - \max(0, C^F_{i,j}) = C^F_{i,j}(x_j - 1) \leqslant 0$,
- si $C_{i,j} = 0$ alors $C^F_{i,j} x_j - \max(0, C^F_{i,j}) = 0$,
- si $C_{i,j} < 0$ alors $C^F_{i,j} x_j - \max(0, C^F_{i,j}) = C^F_{i,j} x_j \leqslant 0$.

On en déduit donc que pour que la somme soit $\leqslant -1$, il faut et il suffit qu'au moins une entrée soit négative :

$$\begin{aligned}(C^F x)_i \leqslant (b^F)_i &\Leftrightarrow \exists j,\ \left(C^F_{i,j} x_j - \max(0, C^F_{i,j})\right) \leqslant -1 \\ &\Leftrightarrow \exists j, \begin{cases} C_{i,j} > 0 \text{ et } x_j = 0, \text{ ou} \\ C_{i,j} < 0 \text{ et } x_j = 1. \end{cases} \\ &\Leftrightarrow \exists j, \begin{cases} x_j \text{ apparaît avec négation dans } C_i \text{ et } x_j = 0, \text{ ou} \\ x_j \text{ apparaît sans négation dans } C_i \text{ et } x_j = 1. \end{cases} \\ &\Leftrightarrow C_i \text{ est vraie pour l'assignement } x.\end{aligned}$$

On en déduit donc immédiatement que x satisfait F si et seulement si x satisfait toutes les clauses C_i, si et seulement si $(C^F x)_i \leqslant (b^F)_i$ pout tout i, si et seulement si $C^F x \leqslant b^F$. □

On considère la variante suivante du problème PROG–LIN–ENTIER qui sera utile pour la suite.

PROG–LIN–ENTIER–CONJONCTION :
- **Donnée**: deux matrices $C \in \mathbb{Z}^{n \times m}$ et $C' \in \mathbb{Z}^{n' \times m}$, et deux vecteurs $b \in \mathbb{Z}^n$ et $b' \in \mathbb{Z}^{n'}$
- **Réponse**: existe-t-il $x \in \mathbb{Z}^m$ tel que $Cx \leqslant b$ et $C'x \leqslant b'$?

Question 3.3. *Montrer que PROG–LIN–ENTIER–CONJONCTION se réduit en temps polynomial au problème PROG–LIN–ENTIER.*

Solution : Soit (C, C', b, b') une instance de PROG–LIN–ENTIER–CONJONCTION. On construit, en temps polynomial, l'instance (D, e) où

$$D = \begin{pmatrix} C \\ C' \end{pmatrix}, \qquad e = \begin{pmatrix} b \\ b' \end{pmatrix}.$$

On voit facilement que pour tout $x \in \mathbb{Z}^m$,

$$Dx \leqslant e \;\Leftrightarrow\; \begin{pmatrix} Cx \\ C'x \end{pmatrix} \leqslant \begin{pmatrix} b \\ b' \end{pmatrix} \;\Leftrightarrow\; Cx \leqslant b \wedge C'x \leqslant b'.$$

On a donc bien réduit PROG–LIN–ENTIER–CONJONCTION à PROG–LIN–ENTIER en temps polynomial. □

Question 3.4. *Montrer qu'il existe une matrice $D \in \mathbb{Z}^{2m \times m}$ et un vecteur $d \in \mathbb{Z}^{2m}$, tels que pour tout $x \in \mathbb{Z}^m$, $Dx \leqslant d$ si et seulement si $x \in \{0,1\}^m$.*

Solution : Cela se fait facilement en posant la matrice et le vecteur

$$D = \begin{pmatrix} I_n \\ -I_n \end{pmatrix}, \qquad d = \begin{pmatrix} 1_n \\ 0_n \end{pmatrix},$$

où I_n est la matrice identité, 1_n le vecteur de 1 et 0_n le vecteur de 0. On vérifie facilement que

$$Dx \leqslant d \quad \Leftrightarrow \quad x \leqslant 1_m \wedge -x \leqslant 0 \quad \Leftrightarrow \quad x \in \{0,1\}^m.$$

□

Question 3.5. *Montrer que PROG–LIN–ENTIER est* NP*-complet.*

Solution : On a admis que le problème est dans NP, il suffit de montrer qu'il est NP-difficile. On fait une réduction à partir de 3-SAT, qui est NP-difficile. Soit F une instance. On construit, en temps polynomial, la matrice C^F et le vecteur b^F. On a alors que F est satisfiable et seulement s'il existe $x \in \{0,1\}^m$ tel que $C^F x \leqslant b^F$ d'après la question 3.2. C'est **presque mais tout à fait** ce que l'on veut : la quantification de x dans PROG–LIN–ENTIER est sur $x \in \mathbb{Z}^m$. Il faut donc ajouter la contrainte $x \in \{0,1\}^m$ grâve à la question 3.4 qui nous donne D et d tel que $x \in \{0,1\}^m$ si et seulement $Dx \leqslant d$. De plus, D et d sont clairement calculables en temps polynomial. On a alors

$$C^F x \leqslant b^F \wedge Dx \leqslant d \quad \Leftrightarrow \quad C^F x \leqslant b^F \wedge x \in \{0,1\}^m \quad \Leftrightarrow \quad F \text{ est satisfiable.}$$

On a donc réduit 3-SAT à PROG–LIN–ENTIER–CONJONCTION en temps polynomial. Or PROG–LIN–ENTIER–CONJONCTION se réduit en temps polynomial à PROG–LIN–ENTIER d'après la question 3.3 ce qui montre bien que ce derniers est NP-complet. □

Une variante importante du problème PROG–LIN–ENTIER est celui de l'existence d'une solution à un système linéaire en nombre entier. Pour $E = \mathbb{N}$ ou $\mathbb{Z}$, définit le problème suivant, dont on admettra qu'il est dans NP :

E–SYS-LINEAIRE :
- **Donnée**: une matrice $B \in \mathbb{Z}^{n\times m}$ et un vecteur $z \in \mathbb{Z}^n$
- **Réponse**: existe-t-il $x \in E^m$ tel que $Bx = z$?

