Banques MP et MPI inter-ENS – Session 2023
Rapport relatif à l'épreuve orale d'informatique

— Épreuve commune à toutes les ENS
— Coefficients (en pourcentage du total des points de chaque concours) :

École	Concours MP	Concours MP - Info	Filière MPI
Paris	23.1%	13.3%	13.3%
Lyon	10.8%	14.1%	14.1%
Paris-Saclay	23.1%	13.2%	13.2%
Rennes	23.1%	8.6%	13.9%

— Membres du jury :
  — Romain Demangeon
  — Stéphane Le Roux
  — Brice Minaud
  — Raphaël Monat
  — Charles Paperman
  — André Schrottenloher
  — Pierre Senellart

L'épreuve orale d'informatique fondamentale décrite dans ce rapport est commune à toutes les Écoles Normales Supérieures.

Cette année, le jury a interrogé :

— 132 candidat(e)s pour la filière MP. Les notes données s'échelonnent entre 5 et 20, avec une médiane à 12, une moyenne à 12.52 et un écart-type de 3.34. La figure 1 présente l'histogramme complet des notes.

— 85 candidat(e)s pour la filière MPI. Les notes données s'échelonnent entre 5 et 20, avec une médiane à 13, une moyenne à 13.19 et un écart-type de 3.27. La figure 2 présente l'histogramme complet des notes.

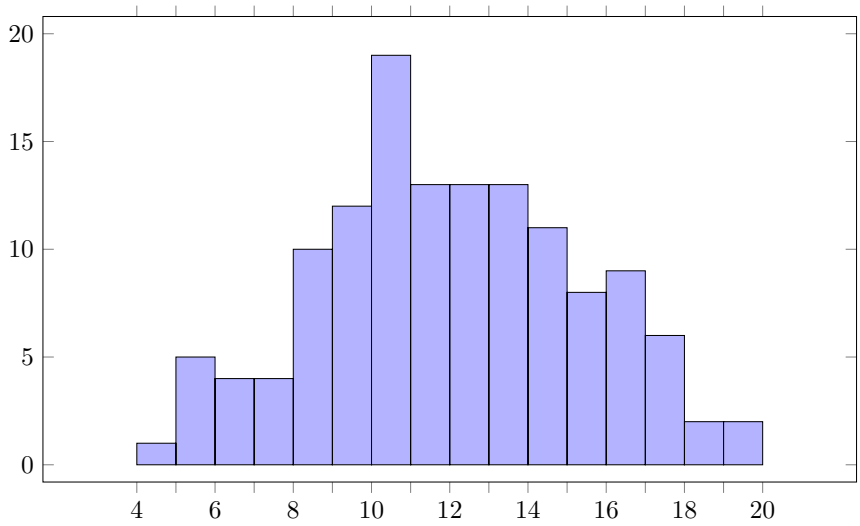


FIGURE 1 – Histogramme des notes de l'épreuve (filière MP). La colonne positionnée entre x et $x+1$ comptabilise les notes comprises entre x exclus et $x+1$ inclus.

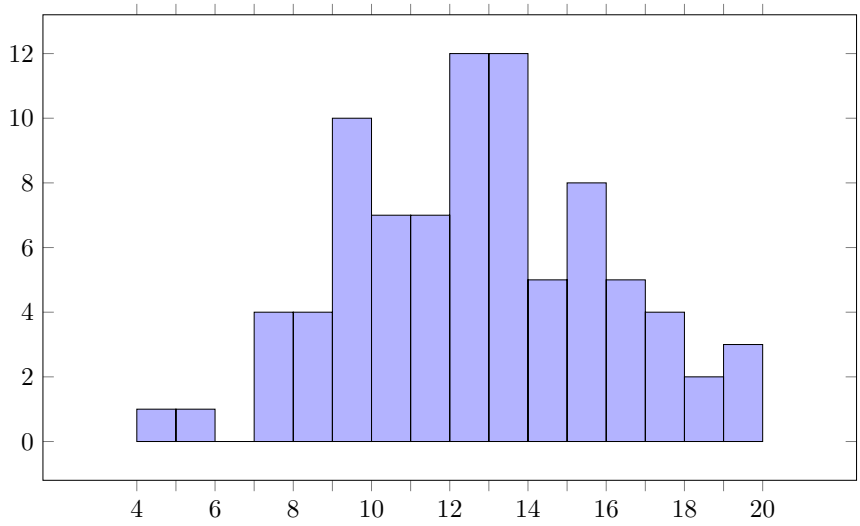


FIGURE 2 – Histogramme des notes de l'épreuve (filière MPI). La colonne positionnée entre x et $x+1$ comptabilise les notes comprises entre x exclu et $x+1$ inclus.

Après avoir reçu un sujet, les candidat(e)s disposent de 30 minutes de préparation, suivies de 28 minutes d'interrogation devant un des examinateurs. En effet, deux minutes sont utilisées par l'examinateur pour aller chercher la/le candidat(e) en salle de préparation. Nous rappelons également que jusqu'à quatre candidat(e)s peuvent passer l'épreuve à la suite sur le même sujet, auquel cas la première/le premier d'entre eux est invité(e) à patienter 30 minutes dans la salle de préparation à l'issue de son oral, afin de garantir la confidentialité du sujet.

Comme il a été rappelé en préambule des sujets distribués :

> Le but de cette épreuve est d'évaluer la progression des candidates et candidats dans les questions, mais aussi la qualité de leur exposé des solutions, ainsi que l'autonomie dont elles ou ils font preuve pendant l'oral.

Le jury a proposé 24 sujets originaux (16 pour le concours MP et 8 pour le concours MPI), dont la liste est donnée en annexe de ce document. En moyenne 9 à 10 candidat(e)s étaient interrogés sur chaque sujet (parfois jusqu'à 12 candidat(e)s, et au minimum 4), ce qui permet d'harmoniser les évaluations d'un sujet donné.

Chaque sujet débute par un énoncé qui présente un problème d'informatique et introduit ses notations, puis comporte des questions de difficulté globalement croissante. Les premières questions sont faciles d'accès et permettent de démontrer que l'on a compris l'énoncé, ou d'aider à sa compréhension. Les dernières questions d'un sujet sont souvent des questions d'ouverture plus difficiles. En général, il n'est pas attendu que les candidat(e)s traitent l'intégralité des questions.