Question 3.6. *Montrer que* $\mathbb{N}$–*SYS-LINEAIRE est* NP*-complet.*

Solution : On a admis que ce problème est dans NP. Il reste donc à montrer qu'il est NP-difficile. Faisons une réduction depuis PROG–LIN–ENTIER : soit (C, b) une instance. On cherche à savoir s'il existe $x \in \mathbb{Z}^m$ tel que $Cx \leqslant b$, c'est à dire si pour tout i,

$$C_{i,1}x_1 + \cdots + C_{i,m}x_m \leqslant b_i. \tag{1}$$

Notons que $x_j \in \mathbb{Z}$ mais on peut introduire deux variables $x_j^+, x_j^- \in \mathbb{N}$ et écrire $x_i = x_i^+ - x_i^-$. On a alors

$$C_{i,1}x_1^+ - C_{i,1}x_1^- + \cdots + C_{i,m}x_m^+ - C_{i,m}x_m^- \leqslant b_i.$$

De plus, il existe $y_i \in \mathbb{N}$ tel que

$$C_{i,1}x_1^+ - C_{i,1}x_1^- + \cdots + C_{i,m}x_m^+ - C_{i,m}x_m^- + y_i = b_i. \tag{2}$$

On vérifie sans difficulté qu'il existe $x \in \mathbb{Z}^m$ tel que (1) est vrai si et seulement il existe $x^\pm \in \mathbb{Z}^{2m}$ et $y \in \mathbb{Z}^m$ tels que (2) est vrai. On peut récrire (2) sous la forme d'un système linéaire

$$B\begin{pmatrix} x^\pm \\ y \end{pmatrix} = b$$

où $B \in \mathbb{Z}^{n\times 3m}$ est facilement calculable en temps polynomial à partir de C. On a donc montré que (B, b) est une instance positive de $\mathbb{N}$–SYS-LINEAIRE si et seulement si (C, b) est une instance positive de PROG–LIN–ENTIER, ce qui est clos la preuve puisque la réduction est en temps polynomial et que PROG–LIN–ENTIER est NP-difficile. □

4 Problèmes de (semi-)groupes de matrices

Dans la suite, on se donne un ensemble fini $A \subseteq R^{n\times n}$ de matrices, à coefficients dans un anneau R qui sera soit $\mathbb{Z}$ soit $\mathbb{Q}$. On définit le *semi-groupe* généré par A, noté $[\![A]\!]$, comme l'ensemble des produits finis de matrices de A. Formellement,

$$[\![A]\!] = \{X_1 \cdots X_m : m \in \mathbb{N}, X_1, \ldots, X_m \in A\}.$$

Si toutes les matrices de A sont inversibles, on définit aussi le *groupe* généré par A, noté $(\!|A|\!)$, qui est l'ensemble des produits finis de matrices de A ainsi que leurs inverses. Formellement,

$$(\!|A|\!) = [\![A \cup \overline{A}]\!], \qquad \overline{A} = \{M^{-1} : M \in A\}.$$

On **prendra garde** que $[\![A]\!] \neq (\!|A|\!)$ en général. On s'intéressera dans la suite aux problèmes suivants :

(R, n)–SEMIGROUPE–MORTALITÉ :
- **Donnée**: ensemble fini $A \subseteq R^{n\times n}$
- **Réponse**: $[\![A]\!]$ contient-il 0_n ?

(R, n)–SEMIGROUPE–APPARTENANCE :
- **Donnée**: ensemble fini $A \subseteq R^{n\times n}$ et $T \in R^{n\times n}$
- **Réponse**: $[\![A]\!]$ contient-il T ?

(R, n)–GROUPE–IDENTITÉ :
- **Donnée**: ensemble fini $A \subseteq R^{n\times n}$
- **Réponse**: $(\!|A|\!)$ contient-il I_n ?

(R, n)–GROUPE–APPARTENANCE :
- **Donnée**: ensemble fini $A \subseteq R^{n\times n}$ et $T \in R^{n\times n}$
- **Réponse**: $(\!|A|\!)$ contient-il T ?

Question 4.1. *Montrer que* $[\![A]\!] \neq (\!|A|\!)$ *pour* $A = \left\{ \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix} \right\}$.

Solution : Posons $M = \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix}$. Notons tout d'abord que $[\![A]\!] = \{M^k : k \in \mathbb{N}\}$ et $(\!|A|\!) = \{M^k : k \in \mathbb{Z}\}$. On vérifie que $M^{-1} = \begin{pmatrix} 1 & -1 \\ 0 & 1 \end{pmatrix}$, puis par induction que pour $k \in \mathbb{Z}$, $M^k = \begin{pmatrix} 1 & k \\ 0 & 1 \end{pmatrix}$. Il s'en suit que M^{-1} appartient à $(\!|A|\!)$ mais pas à $[\![A]\!]$. □

Question 4.2. *Montrer que pour tout* $R \in \{\mathbb{Z}, \mathbb{Q}\}$ *et* n, (R,n)*–*`SEMIGROUPE–MORTALITÉ` *se réduit en temps polynomial au problème* (R,n)*–*`SEMIGROUPE–APPARTENANCE`*. Montrer que* (R,n)*–*`GROUPE–APPARTENANCE` *se réduit en temps polynomial à* (R,n)*–*`SEMIGROUPE–APPARTENANCE`*.*

Solution : Soit A une instance de (R,n)–`SEMIGROUPE–MORTALITÉ`. On construit en temps polynomial l'instance $(A, 0_n)$ de (R,n)–`SEMIGROUPE–APPARTENANCE`. On a alors que $(A, 0_n)$ est une instance positive si et seulement si $0_n \in [\![A]\!]$, si et seulement si, A est une instance positive. On a donc bien réduit, en temps polynomial, le problème (R,n)–`SEMIGROUPE–MORTALITÉ` au problème (R,n)–`SEMIGROUPE–APPARTENANCE`.

Prenons maintenant une instance (A,T) de (R,n)–`GROUPE–APPARTENANCE`. On peut construit en temps polynomial l'ensemble $B = A \cup \overline{A}$ car l'inversion de matrices sur $\mathbb{Z}$ ou $\mathbb{Q}$ se fait en temps polynomial. Mais alors, (A,T) est une instance positive si et seulement si $T \in (\!|A|\!)$, si et seulement si $T \in [\![A \cup \overline{A}]\!]$, si et seulement si $T \in [\![B]\!]$, si et seulement si (B,T) est une instance positive de (R,n)–`SEMIGROUPE–APPARTENANCE`. □

Question 4.3. *Soit* L *et* L' *deux langages. Montrer que si* L *se réduit en temps polynomial vers* L' *et si* $L' \in$ NP *alors* $L \in$ NP*.*

Solution : Soit f une réduction en temps polynomial de L vers L'. Puisque L' est dans NP, il existe un vérificateur en temps polynomial pour L. Plus précisemment, il existe une machine M' fonctionnant en temps polynomial et un polynôme p tels que pour tout mot w',

$$w' \in L' \Leftrightarrow \exists u,\ |u| \leqslant p(|w'|) \wedge M'(w',u) \text{ accepte.}$$

On construit maintenant la machine M qui sur l'entrée (w,u) calcule $w' = f(w)$ puis lance M' sur l'entrée (w',u). Clairement M fonctionne en temps polynomial puisque f et M sont en temps polynomial. On vérifie maintenant que M est un vérificateur pour L. Pour cela, posons g le polynôme qui borne le temps d'exécution de f.

- Si $w \in L$ alors $w' = f(w) \in L'$ par définition d'une réduction, donc il existe u de longueur au plus $p(|w'|)$ tel que $M'(w',u)$ accepte. Ainsi, M accepte (w,u). De plus, $|w'| \leqslant g(|w|)$ donc $|u| \leqslant p(g(|w|))$ qui est un polynôme en $|w|$.
- Si $w \notin L$ alors $w' = f(w) \notin L'$ par définition. Ainsi, pour tout mot u, M' rejette (w',u) et donc M rejette (w,u).

□

5 Semi-groupes : le cas de la dimension 1

On s'intéresse tout d'abord au cas où $n = 1$, et on identifie les matrices de taille 1×1 avec R.

On rapelle que pour qu'un problème soit dans NP, il doit admettre un vérificateur polynomial et des certificats de taille **polynomial en la taille de l'entrée**.

Question 5.1. *Pourquoi est-ce qu'il n'est pas évident a priori que les problèmes définis dans la partie 4 sont dans* NP*, même pour* $R = \mathbb{Z}$ *et* $n = 1$ *?*

Solution : Étant donné un produit, on peut facilement vérifier en temps polynomial **en la longueur du produit** que ce produit donne la matrice souhaitée (I_n, 0_n ou T selon le problème). Toutefois, il n'est pas clair qu'il existe un tel produit de longueur polynomial **en la taille de l'instance**. Or, on se souvient que pour qu'un problème soit dans un NP, il faut qu'il admette un certificat de taille polynomial **en la taille de l'instance**. Ainsi, il faudrait soit un argument pour dire qu'il existe toujours un produit de longueur polynomial, ou bien trouver un autre type de certificat plus malin. □

Question 5.2. *Soit $A \subseteq \mathbb{Z}$ un ensemble fini et $t \in \mathbb{Z}$. Montrer que si $t \in [\![A]\!]$ alors il existe $a_1, \dots, a_m \in A$ tels que $t = a_1 \cdots a_m$ et $m \leqslant 1 + \log_2^+ |t|$.*

Solution : Supposons que $t = a_1 \cdots a_m$ pour un certain choix de $a_1, \dots, a_m \in A$. Si $t = 0$ alors c'est trivial car le produit est nul et si seulement $a_i = 0$ pour un certain i donc on peut toujours prendre $m = 1$.

Supposons que $t \neq 0$. Quitte à raccourcir le produit en supprimant des a_i, on peut supposer qu'il y a au plus un a_i qui appartient à $\{-1, 0, 1\}$. De plus, pour tous les autres a_i, on a que $|a_i| \geqslant 2$ puisque ce sont des entiers. Notons k le nombre de i tels que $a_i \notin \{-1, 0, 1\}$. On a donc que $k \geqslant m - 1$. Ainsi,

$$|t| = |a_1| \cdots |a_2| \cdots |a_m| \geqslant 2^k \geqslant 2^{m-1}.$$

On en déduit que $m - 3 \leqslant 1 + \log_2 |t|$ ce prouve le résultat. □

Question 5.3. *En déduire que tous les problèmes ci-dessus sont dans* NP *pour $R = \mathbb{Z}$ et $n = 1$.*

Solution : D'après la question 4.2 et la question 4.3, il suffit de montrer que $(\mathbb{Z}, 1)$–`SEMIGROUPE–APPARTENANCE` est dans NP. On va donc donner un vérificateur polynomial pour ce problème.