Les sujets proposés portent sur des thèmes variés de science informatique : langages, graphes, logique, *etc.*, en lien avec les programmes respectifs des filières MP et MPI. Ils nécessitent de s'approprier des concepts nouveaux, démontrer des résultats théoriques et construire des solutions techniques telles que des algorithmes. Bien que l'épreuve mette l'accent sur les concepts théoriques de l'informatique, on veille à garder à l'esprit le sens des objets que l'on étudie, et un certain sens pratique (par exemple dans la conception d'algorithmes) est apprécié.

Le jury tient à féliciter les candidates et candidats qui ont pour la plupart démontré des acquis très solides, et fait preuve d'idées excellentes malgré les contraintes de temps inhérentes à l'épreuve. Même celles et ceux ayant obtenu de moins bonnes notes ont montré des qualités certaines. En particulier, la plupart des candidat(e)s affiche une bonne maîtrise des concepts mathématiques essentiels (induction, disjonction de cas) ainsi qu'une bonne initiative (par exemple le fait de tester un algorithme sur un petit exemple avant d'en déduire le cas général).

Le jury adresse les conseils suivants aux futur(e)s candidat(e)s.

Concours et autres voies d'accès Le jury invite toutes les personnes intéressées par l'enseignement et la recherche à se présenter au concours. Une page web [1] recense des informations supplémentaires sur les études en informatique dans les ENS, ainsi que les différentes voies d'accès pour les candidates et candidats de filières MPI et MP (concours *et* dossier).

Début de l'oral. Au début de l'oral, il est possible de décrire le sujet en une ou deux phrases, en particulier si cela donne le contexte et illustre le recul du candidat. Cependant, il ne s'agit pas de répéter tout l'énoncé et de perdre du temps : l'examinateur a lui aussi l'énoncé sous les yeux durant l'oral.

Le jury conseille aux candidat(e)s d'annoncer les questions qui ont été traitées intégralement ou presque, puis les questions auxquelles elles et ils ont réfléchi sans trouver de solution. Cela permet à l'examinateur de planifier l'oral afin de rentabiliser le temps de préparation.

Le raisonnement. Lorsqu'une question demande un raisonnement par induction, il est important de mentionner le cas de base même si celui-ci est trivial (a fortiori lorsqu'il ne l'est pas), ce qui a été omis par certain(e)s candidat(e)s. Le jury a par ailleurs constaté que le raisonnement par l'absurde était très apprécié des candidat(e)s, alors qu'il s'agissait parfois d'une simple preuve par contraposée.

Le jury a remarqué que certaines personnes étaient déstabilisées par des questions de combinatoire des mots : par exemple par l'étude de cas d'un mot $uv = st$ en fonction de si $|u| \leq |s|$ ou non.

1. `https://diplome.di.ens.fr/informatique-ens/`

Pédagogie, intuition, formalisme. Si la situation le permet, il est souhaitable de décrire (en général oralement ou/et par un dessin) une preuve dans les grandes lignes avant de se lancer dans la preuve formelle. Cela permet à l'examinateur de s'assurer que le ou la candidate a compris le principe de la preuve. De même, il est souhaitable de décrire un algorithme dans les grandes lignes avant de se lancer dans l'écriture du pseudo-code.

Si une question attend une réponse oui/non, il est conseillé d'annoncer cette réponse avant de se lancer dans la preuve. De même, si une question contient plusieurs résultats à prouver, annoncer lequel sera prouvé en premier, *etc.*

Certains sujets demandent d'absorber plusieurs notions nouvelles, parfois au moyen d'un formalisme assez lourd. On attend alors des candidat(e)s une capacité à s'affranchir du formalisme pour se concentrer sur la signification des objets. Cependant, si l'examinateur demande des précisions, le ou la candidate doit être en mesure de justifier formellement son raisonnement. Lorsque des questions demandent des représentations graphiques, il est important de traiter celles-ci : elles sont ici pour guider la personne candidate à établir une intuition, afin de faciliter le traitement des questions suivantes.

Gestion du tableau. Le jury est satisfait de l'organisation du tableau par les candidat(e)s. Il conseille aux candidat(e)s d'écrire assez grand et de prendre plus de place, notamment si cela peut permettre d'ajouter des détails plus tard tout en restant clair. Enfin, il rappelle qu'une brosse est à disposition des candidat(e)s pour effacer le tableau (l'effacement à la main rendant rapidement le tableau illisible pour l'examinateur).

Remarques diverses. Le jury rappelle que l'algorithme de Floyd-Warshall est au programme MPI, ainsi que le langage C. Il est donc intéressant de savoir comment implémenter en pratique l'algorithme, notamment pour représenter les poids $+\infty$ des matrices d'adjacence, ainsi que de garder en tête que les entiers machine ont une taille bornée et que des dépassements de capacité peuvent donc se produire.

Il s'est étonné du fait que certaines personnes confondaient lettres grecques et latines, par exemple w et ω (et même W et Ω), ou v et ν. Ce n'est pas pénalisant en soi, mais peut prêter à confusion pour l'examinateur.

Une question demandait de calculer les ensembles minimaux pour l'inclusion parmi les ensembles intersectant chaque ensemble d'une collection donnée. Bien moins de la moitié des candidat(e)s ont pu décrire un algorithme dans les grandes lignes. La capacité à le faire ne nous a pas semblé liée à l'agilité mathématique sur les autres questions. La solution était conceptuellement simple mais nécessitait peut-être un peu de recul.

A Annexe : sujets proposés pour la filière MP

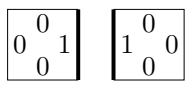
Certains sujets sont accompagnés d'*ébauches* de solution.

Pavages apériodiques du plan

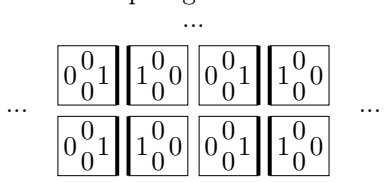
Définitions.