Un certificat pour une instance (A, T) est une liste $X_1, \dots, X_m$ de matrices dans A. Le vérificateur se contente de vérifier que les X_i appartiennent bien à A puis vérifie que $X_1 \cdots X_m = T$. Toutes ces opérations se font en temps polynomial en n et en m. De plus, d'après la question 5.2, si (A, t) est une instance positive, il existe toujours un tel produit avec $m \leqslant 1 + \log_2(1, |t|)$. En particulier, m est polynomial en la taille de l'entrée (qui est $\sum_{a \in A} \log_2^+ |a| + \log_2^+ |t|$). □

On suppose maintenant que $R = \mathbb{Q}$ (ce qui englobe le cas $R = \mathbb{Z}$ de toute façon). Fixons une instance (A, t) de $(\mathbb{Q}, 1)$–`SEMIGROUPE–APPARTENANCE` ou $(\mathbb{Q}, 1)$–`GROUPE–APPARTENANCE` . Sans perte de généralité, on peut supposer que $t \neq 0$ car le problème est trivial sinon. De plus, on peut supposer que $0 \notin A$ car 0 ne sera jamais utilisé dans la solution si $t \neq 0$.

Écrivons $A = \{a_1, \dots, a_k\}$ avec $a_j \neq 0$ et notons $p_1, \dots, p_\ell$ l'ensemble des facteurs premiers qui apparaissent dans les a_j et t. On décompose alors chaque a_j en facteurs premiers

$$a_j = p_1^{m_{1,j}} \cdots p_\ell^{m_{\ell,j}}$$

avec $m_{i,j} \in \mathbb{Z}$ (il peut y avoir des puissances négatives pour les rationnels). Par commutativité du produit dans $\mathbb{Q}$, on a donc que tout élément du semi-groupe $[\![A]\!]$ (resp. du groupe $(\!|A|\!)$) s'écrit sous la forme

$$a_1^{x_1} \cdots a_k^{x_k} = \prod_{i=1}^{\ell} p_i^{\sum_{j=1}^{k} x_j m_{i,j}} \tag{3}$$

avec $x_1, \dots, x_k \in \mathbb{N}$ (resp. $x_1, \dots, x_k \in \mathbb{Z}$). **Afin d'unifier les notations, on pose $E = \mathbb{N}$ si on regarde le problème d'appartenance à un semi-groupe et $E = \mathbb{Z}$ si on regarde un groupe.** Écrivons de même

$$t = p_1^{y_1} \cdots p_\ell^{y_\ell} \tag{4}$$

avec $y_1, \dots, y_\ell \in \mathbb{Z}$. On introduit la matrice $M_A = (m_{i,j})_{i,j} \in \mathbb{Z}^{\ell \times k}$, ainsi que le vecteur $y_A = (y_j)_j \in \mathbb{Z}^m$.

Question 5.4. *Montrer que (A, t) est une instance positive si et seulement s'il existe $x \in E^k$ tel que $M_A x = y_A$.*

Solution : Par unicité de la décomposition en facteurs premiers, d'après les équations (3) et (4), on a que $t \in [\![G]\!]$ (resp. $t \in (\!|A|\!)$) si et seulement s'il existe $x_1, \ldots, x_k \in E$ tels que

$$\bigwedge_{i=1}^{\ell} \left(\sum_{j=1}^{k} m_{i,j} x_j = y_i \right).$$

L'équation ci-dessus devient donc équivalente à

$$M_A x = y_A.$$

□

On pourra admettre dans la suite le résultat suivant : il existe un algorithme `NPremiers` qui sur l'entrée $m \in \mathbb{N}$ renvoie les m premiers nombres premiers en temps polynomial en m (la valeur de m, pas sa taille !).

Question 5.5. *Montrer qu'il existe un algorithme qui prend en entrée une matrice $B \in \mathbb{Z}^{\ell \times k}$ et un vecteur $z \in \mathbb{Z}^{\ell}$ et construit une instance (A, t), **en temps polynomial** en la taille de B et z, telle que*

- *$t \in [\![A]\!]$ si et seulement s'il existe $x \in \mathbb{N}^k$ tel que $Bx = z$,*
- *$t \in (\!|A|\!)$ si et seulement s'il existe $x \in \mathbb{Z}^k$ tel que $Bx = z$.*

Solution : D'après la question 5.4 on voit qu'il faut construire A et t afin que le $M_A = B$ et $y_A = z$. Pour cela, il suffit d'observer comme M_A est obtenu à partir de A : $(M_A)_{i,j} = m_{i,j}$ est la puissance de p_i dans a_j. Dans la construction, p_i était le i^{eme} nombre premier apparaissant dans les a_j et y.

On pose donc p_i le i^{eme} nombre premier, pour i allant de 1 à ℓ. D'après le résultat admis, on peut calculer $p_1, \ldots, p_\ell$ en temps polynomial en ℓ. En particulier, puisque l'algorithme doit écrire les nombre, chaque p_i est de taille polynomial en ℓ. On pose ensuite, pour j allant de 1 à k,

$$a_j = p_1^{B_{1,j}} \cdots p_\ell^{B_{\ell,j}}, \qquad t = p_1^{z_1} \cdots p_\ell^{z_\ell}.$$

Notons que a_j et t sont des nombres rationnels car les $B_{i,j}$ et y_j peuvent être négatifs. Par exponentiation rapide, on peut calculer les a_j et t en temps polynomial en la taille des p_j, $\log_2^+ |m_{i,j}|$ et $\log_2^+ |z_j|$. Or on a déjà expliqué que les p_j sont de taille polynomialle en m et donc le calcule des a_j et t se fait en temps polynomial en $m + \max_{i,j} \log_2^+ |B_{i,j}| + \max_j \log_2^+ |z_j|$, c'est à dire polynomial en la taille de l'instance (B, z). Par unicité de la décomposition en facteur premiers, on déduit que la décomposition des a_j et t de la forme (3) et (4) satisfait $M_A = B$ et $y_A = y$. Ainsi, par la question 5.4, $t \in [\![A]\!]$ (resp. $t \in (\!|A|\!)$) si et seulement s'il existe $x \in E^k$ tel que $M_A x = y_A$ si et seulement s'il existe $x \in E^k$ tel que $Bx = z$, où $E = \mathbb{N}$ (resp. $E = \mathbb{Z}$). □

Question 5.6. *Montrer que $(\mathbb{Q}, 1)$–`SEMIGROUPE–APPARTENANCE` est* NP*-complet.*

Solution : On a montré que ce problème est dans NP à la question 5.3. Il reste donc à montrer qu'il est NP-difficile. On réduit depuis $\mathbb{N}$–`SYS-LINEAIRE`. Soit (B, z) une instance de $\mathbb{N}$–`SYS-LINEAIRE`. D'après la question 5.5, il existe un algorithme qui construit en temps polynomial une instance (A, t) de $(\mathbb{Q}, 1)$–`SEMIGROUPE–APPARTENANCE` telle que (A, t) est positive si et seulement s'il existe $x \in \mathbb{N}^k$ (**attention :** c'est bien sur $\mathbb{N}$ car on parle de semi-groupe) tel que $Bx = z$, c'est à dire si et seulement si (B, z) est une instance positive. □

Question 5.7. *On admet*‡ *que* $\mathbb{Z}$*–SYS-LINEAIRE est décidable en temps polynomial. Pourquoi la preuve de la question 5.6 ne marche pas pour* $(\mathbb{Q}, 1)$*–GROUPE–APPARTENANCE ? Pourquoi ne peut-on pas déduire de la question 5.4 que* $(\mathbb{Q}, 1)$*–GROUPE–APPARTENANCE est décidable en temps polynomial ?*

Solution : La preuve de la question 5.6 ne marche pas pour $(\mathbb{Q}, 1)$–GROUPE–APPARTENANCE car la réduction de la question 5.5 nous obligerait à faire une réduction depuis $\mathbb{Z}$–SYS-LINEAIRE au lieu de $\mathbb{N}$–SYS-LINEAIRE. Or $\mathbb{Z}$–SYS-LINEAIRE n'est pas NP-difficile donc on ne peut pas conclure par cette approche.

A l'inverse, essayons de montrer que $(\mathbb{Q}, 1)$–GROUPE–APPARTENANCE est décidable en temps polynomial. Tentons d'écrire la réduction pour voir où est le problème. On se donne une instance de (A, t) de $(\mathbb{Q}, 1)$–GROUPE–APPARTENANCE, puis on calcule M_A et y_A et d'après la question 5.4, (M_A, y_A) est une instance positive de $\mathbb{Z}$–SYS-LINEAIRE si et seulement si (A, t) est une instance positive de $(\mathbb{Q}, 1)$–GROUPE–APPARTENANCE. Le problème se situe au niveau du calcule de M_A et y_A : il faut pouvoir calculer en temps polynomial la décomposition en facteur premier d'un nombre rationnel (ou un entier, c'est équivalent) quelconque. Or, on soupçonne très fortement que ce problème n'est ni dans P ni dans NP-complet, ce serait un problème dit intermédiaire. Les meilleurs algorithmes connus pour factoriser fonctionnent tous en temps super-polynomial (et sous-exponentiel). De plus, si on savait factoriser en temps polynomial, on pourrait casser le système de cryptographie à clef publique RSA. □

6 Semi-groupes : le cas de la dimension 3

On s'intéresse maintenant au cas des matrices de taille 3×3. Étant donné un ensemble fini $\mathcal{U}$ de paires de mots sur un alphabet Σ (c'est à dire que $\mathcal{U} \subseteq \Sigma^* \times \Sigma^*$), on pose $\mathcal{U}^*$ l'ensemble des paires

$$(u_1 u_2 \cdots u_m, v_1 v_2 \cdots v_m)$$

avec $m \in \mathbb{N}$ et $(u_1, v_1), \ldots, (u_m, v_m) \in \mathcal{U}$. On rappelle/admettra que le problème suivant, dit de la correspondance de Post, est indécidable, même pour un alphabet Σ de taille 2.

PCP :

— **Donnée**: Un ensemble fini $\mathcal{U}$ paires de mots sur un alphabet fini Σ

— **Réponse**: Décider si $\mathcal{U}$ admet une solution : existe-t-il $u \in \Sigma^*$ tel que $(u, u) \in \mathcal{U}^*$?

A titre d'exemple, sur l'alphabet $\Sigma = \{a, b\}$, l'ensemble $\mathcal{U} = \{(a, baa), (ab, aa), (bba, bb)\}$ admet une solution car pour $u = bbaabbba$, $m = 4$, $(u_1, v_1) = (bba, bb)$, $(u_2, v_2) = (ab, aa)$, $(u_3, v_3) = (bba, bb)$ et $(u_4, v_4) = (a, baa)$, on a

$$(u, u) = (bba \cdot ab \cdot bba \cdot a, bb \cdot aa \cdot bb \cdot baa) \in \mathcal{U}^*.$$

A l'inverse, l'ensemble $\mathcal{U} = \{(a, aa)\}$ ne peut pas admettre de solution puisque

$$\mathcal{U}^* = \{(a^n, a^{2n}) : n \geqslant 1\}$$

et il impossible d'avoir $a^n = a^{2n}$ pour $n > 0$.

On fixe $\Gamma = \{1, 2, 3\}$ dans le reste de cet exercice. Pour tout mot $w \in \Gamma^*$, on introduit l'encodage $\langle w \rangle \in \mathbb{Z}$ défini par

$$\langle w \rangle = \sum\nolimits_{i=1}^{|w|} w_i 4^{i-1}.$$

Intuitivement, $\langle w \rangle$ est l'entier dont l'écriture en base 4 est exactement le mot w.

‡. Pour les curieux, c'est une conséquence assez immédiate du fait de pouvoir calculer en temps polynomial la forme normale de Smith d'une matrice.