- **Couleurs.** On appelle *couleur* un entier positif ou nul.
- **Tuile.** Une *tuile* est une suite ordonnée de 4 couleurs. Intuitivement, cette suite décrira les couleurs des côtés gauche, haut, droit, et bas d'un carré de taille fixe. Pour faciliter les notations, on pourra écrire une tuile $t \in \mathbb{N}^4$ sous la forme $(t_{\mathsf{gauche}}, t_{\mathsf{haut}}, t_{\mathsf{droite}}, t_{\mathsf{bas}})$.
- **Jeu de tuiles.** Un *jeu de tuiles* est un ensemble fini non vide de tuiles.
- **Pavage du plan.** Étant donné un jeu de tuiles T, un *T-pavage du plan* est une application $f : \mathbb{Z}^2 \to T$. Intuitivement, cette application décrit un choix d'une tuile de T pour chaque case d'un damier infini couvrant le plan.
- **Pavage valide.** Un pavage est dit *valide* si deux tuiles adjacentes ont la même couleur le long du côté où elles se touchent. Formellement, un T-pavage $f : \mathbb{Z}^2 \to T$ est valide si et seulement si pour tout $i, j \in \mathbb{Z}$, $f(i,j)_{\mathsf{haut}} = f(i,j+1)_{\mathsf{bas}}$, et $f(i,j)_{\mathsf{droite}} = f(i+1,j)_{\mathsf{gauche}}$.
- **Jeu de tuiles pavable.** Un jeu de tuiles T est dit *pavable* s'il existe un T-pavage valide.

Exemple. Avec le jeu de tuiles $\{(0,0,1,0),(1,0,0,0)\}$, on peut créer un pavage valide comme suit.



Tuiles $(0,0,1,0)$ et $(1,0,0,0)$



Pavage valide

Question 1. Donner un exemple de jeu de tuiles qui n'est pas pavable, avec 3 tuiles ou plus, et 2 couleurs.

Question 2. On dit qu'un jeu de tuiles T est clos par rotation si $(a,b,c,d) \in T \implies (b,c,d,a) \in T$. Montrer que tout jeu de tuiles clos par rotation est pavable.
Indication. On pourra commencer par montrer que le jeu qui contient la tuile $(1,2,3,4)$ et ses rotations est pavable, puis déduire le cas général.

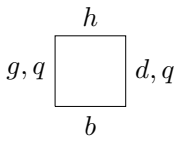
Pavage périodique. Un pavage $f : \mathbb{Z}^2 \to T$ *admet la période* $(a,b) \in (\mathbb{Z}^2) \setminus \{0,0\}$ si et seulement si $f(i+a, j+b) = f(i,j)$ pour tout $i, j \in \mathbb{Z}$. Un pavage est *périodique* s'il admet une période.
Pavage doublement périodique. Un pavage est *doublement périodique* s'il admet une période de la forme $(a,0)$, et une période de la forme $(0,b)$.

Question 3. Donner un exemple de jeu de tuiles qui admet des pavages valides périodiques et non périodiques.

Question 4. Montrer que si un jeu de tuiles admet un pavage valide de période $(0,b)$, alors il admet un pavage valide doublement périodique.

Question 5. Montrer qu'un un jeu de tuiles admet un pavage valide périodique si et seulement si il admet un pavage valide doublement périodique.

Dans la suite, on cherche à construire un jeu de tuiles dont les seuls pavages valides sont apériodiques. Pour cela, il est commode d'étendre l'ensemble des couleurs à $\mathbb{N} \cup (\mathbb{N} \times \mathbb{Q})$ On s'intéresse à des tuiles de la forme :



où $g, h, d, b \in \mathbb{N}$ et $q \in \mathbb{Q}$. On dit qu'une telle tuile est *multiplicative* si $h = q(b + d - g)$, et son *facteur* est q.

Représentation. Soit T un jeu de tuiles multiplicatives, et f un T-pavage. On dit que la i-ième ligne de f *représente* le nombre réel x si les sommes partielles $\frac{1}{2k+1}\sum_{j=-k}^{k} f(i,j)_{\mathsf{bas}}$ convergent vers x.

Question 6. Soit T est un jeu de tuiles multiplicatives, et f un T-pavage doublement périodique valide. Soit q_i le facteur de la tuile $f(i, 0)$. Montrer que si la i-ième ligne représente x, alors la $(i + 1)$-ième ligne représente qx. Qu'en est-il si le pavage n'est pas périodique ?

Jeu complet. Soit $a, b \in \mathbb{N}^*$. Un jeu de tuiles T multiplicatif de facteur q est *complet sur* $[a, b]$ si : pour toute suite $(u_i)_{i\in\mathbb{Z}}$ à valeurs dans $[a, b] \cap \mathbb{N}$, il existe un T-pavage f (pas nécessairement valide) tel que pour tout j : $f(0, j)_{\mathsf{bas}} = u_j$, et $f(0, j)_{\mathsf{droite}} = f(0, j + 1)_{\mathsf{gauche}}$.

Question 7. Construire un jeu de tuiles T_2 multiplicatif complet de facteur 2 sur $[1, 3]$.

Question 8. Montrer qu'il existe un jeu de tuiles $T_{1/3}$ multiplicatif complet de facteur $1/3$ sur $[3, 6]$.

Soit $T_{\mathsf{ap}} = T_2 \cup T_{1/3}$. Pour simplifier, on admet que T_2 et $T_{1/3}$ ont été créés avec la propriété de compatibilité suivante : pour tout $x \in [1, 3] \setminus \{2^x 3^y : x, y \in \mathbb{Z}\}$, il existe une T_{ap}-pavage valide dont la ligne 0 représente x. On suppose aussi que les couleurs en haut des tuiles de T_2 et $T_{1/3}$ sont strictement positives.

Question 9. Montrer que T_{ap} admet des pavages apériodiques valides. Que peut-on dire sur le nombre de tels pavages ?

Question 10. Montrer que T_{ap} n'admet pas de pavage périodique valide.

Treillis distributifs

Un ensemble E muni de deux lois de composition interne $\vee$ et $\wedge$ est un *treillis distributif* si les conditions suivantes sont satisfaites.