Question 6.1. *Que vaut* $\langle 1\rangle$ *?* $\langle 23\rangle$ *?* $\langle 123\rangle$ *?*

Solution : On calcule :

$$\begin{aligned}\langle 100\rangle &= 1\cdot 4^0 = 1,\\ \langle 11\rangle &= 2\cdot 4^0 + 3\cdot 4^1 = 14,\\ \langle 123\rangle &= 1\cdot 4^0 + 2\cdot 4^1 + 3\cdot 4^2 = 57.\end{aligned}$$

□

Question 6.2. *Montrer que pour tous* $u, v \in \Sigma^*$, $\langle uv\rangle = \langle u\rangle + 4^{|u|}\langle v\rangle$.

Solution : On calcule :

$$\begin{aligned}\langle uv\rangle &= \sum_{i=1}^{|uw|}(uv)_i 4^{i-1}\\ &= \sum_{i=1}^{|u|}(uv)_i 4^{i-1} + \sum_{i=|u|+1}^{|u|+|v|}(uv)_i 4^{i-1}\\ &= \sum_{i=1}^{|u|} u_i 4^{|v|-i} + \sum_{i=1}^{|v|} v_i 4^{|u|+i-1}\\ &= \langle u\rangle + 4^{|u|}\langle v\rangle.\end{aligned}$$

On vérifie sur l'exemple précédent :

$$\langle 1\rangle + 4^{|1|}\langle 23\rangle = 1 + 4\cdot 14 = 57 = \langle 10011\rangle.$$

□

Question 6.3. *Montrer que* $\langle\cdot\rangle$ *est injective sur* Γ^*.

Solution : On note bien que $0 \notin \Gamma$, sinon on aurait un problème car

$$\langle 0\rangle = 0 = \langle 00\rangle.$$

On va montrer par récurrence (forte) sur $\max(|u|,|v|)$ que pour tout $u, v \in \Gamma^*$, si $\langle u\rangle = \langle v\rangle$ alors $u = v$. Si $0 = \max(|u|,|v|)$ alors $u = v = \varepsilon$ donc c'est trivial. Sinon supposons (sans perte de généralité) que $|u| \geqslant 1$. Alors $\langle u\rangle = \sum_{i=0}^{|u|} u_i \cdot 4^{i-1} \geqslant u_1 \geqslant 1$ car $u_1 \in \Gamma = \{1,2,3\}$. Mais comme par ailleurs $\langle\varepsilon\rangle = 0$, on ne peut pas avoir $v = \varepsilon$ donc $|v| \geqslant 1$. Observons par ailleurs que $\langle u\rangle = \langle v\rangle$ donc $\langle u\rangle = \langle v\rangle \bmod 4$, c'est à dire que $u_1 = v_1$. On pose alors $u' = u_2\cdots u_{|u|}$ et $v' = v_2\cdots v_{|v|}$ et par la question 6.2,

$$\langle u'\rangle = \frac{\langle u\rangle - u_1}{4} = \frac{\langle v\rangle - v_1}{4} = \langle v'\rangle.$$

Par ailleurs, $\max(|u'|,|v'|) = \max(|u|,|v|) - 1$ donc, par récurrence, $u' = v'$ et donc $u = v$. □

Pour toute paire de mots $u, v \in \Sigma^*$, on pose la matrice

$$M(u,v) = \begin{pmatrix} 4^{|u|} & 0 & \langle u\rangle\\ 0 & 4^{|v|} & \langle v\rangle\\ 0 & 0 & 1\end{pmatrix}.$$

Question 6.4. *Montrer que pour tous* $u, v \in \Sigma^*$, $M(u,v)$ *est inversible.*

Solution : La matrice est triangulaire supérieure donc son déterminant est le produit des éléments sur la diagonale, c'est à dire $4^{|u|+|v|}$ qui est toujours non nul. □

Question 6.5. *Montrer que pour tous $u, v, u', v' \in \Sigma^*$, $M(u,v)M(u',v') = M(uu', vv')$.*

Solution : On calcule :

$$\begin{aligned}
M(u,v)M(u',v') &= \begin{pmatrix} 4^{|u|} & 0 & \langle u\rangle \\ 0 & 4^{|v|} & \langle v\rangle \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 4^{|u'|} & 0 & \langle u'\rangle \\ 0 & 4^{|v'|} & \langle v'\rangle \\ 0 & 0 & 1 \end{pmatrix} \\
&= \begin{pmatrix} 4^{|u|+|u'|} & 0 & 4^{|u|}\langle u'\rangle + \langle u\rangle \\ 0 & 4^{|v|+|v'|} & 4^{|v|}\langle v'\rangle + \langle v\rangle \\ 0 & 0 & 1 \end{pmatrix} \\
&= \begin{pmatrix} 4^{|u|+|u'|} & 0 & \langle uu'\rangle \\ 0 & 4^{|v|+|v'|} & \langle vv'\rangle \\ 0 & 0 & 1 \end{pmatrix} \qquad \text{d'après la question 6.2} \\
&= M(uu', vv').
\end{aligned}$$

□

Soit $\mathcal{U}$ un ensemble fini de paires de mots de Γ^*. On pose $\mathcal{M}(\mathcal{U}) = [\![\{M(u,v) : (u,v) \in \mathcal{U}\}]\!]$ le semi-groupe généré par les matrices qui encodent les paires de $\mathcal{U}$.