1. **Associativité.** $\forall a, b, c \in E$, $(a \vee b) \vee c = a \vee (b \vee c)$, et $(a \wedge b) \wedge c = a \wedge (b \wedge c)$.
2. **Commutativité.** $\forall a, b \in E$, $a \vee b = b \vee a$, et $a \wedge b = b \wedge a$.
3. **Distributivité.** $\forall a, b, c \in E$, $(a \wedge b) \vee c = (a \vee c) \wedge (b \vee c)$, et $(a \vee b) \wedge c = (a \wedge c) \vee (b \wedge c)$.
4. **Absorption.** $\forall a, b \in E$, $a \vee (a \wedge b) = a$, et $a \wedge (a \vee b) = a$.

Un treillis distributif $(E, \vee, \wedge)$ est dit *fini* si E est fini.

Exemples de treillis distributifs :

— $(\mathcal{P}(S), \cup, \cap)$, où S est un ensemble quelconque, et $\mathcal{P}(S)$ l'ensemble de ses sous-ensembles.

— $(\mathbb{Z}^d, \max_d, \min_d)$, pour $d \in \mathbb{N}^*$ et $\max_d : \mathbb{Z}^d \times \mathbb{Z}^d \to \mathbb{Z}^d$ défini par $\max_d((x_i)_{i=1}^d, (y_i)_{i=1}^d) \mapsto (\max(x_i, y_i))_{i=1}^d$, similairement pour $\min_d$.

On admet que ces exemples sont valides, on ne demande pas de les vérifier.

Question 1. Donner un exemple de treillis distributif à trois éléments.

Question 2. Étant donné un treillis distributif $(E, \vee, \wedge)$, on définit la relation $\leq$ sur E par : $a \leq b$ si et seulement si $a \wedge b = a$. Montrer que $\leq$ est une relation d'ordre partiel. Montrer que, de plus, toute paire (a, b) d'éléments de $(E, \leq)$ admet une borne supérieure (c'est-à-dire : l'ensemble des majorants de $\{a, b\}$ est non-vide et admet un minimum) et une borne inférieure (c'est-à-dire : l'ensemble des minorants est non-vide et admet un maximum).

Question 3. On dit qu'un ordre partiel sur E obtenu comme dans la question précédente est *issu* du treillis distributif $(E, \vee, \wedge)$. Donner un exemple d'ordre partiel qui ne peut pas être issu d'un treillis distributif.

Morphisme. Soit A, B deux treillis distributifs. Une application $\varphi : A \to B$ est un *morphisme* de treillis si pour tout $a, a' \in A$, $\varphi(a \wedge_A a') = \varphi(a) \wedge_B \varphi(a')$, et $\varphi(a \vee_A a') = \varphi(a) \vee_B \varphi(a')$.
Plongement. On dit que A se *plonge* dans B s'il existe un morphisme injectif de A dans B.
Dimension. La *dimension* $\dim(E)$ d'un treillis distributif E est le plus petit entier $d \in \mathbb{N}$, s'il existe, tel que E se plonge dans $(\mathbb{Z}^d, \max_d, \min_d)$. Si un tel d n'existe pas, on dit que la dimension est infinie.

Question 4. Soit S un ensemble fini. Montrer que la dimension de $(\mathcal{P}(S), \cup, \cap)$ est finie.

Dans la suite, on pose $(E, \vee, \wedge)$ un treillis distributif fini. On dit qu'un élément a est *irréductible* si $a = a_1 \vee a_2$ implique $a = a_1$ ou $a = a_2$.

Question 5. Soit $b \in E$, et soit $\{a_1, \ldots, a_k\}$ l'ensemble des éléments irréductibles inférieurs ou égaux à b pour l'ordre de la question 2. Montrer $a_1 \vee \cdots \vee a_k = b$.
Indication : on peut regarder d'abord le cas $(E, \vee, \wedge) = (\mathcal{P}(S), \cup, \cap)$, et s'en inspirer pour le cas général.

Question 6. Construire un plongement de E dans $(\mathcal{P}(E), \cup, \cap)$. En déduire que la dimension d'un treillis distributif fini est finie.

Couvrir. Soit a et b deux éléments d'un treillis distributif $(E, \vee, \wedge)$, et $<$ la relation d'ordre strict correspondant à l'ordre partiel de la question 2 (c'est-à-dire : $a < b \Leftrightarrow (a \wedge b = a$ et $a \neq b)$. On dit que b *couvre* a, noté $a \prec b$, si $a < b$ et $\neg(\exists c, a < c < b)$.
Couverture. La *couverture* de b est : $\mathsf{couv}(b) = |\{a : a \prec b\}|$.

Question 7. Soit a_1, a_2, b distincts tels que $a_1 \prec b$ et $a_2 \prec b$. Montrer $a_1 \vee a_2 = b$.

Question 8. Soit $E \subset \mathbb{Z}^d$ un sous-ensemble fini de $\mathbb{Z}^d$. On se place dans le treillis distributif $(E, \max_d, \min_d)$. Soit $b \in E$. Montrer $\mathsf{couv}(b) \leq d$.

Question 9. Soit E un treillis distributif fini. Montrer $\dim E \geq \max_{a \in E} \mathsf{couv}(a)$.

> Soit $(S, \leq)$ un ordre partiel fini.
> **Chaîne.** Une *chaîne* est un ensemble d'éléments de S deux à deux comparables pour $\leq$.
> **Antichaîne.** Une *antichaîne* est un ensemble d'éléments de S deux à deux incomparables pour $\leq$.
> **Idéal.** Un *idéal* est un ensemble $I \subseteq S$ tel que $b \in I \Rightarrow \forall a \leq b, a \in I$.
> **Théorème** *(Dilworth).* Si k est la cardinalité de la plus grande antichaîne de S, alors il existe une partition de S en k chaînes.

Question 10. Soit $\mathsf{Id}(S)$ l'ensemble des idéaux de S. Montrer que $(\mathsf{Id}(S), \cup, \cap)$ est un treillis distributif.

Question 11. En admettant le théorème de Dilworth, montrer :

$$\dim \mathsf{Id}(S) = \max_{I \in \mathsf{Id}(S)} \mathsf{couv}(I).$$

Indication : montrer que $\max_{I \in \mathsf{Id}(S)} \mathsf{couv}(I)$ est égal à la taille de la plus grande antichaîne dans S.

Ordonnancement

On cherche à ordonnancer n tâches indépendantes $T_1, \ldots, T_n \in \mathcal{T}$ sur un seul processeur, qui ne peut exécuter qu'une seule tâche à la fois. Chaque tâche T_i est décrite par deux entiers d_i et w_i représentant respectivement la date limite avant laquelle une tâche doit s'exécuter, et la pénalité à payer si la tâche est exécutée en retard. Le temps est représenté par des entiers naturels. Chaque tâche s'exécute en une unité de temps. Un ordonnancement est une fonction $\sigma : \mathcal{T} \to \mathbb{N}$ associant à chaque tâche sa date d'activation. La première tâche peut être activée à la date 0. On dit qu'une tâche est ordonnancée à l'heure si elle finit son exécution avant ou à sa date limite, et qu'elle est en retard sinon.
Le but est de trouver un ordonnancement qui minimise la somme des pénalités de retard. Les ordonnancements qui minimisent la somme des pénalités de retard sont dits optimaux.

Question 1. Quelle condition l'ordonnancement σ doit-il satisfaire pour être bien formé ?

Solution : Il n'y a qu'un seul processeur, deux tâches ne peuvent pas s'exécuter en même temps :

$$\forall i, j, i \neq j \implies \sigma(T_i) \neq \sigma(T_j)$$

Dit autrement, σ doit être injective.

Un ordonnancement est dit canonique si :

- Les tâches à l'heure sont exécutées avant les tâches en retard.
- Les tâches à l'heure sont ordonnées par date limite croissantes.

Question 2. Montrer qu'il existe un ordonnancement optimal qui est canonique.

Solution : Soit σ un ordonnancement optimal.

- Supposons qu'une tâche à l'heure T_i ait lieu après une tâche en retard T_j. L'échange des deux tâches ne change rien (ou cela contredit l'optimalité).
- Supposons que l'on ait deux tâches à l'heure $\sigma(T_i) < \sigma(T_j)$ et $d_i > d_j$. On peut échanger ces deux tâches : T_j reste à l'heure car elle est exécutée avant, et T_i car T_j était à l'heure et $d_j < d_i$.

Question 3. On suppose que les tâches sont ordonnées par pénalité décroissante. Écrire un algorithme glouton qui résout le problème. Quelle est sa complexité ? On ne cherchera pas à prouver l'optimalité de cet algorithme dans cette question.

Solution :

1. Trier les tâches par pénalité décroissante.
2. Itérer sur les tâches :
   (a) Considérer un nouvel ordonnancement où la tâche courante est insérée dans l'ordonnancement en préservant l'ordre sur les dates limites.
   (b) Si ce nouvel ordonnancement a des tâches en retard, on continue avec l'ancien ordonnancement.
3. Ajouter toutes les tâches en retard dans un ordre arbitraire (il faudra de toute façon payer les pénalités).

Complexité $\mathcal{O}(n^2)$ (car il faut tester à chaque fois si toutes les tâches sont à l'heure.)

Question 4. Illustrer l'algorithme sur l'exemple suivant :

$$w_1 = 7, w_2 = 6, w_3 = 5, w_4 = 4, w_5 = 3, w_6 = 2, w_7 = 1$$
$$d_1 = 4, d_2 = 2, d_3 = 4, d_4 = 3, d_5 = 1, d_6 = 4, d_7 = 6$$

Solution : Itérations :

— 1

— 2, 1

— 2, 1, 3

— 2, 4, 1, 3

— On ne peut pas ajouter T_5 car dans l'ordre canonique 5, 2, 4, 1, 3, T_3 est en retard

— On ne peut pas ajouter T_6, le résultat final est donc 2, 4, 1, 3, 7, avec un coût de 5.

Au final, l'ordonnancement des tâches à l'heure donne 2, 4, 1, 3, 7. Il faut ensuite ajouter les tâches 5 et 6 dans n'importe quel ordre, cela ne change rien car elles seront en retard et qu'il faudra payer la pénalité.

Soit S un ensemble à n éléments, $\mathcal{I} \in \mathcal{P}(S)$. $(S, \mathcal{I})$ est un matroïde s'il satisfait les propriétés suivantes :

1. Hérédité : $X \in \mathcal{I} \implies (\forall Y \subset X, Y \in \mathcal{I})$
2. Échange : $(A \in \mathcal{I}, B \in \mathcal{I}, |A| < |B|) \implies \exists x \in B \setminus A$ tel que $A \cup \{x\} \in \mathcal{I}$.

Les $X \in \mathcal{I}$ sont appelés des indépendants.
On dit qu'un indépendant $F \in \mathcal{I}$ est maximal s'il n'existe pas de $x \in S \setminus F$ tel que $F \cup \{x\} \in \mathcal{I}$.

Un matroïde pondéré est un matroïde enrichi d'une fonction de poids $w : S \to \mathbb{N}$. Le poids d'un sous-ensemble $X \subseteq S$ est la somme des poids des éléments : $w(X) = \sum_{x \in X} w(x)$.

On considère l'algorithme suivant, cherchant à trouver un indépendant de poids maximal :

```
1: fonction IndependantMax(M = (S, I), w : S → ℕ)
2:     S ← tri_décroissant(S, w)                ▷ Ainsi : w(S[0]) ≥ ... ≥ w(S[n − 1])
3:     A ← ∅
4:     pour i = 0 to n − 1 faire
5:         si A ∪ {S[i]} ∈ I alors
6:             A ← A ∪ {S[i]}
7:         fin si
8:     fin pour
9:     renvoyer A
10: fin fonction
```

Question 5. Soit x le premier élément de S (trié par ordre décroissant de poids) tel que $\{x\}$ est indépendant. Montrer que si x existe, alors il existe une solution optimale A qui contient x.

Solution : Soit B une solution optimale. Si $x \in B$, c'est bon. On observe que $w(x) \geq w(y) \forall y \in B$, par tri de S et le fait que $\{y\}$ est indépendant (hérédité de B). On pose $A = \{x\}$ et on ajoute (par la propriété d'échange) des éléments de B vers A jusqu'à ce que $|A| = |B|$. On a alors $A = (B \setminus \{y\}) \cup \{x\}$. Ainsi $w(A) \geq w(B)$. Comme B est une solution optimale, $w(A) = w(B)$.