Question 6.6. *Montrer que $\mathcal{M}(\mathcal{U}) = \{M(u,v) : (u,v) \in \mathcal{U}^*\}$.*

Solution : Clair d'après la question 6.5 et la définition de $\mathcal{U}^*$. □

On introduit maintenant les matrices suivantes :

$$S = \begin{pmatrix} 1 & -1 & 0 \\ -1 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix}, \qquad T = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \end{pmatrix}.$$

Ces matrices ont été choisies pour satisfaire les équations suivantes, pour tout $u, v \in \Gamma^*$,

$$S \cdot M(u,v) \cdot T = (\langle u1\rangle - \langle v\rangle)ST, \qquad S \cdot M(u,v) \cdot S = (4^{|u|} + 4^{|v|})S, \tag{5}$$

$$T \cdot M(u,v) \cdot T = \langle u1\rangle T, \qquad T \cdot M(u,v) \cdot S = 4^{|u|}TS. \tag{6}$$

On ne demande pas de vérifier ces équations.

On pose $\mathcal{S}(\mathcal{U}) = [\![\{S, T\} \cup \{M(u,v) : (u,v) \in \mathcal{U}\}]\!]$ le semi-groupe généré par les mêmes matrices que $\mathcal{M}(\mathcal{U})$, ainsi que S et T.

Question 6.7. *Montrer que tout élément de $\mathcal{S}(\mathcal{U})$ peut s'écrire sous la forme*

$$\alpha_1 \cdots \alpha_k XAY \tag{7}$$

avec $A \in [\![S, T]\!] \cup \{I_3\}$, $X, Y \in \mathcal{M}(\mathcal{U}) \cup \{I_3\}$ et pour tout $i = 1, \ldots, k$,

$$\alpha_i \in \{1, \langle x_i 1\rangle - \langle y_i\rangle, 4^{|x_i|} + 4^{|y_i|}, 4^{|x_i|}, \langle x_i 1\rangle\}$$

pour une certaine paire $(x_i, y_i) \in \mathcal{U}^*$.

Solution : On peut procéder par récurrence sur la longueur du produit. Prenons donc un produit Z dans $\mathcal{S}(\mathcal{U})$. On commence par regrouper ensemble sous-produits de $M(u,v)$ que l'on peut voir comme des éléments de $\mathcal{M}(\mathcal{U})$. Supposons que le produit contient un sous-produit d'une des forme

$$SXT, \qquad SXS, \qquad TXS, \quad \text{ou} \quad TXT$$

avec $X \in \mathcal{M}(\mathcal{U})$. D'après la question 6.6, $X = M(x,y)$ pour $(x,y) \in \mathcal{U}^*$. On peut simplifier ce sous-produit grâce à (5) qui devient

$$\alpha ST, \qquad \alpha SS, \qquad \alpha TS, \quad \text{ou} \quad \alpha TT$$

et $\alpha \in \{\langle x_i 1\rangle - \langle y_i\rangle, 4^{|x_i|} + 4^{|y_i|}, 4^{|x_i|}, \langle x_i 1\rangle\}$. En déplaçant le α (qui est un scalaire), on a donc $Z = \alpha \cdot Z'$ avec Z' un produit strictement plus court que Z (puisqu'on a supprimé un X). On applique alors la récurrence à Z' et on conclue.

Supposons maintenant que le produit ne contienne aucun des sous-produit listés plus haut. Alors nécessairement Y est de la forme

$$XAY$$

avec X et Y des produits (potentiellement vides) de $M(u,v)$ et A un produit (potentiellement vide) de S et T. On a donc $A \in [\![S,T]\!] \cup \{I_3\}$, $X, Y \in \mathcal{M}(\mathcal{U}) \cup \{I_3\}$ et on peut prendre $\alpha_1 = 1$ pour avoir la bonne forme. □

On admettra, car il s'agit d'un simple calcul, qu'on a les relations suivantes :

$$S^2 = 2S, \qquad T^2 = T, \qquad STS = S, \qquad TST = T. \tag{8}$$

Question 6.8. *Montrer que* $0 \notin [\![S,T]\!]$.

Solution : Prenons un élément de $[\![S,T]\!]$: d'après (8) on voit que tout produit de longueur 3 de S et T se simplifie :

$$\begin{array}{llll} SSS = 4S, & SST = 2ST, & STT = ST, & STS = S, \\ TSS = 2TS, & TST = T, & TTT = T, & TTS = TS. \end{array}$$

Par ailleurs, on a aussi les simplifications

$$SS = 2S, \qquad\qquad TT = T.$$

On en déduit que tout produit de $[\![S,T]\!]$ peut se simplifier jusqu'à être de l'une des formes suivante :

$$2^p S, \qquad 2^p T, \qquad 2^p ST, \qquad 2^p TS$$

avec $p \in \mathbb{N}$. Or on vérifie que les produits ST et TS sont non-nuls, ce qui conclue. □

Question 6.9. *Montrer que le produit* (7) *est nul alors nécessairement l'un des* α_i *est nul.*

Solution : On vérifie que $M(u,v)$ est inversible pour tout $(u,v) \in \Gamma^*$. Ainsi le semi-groupe $\mathcal{M}(\mathcal{U})$ ne contient que des matrices inversibles. Par ailleurs, I_3 est aussi inversible donc $XAY = 0$ si et seulement si $A = 0$. Or soit $A = I_3$, soit $A \in [\![S,T]\!]$ et donc ne peut pas être nul d'après la question 6.8. On en déduit que $XAY \neq 0$ et que la seule possibilité est que le produit $\alpha_1 \cdots \alpha_k$ est nul, c'est à dire que l'un des α_i est nul. □

Question 6.10. *Montrer que si* $0 \in \mathcal{S}(\mathcal{U})$ *alors existe* $(x,y) \in \mathcal{U}^*$ *tels que* $x1 = y$.