Question 6. Montrer que l'algorithme retourne une réponse optimale, puis donner la complexité de cet algorithme.

Solution : On prouve la propriété par récurrence sur la taille de S. - Initialisation triviale. - Hérédité : supposons la propriété vraie au rang n. Soit $M = (S, I)$ un matroïde pondéré avec S de taille $n + 1$. Soit x le premier élément choisi par l'algorithme, on sait qu'il existe une solution optimale qui contient x. On applique l'hypothèse de récurrence sur $M' = (S \setminus \{x\}, \{X \subseteq S \setminus \{x\} | X \cup \{x\} \in I\})$ (qui est aussi un matroïde) pour obtenir A'. On note $A = A' \cup \{x\}$ est un indépendant. A' était une solution de poids maximal sur M', donc A est une solution de poids maximal sur M.

Complexité : $\mathcal{O}(n \log n + nf(n))$ où $f(n)$ est le temps pour tester si un ensemble d'au plus n éléments est indépendant.

Question 7. Prouver l'optimalité de l'algorithme d'ordonnancement de la question 3.

Solution : S est l'ensemble des tâches, les indépendants sont les ensembles de tâches qui peuvent être exécutées à l'heure.

- Hérédité : immédiat.
- Échange : soient $(A, B) \in \mathcal{I}^2$ avec $|A| < |B|$. On cherche $T_i \in B$ telle que $A \cup \{T_i\}$ soit encore exécutable à l'heure.

  On note $N_t(A)$ le nombre de tâches de A avec une date limite <= à t.

  On montre le résultat intermédiaire suivant : A est indépendant ssi $\forall 0 \leq t \leq n, N_t(A) \leq t$

  - (1) -> (2) par contraposée on aura trop de tâches.
  - (2) -> (1) la ième plus grande date limite est au plus i, donc il n'y a pas de soucis d'ordonnancement

  On pose $k = max_t N_t(A) \geq N_t(B)$ (ce maximum existe car $N_0(A) = N_0(B) = 0$). En particulier $N_{k+1}(B) > N_{k+1}(A)$: il y a donc plus de tâches avec date limite $k + 1$ dans B que dans A. On choisit donc $T_i \in B \setminus A$ avec $d_i = k + 1$.

  $A' := A \cup \{T_i\}$ est un indépendant, on le montre en prouvant que $\forall t, N_t(A') \leq t$ puis en utilisant le lemme ci-dessous. Pour $0 \leq i \leq t'$, $N_t(A') = N_t(A) \leq t$ car A indep. Pour $t \leq i \leq n, N_t(A') \leq N_t(B) \leq t$.

Jeu des connaissances

On considère un graphe non orienté $G = (V, E)$ où V est un ensemble de *sommets* et E un ensemble d'arêtes ($E \subseteq V^2$). On note $N = |V|$.
On place N agents sur les sommets. Ce placement peut être modélisé par une bijection p de V vers $\{1, \ldots, N\}$ qui, à chaque sommet, associe l'agent qui l'occupe.
Si deux agents sont sur des sommets voisins, on dit qu'ils ont *fait connaissance*. Une étape du *jeu des connaissances* consiste à échanger un nombre quelconque de paires disjointes d'agents voisins ; donc, à composer p par une série de transpositions de supports disjoints.
L'objectif est que toutes les paires d'agents $\{i, j\}, 1 \leq i, j \leq N$ aient fait connaissance suite à un nombre minimal d'étapes.

Question 1. Montrer que pour toute stratégie en t étapes, il existe une stratégie en $2t$ étapes qui ramène les agents à leur position initiale.

FIGURE 1 – Graphe-ligne à N sommets.

Question 2. Soit un graphe-ligne comme sur la figure ci-dessus, où les sommets sont numérotés de 1 à N dans l'ordre de la ligne, et identifiés avec ces numéros. Un placement est donc une bijection de $\{1, \ldots, N\}$; le placement initial est l'identité. Soit p un placement obtenu après t étapes du jeu. Montrer que pour tous agents i, j tels que $i > j$, si $p^{-1}(i) < p^{-1}(j)$, alors les agents i et j ont fait connaissance.

Question 3. Soit un graphe ligne à $2N$ sommets. Montrer qu'en répétant N fois les deux étapes suivantes : 1) pour tout $0 \leq i < N$, échanger les agents positionnés sur les sommets X_{2i+1}, X_{2i+2} ; 2) pour tout $1 \leq i < N$, échanger les agents positionnés sur les sommets X_{2i}, X_{2i+1} ; tous les agents font connaissance.

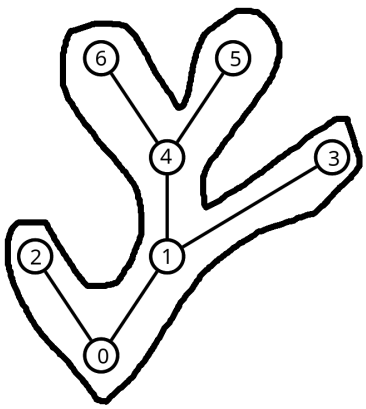


FIGURE 2 – Illustration d'un contour

Soit un arbre $\mathcal{T}$ à N sommets enraciné en X. On suppose qu'il existe un ordre total sur l'ensemble de ses sommets V. Le *contour* $C(\mathcal{T})$ (voir figure ci-dessus) est une séquence de sommets $(X_1 \dots X_k) \in V^k$ définie ainsi :

- Si $\mathcal{T}$ est une feuille X, $C(\mathcal{T}) = (X)$
- Si $\mathcal{T}$ possède des sous-arbres $\mathcal{T}_1, \dots, \mathcal{T}_i$, ordonnés selon leurs racines : $C(\mathcal{T}) = (X \cdot C(\mathcal{T}_1) \cdot X \cdot C(\mathcal{T}_2) \cdots X \cdot C(\mathcal{T}_i) \cdot X)$

où $\cdot$ est une concaténation des séquences.

Question 4. Donner l'expression du contour de la figure 2, où les sommets sont identifiés avec leurs numéros (que l'on prend dans l'ordre croissant) et la racine est le sommet 0.

Question 5. Montrer les propriétés suivantes du contour $C(\mathcal{T})$:

1. Chaque feuille X apparaît exactement une fois
2. Chaque sommet non-feuille avec c enfants apparaît exactement $c+1$ fois
3. Pour chaque sommet non-feuille X, de hauteur h :
   - à la première occurrence de X dans le contour, le sommet précédent (s'il y en a un) est de hauteur $h-1$ et le sommet suivant (s'il y en a un) est de hauteur $h+1$;
   - à la dernière occurrence de X, c'est l'inverse ;
   - pour toutes les autres occurrences de X, les deux sommets voisins sont de hauteur $h+1$.

Question 6. En déduire qu'on peut réécrire $C(\mathcal{T})$ sous la forme :

$$M_1 Y_1 D_1 Z_1 M_2 Y_2 D_2 Z_2 M_3 Y_3 \dots Y_k D_k$$

où : 1) chaque Y_i est une feuille ; 2) chaque Z_i est un sommet non-feuille ; 3) les M_i, D_i sont des sous-séquences de sommets non-feuilles (potentiellement vides) ; 4) tout sommet non-feuille n'apparaît au plus qu'une fois dans les sous-séquences M_i, et au plus une fois dans les sous-séquences D_i.

Question 7. On va maintenant placer les agents sur ce contour, c'est-à-dire que pour chaque sommet X, on sélectionne une des occurrences de X dans $C(\mathcal{T})$ et on « marque » cette occurrence avec l'agent correspondant.

Déduire de la question précédente une stratégie de placement pour laquelle la distance entre deux agents successifs, dans la séquence du contour, est d'au plus trois sommets.

Question 8. On admet qu'un graphe de degré d peut être colorié avec $d+1$ couleurs, de sorte que deux sommets voisins ont une couleur différente.

Montrer qu'il existe une stratégie à $O(Nd)$ étapes pour le jeu des connaissances sur un arbre à N sommets de degré d.

Indice : on déterminera d'abord une stratégie en $O(N^2)$ transpositions d'agents. On déterminera ensuite une stratégie en $O(Nd)$ étapes du jeu.

Bisimulation dans CCS

Le langage CCS (*Calculus of Communicating Systems*) est un langage de description de processus concurrents. On s'intéresse d'abord à une version simplifiée du langage, dont la syntaxe, construite à partir d'un ensemble infini d'*actions* $a, b, c, \ldots$ est donnée par :

$$\begin{array}{lll} P, Q & ::= & 0 \mid P \mid\mid Q \mid \alpha.P \mid P + Q \\ \alpha & ::= & a \mid \overline{a} \end{array}$$

0 est le processus inactif ; $P \mid\mid Q$ est la composition parallèle des processus P et Q ; $a.P$ (respectivement $\overline{a}.P$) est le processus qui effectue l'action a (respectivement, le dual de l'action a) puis se comporte comme P ; $P + Q$ est le choix non-déterministe entre P et Q.
Une relation de *congruence structurelle* $\equiv$ permet d'identifier des processus qui s'écrivent différemment mais représentent le même objet :

- $P \mid\mid Q \equiv Q \mid\mid P$ et $P \mid\mid (Q \mid\mid R) \equiv (P \mid\mid Q) \mid\mid R$ et $0 \mid\mid P \equiv P$,
- $P + Q \equiv Q + P$ et $P + (Q + R) \equiv (P + Q) + R$ et $0 + P \equiv P$.
- si $P \equiv P'$, alors $\alpha.P \equiv \alpha.P'$,
- si $P \equiv P'$ et $Q \equiv Q'$, alors $P \mid\mid Q \equiv P' \mid\mid Q'$ et $P + Q \equiv P' + Q'$.
- $\equiv$ est réflexive, symétrique et transitive.

Dans la suite, on considéra les processus « modulo $\equiv$ », c'est-à-dire qu'on manipulera des classes d'équivalence de processus pour $\equiv$.
Une *étiquette* l est soit une action (ou une action duale) α soit une *synchronisation* τ. La sémantique de CCS (la manière dont les processus évoluent) est donnée par la relation de *transition etiquetée* suivante :

- $\alpha.P \longrightarrow^{\alpha} P$ (l'action α est jouée)
- $a.P \mid\mid \overline{a}.Q \longrightarrow^{\tau} P \mid\mid Q$ (l'action a et l'action duale $\overline{a}$ se synchronisent)
- si $P \longrightarrow^{l} P'$ alors $P \mid\mid Q \longrightarrow^{l} P' \mid\mid Q$
- si $P \longrightarrow^{l} P'$ alors $P + Q \longrightarrow^{l} P'$
- si $P \longrightarrow^{l} P'$, $P \equiv Q$ et $P' \equiv Q'$, alors $Q \longrightarrow^{l} Q'$.

On omettra les occurences de 0 en fin de processus, par exemple on écrira $a.b$ plutôt que $a.b.0$. On considérera que le préfixe . est prioritaire sur la somme $+$, qui est prioritaire sur la composition parallèle $\mid\mid$: ainsi $a.b + c \mid\mid d$ est $((a.b) + c) \mid\mid d$.

Un *graphe de transition de* P est une représentation des processus accessibles depuis P par transitions successives, sous forme de graphe où :

- les sommets sont des classes d'équivalence pour $\equiv$ de processus différentes et
- les arêtes des transitions étiquetées entre les processus.

A titre d'exemple, le graphe de transition du processus $a.\overline{b} \mid\mid \overline{a}$ est donnée dans la Figure 1. On constate, entre autres, que $a.\overline{b} \mid\mid \overline{a}$ peut effectuer trois transitions :

- $a.\overline{b} \mid\mid \overline{a} \longrightarrow^{a} \overline{b} \mid\mid \overline{a}$ (on a joué l'action a)
- $a.\overline{b} \mid\mid \overline{a} \longrightarrow^{\overline{a}} a.\overline{b} \mid\mid 0 \equiv a.\overline{b}$ (on a joué l'action duale $\overline{a}$)
- $a.\overline{b} \mid\mid \overline{a} \longrightarrow^{\tau} \overline{b} \mid\mid 0 \equiv \overline{b}$ (on a synchronisé a et $\overline{a}$).