Solution : Supposons que $0 \in \mathcal{S}(\mathcal{U})$. Alors d'après les questions 7 et 6.9, l'un des α_i est nul. Or si $(x, y) \in \mathcal{U}^*$, on vérifie facilement que $4^{|x|}$, $4^{|x|} + 4^{|y|}$ et $\langle x1 \rangle$ ne peuvent pas être nuls (car ils sont strictement positifs). La seule possibilité est donc que $\langle x1 \rangle = \langle y \rangle$. Or $x, y \in \Gamma^*$ et $1 \in \Gamma$ donc $x1 \in \Gamma^*$. D'après la question 6.3, $\langle \cdot \rangle$ est injective sur Γ^* donc nécessairement $x1 = y$. □

Question 6.11. *Montrer que s'il existe $(x, y) \in \mathcal{U}^*$ tels que $x1 = y$ alors $0 \in \mathcal{S}(\mathcal{U})$.*

Solution : Supposons qu'il existe $(x, y) \in \mathcal{U}^*$ tel que $x1 = y$. Alors

$$SM(x, y)T = (\langle x1 \rangle - \langle y \rangle)ST = 0$$

d'après (5). Par ailleurs, $M(x, y) \in \mathcal{M}(\mathcal{U})$ d'après la question 6.6 donc $SM(x, y)T \in \mathcal{S}(\mathcal{U})$, ce qui conclue. □

On rappelle que $\Gamma = \{1, 2, 3\}$ et on pose $\Sigma = \{2, 3\}$. Soit $\mathcal{U}$ un ensemble fini de paires de mots **sur Σ**. On pose

$$\tilde{\mathcal{U}} = \mathcal{U} \cup \{(u, v1) : (u, v) \in \mathcal{U}\}$$

qui est un ensemble fini de paires de mots **sur Γ**.

Question 6.12. *Montrer qu'il existe $w \in \Sigma^*$ tel que $(w, w) \in \mathcal{U}^*$ si et seulement s'il existe $x \in \Gamma^*$ tel que $(x, x1) \in \tilde{\mathcal{U}}^*$.*

Solution : Si on a $w \in \Sigma^*$ tel que $(w, w) \in \mathcal{U}^*$ alors il existe $m \in \mathbb{N}$ et $(u_1, v_1), \ldots, (u_m, v_m) \in \mathcal{U}$ tels que

$$u_1 u_2 \cdots u_m = w = v_1 v_2 \cdots v_m.$$

Mais alors pour $x = w$, on a

$$x = u_1 u_2 \cdots u_m, \qquad x1 = v_1 v_2 \cdots v_m 1$$

avec $(u_1, v_1), \ldots, (u_{m-1}, v_{m-1}) \in \mathcal{U} \subseteq \tilde{\mathcal{U}}$ et $(u_n, v_m 1) \in \tilde{\mathcal{U}}$ par définition.

Inversement, si on a $x \in \Gamma^*$ tel que $(x, x) \in \mathcal{U}^*$ alors il existe $m \in \mathbb{N}$ et $(u_1, v_1), \ldots, (u_m, v_m) \in \tilde{\mathcal{U}}$ tels que

$$x = u_1 u_2 \cdots u_m, \qquad x1 = v_1 v_2 \cdots v_m.$$

Mais notons que si $(u, v) \in \tilde{\mathcal{U}}$ alors soit $(u, v) \in \mathcal{U}$ soit $(u, v') \in \mathcal{U}$ avec $v = v'1$. Dans tous les cas, on a $u \in \Sigma^*$ et donc $x \in \Sigma^*$. Or $1 \notin \Sigma^*$ donc x ne contient pas de 1. Mais alors, $x1$ ne contient qu'un unique 1 à la fin, ce qui veut dire que $v_1, \ldots, v_{m-1} \in \Sigma^*$ et $v_m = v'_m 1$ avec $v'_m \in \Sigma^*$. Ceci implique donc que $(u_1, v_1), \ldots, (u_{m-1}, v_{m-1}) \in \mathcal{U}$ et $(u_m, v'_m) \in \mathcal{U}$. On a alors bien que

$$u_1 u_2 \cdots u_m = x = v_1 v_2 \cdots v_{m-1} v'_m$$

ce qui prouve que $(x, x) \in \mathcal{U}^*$. □

Question 6.13. *Montrer que $(\mathbb{Z}, 3)$–`SEMIGROUPE–MORTALITÉ` et $(\mathbb{Z}, 3)$–`SEMIGROUPE–APPARTENANCE` sont indécidables.*

Solution : On va montrer que $(\mathbb{Z}, 3)$–`SEMIGROUPE–MORTALITÉ` est indécidable. Il s'en déduit alors que $(\mathbb{Z}, 3)$–`SEMIGROUPE–APPARTENANCE` est indécidable par une réduction immédiate.

On réduit depuis `PCP` avec Σ de taille 2, qui est indécidable. Quitte à renommer les lettres, on peut supposer que $\Sigma = \{2, 3\}$. Soit $\mathcal{U}$ une instance. On construit (en temps polynomial, ou au moins en temps fini) l'ensemble de mots $\tilde{\mathcal{U}}$ sur Γ puis l'ensemble fini de matrices

$$A = \{S, T\} \cup \{M(u, v) : (u, v) \in \tilde{\mathcal{U}}\} \subseteq \mathbb{Z}^{3\times 3}.$$

On a $\mathcal{S}(\tilde{\mathcal{U}}) = [\![A]\!]$ par définition. On va montrer que $\mathcal{U}$ admet une solution si et seulement si $0 \in [\![A]\!]$, c'est à dire si et seulement si A est une instance positive de $(\mathbb{Z}, 3)$–SEMIGROUPE–MORTALITÉ. On a

$$\begin{aligned} 0 \in [\![A]\!] = \mathcal{S}(\tilde{\mathcal{U}}) &\Leftrightarrow \exists (x, y) \in \tilde{\mathcal{U}}^*, x1 = y && \text{par les questions 6.10 et 6.11} \\ &\Leftrightarrow \exists w \in \Sigma^*, (w, w) \in \tilde{\mathcal{U}} && \text{par la question 6.12} \\ &\Leftrightarrow \mathcal{U} \text{ admet une solution.} \end{aligned}$$

□