Question 1. Donner les graphes de transition des cinq processus suivants :

(1.) 0 (2.) $a.b + a.c$ (3.) $a.(b + c)$ (4.) $a.\overline{a} + \overline{a}.a$ (5.) $a \mid\mid \overline{a}$

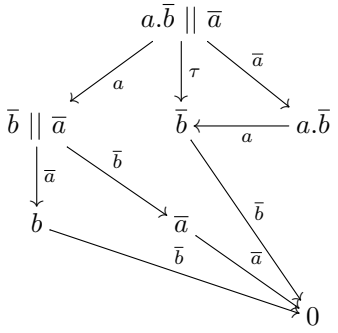


FIGURE 1 – Graphe de transition de $a.\overline{b} \mid\mid \overline{a}$.

Question 2. Montrer qu'un graphe de transition d'un processus est fini.

On ajoute la récursion à la syntaxe du langage :
$P, Q ::= [\cdots] \mid X \mid \mu X.P$ avec X appartenant à un ensemble infini de *variables de récursion*
Et la règle de congruence structurelle suivante :

$$\mu X.P \equiv P[\mu X.P/X]$$

où $P[Q/X]$ est le processus obtenu en remplaçant dans P chaque occurence de X par le processus Q.
En outre, si $P \equiv P'$, alors $\mu X.P \equiv \mu X.P'$.

Par exemple, on a $\mu X.a.X \equiv a.\mu X.a.X \equiv a.a.\mu X.a.X \equiv \ldots$

Question 3. Donner le graphe de transition des processus suivants :

$$(1.) \quad \mu X.(a.X + b) \quad (2.) \quad a.\mu X.b.a.X \quad (3.) \quad \mu Y.a.b.a.b.Y$$

Question 4. Montrer qu'un graphe de transition peut être infini.

Un ensemble E et une relation d'ordre $\leq$ sur E forment un *treillis complet* $(E, \leq)$ quand toute partie $S \subseteq E$ admet une borne inférieure et une borne supérieure pour $\leq$.

Soit f croissante sur un treillis complet $(E, \leq)$.

Question 5. Montrer que si x est un point pré-fixe de f, c'est-à-dire si $x \leq f(x)$, alors $f(x)$ en est un aussi.

Question 6. Montrer que la borne supérieure des points pré-fixes de f est le plus grand point fixe de f.

Question 7. Montrer le Lemme de Knaster-Tarski :

Toute fonction f croissante sur un treillis complet admet un plus petit et un plus grand point fixe.

Soit $\mathcal{P}$ l'ensemble des processus de CCS et $\mathcal{L}$ l'ensemble des étiquettes.
Une relation $\mathcal{R} \subseteq \mathcal{P} \times \mathcal{P}$ est une *bisimulation* si pour tout couple $(P, Q) \in \mathcal{R}$:

— pour tout $(l, P') \in \mathcal{L} \times \mathcal{P}$, si $P \longrightarrow^l P'$, alors il existe $Q' \in \mathcal{P}$ tel que $Q \longrightarrow^l Q'$ et $(P', Q') \in \mathcal{R}$.

— pour tout $(l, Q') \in \mathcal{L} \times \mathcal{P}$, si $Q \longrightarrow^l Q'$, alors il existe $P' \in \mathcal{P}$ tel que $P \longrightarrow^l P'$ et $(P', Q') \in \mathcal{R}$.

La bisimilarité (notée $\sim$) est la plus grande bisimulation (pour l'inclusion).

La définition de la bisimilarité n'est pas inductive : elle est récursive, mais ne contient pas de cas de base (on dit qu'elle est *coinductive*).

Question 8. Montrer que la bisimilarité peut être formellement définie, de manière unique, par le théorème de Knaster-Tarski.

Question 9. Montrer que les processus 2. et 3. de la question 3. sont *bisimilaires*, c'est-à-dire qu'ils appartiennent (en tant que couple de processus) à la bisimilarité.

Question 10. Montrer que les processus 2. et 3. de la question 1. ne sont pas bisimilaires.

Graphes, rayons et applications

Pour X un ensemble, on note $|X|$ son cadinal.
Dans la suite on considère des graphes non orientés $G = (V, E)$ où V est un ensemble de sommets et E un ensemble d'arêtes, c'est-à-dire, un ensemble $E \subseteq \{\{x, y\} \mid (x, y) \in V^2 \text{ et } x \neq y\}$. Dans la suite, la *taille* de G est le nombre de sommets ($|V|$). On dit qu'il est infini si V est infini et dénombrable si V est infini et dénombrable.
Pour $0 \leq n$, un chemin est une suite finie $s_0, \ldots, s_n$ de sommets telle que pour tout entier $0 \leq i < n$ on a $\{s_i, s_{i+1}\} \in E$. Ainsi, une suite de un sommet est un chemin mais pas la suite vide. Le chemin est dit *simple* si tout sommet y apparaît au plus une fois.
Un *rayon* de G est une suite infinie de sommets $s_0, \ldots, s_n, \ldots$ telle que pour tout entier $0 \leq i$, la suite $s_0, \ldots, s_i$ est un chemin simple.
Un graphe est dit *connecté* si pour tout sommet $s, t \in V$, il existe un chemin de s à t.

Question 1. Montrez que si un graphe G admet un rayon, alors il est infini.

Question 2. Un graphe $G = (V, E)$ est dit localement fini si pour tout $s \in V$, l'ensemble $N_s := \{x \mid \{s, x\} \in E\}$ est fini. Il est localement borné s'il existe une constante M, telle que pour tout s, $|N_s| < M$. Donnez :

— Un exemple d'un graphe infini, connecté et localement borné.

— Un exemple de graphe infini, connecté, localement fini mais pas localement borné.

Dans la suite, on dit qu'un graphe admet la propriété (K) s'il est connecté, infini, et localement fini. Nous allons montrer le résultat suivant :

Tout graphe dénombrable qui vérifie (K) admet un rayon.

Question 3. Est-ce que chacune des trois propriétés de (K) sont nécessaires pour que ce théorème soit vrai ?

Question 4. Montrez qu'il existe un graphe G infini et localement fini tel que pour tout k, G possède un chemin simple de taille k mais tel que G n'admette pas de rayon.

Question 5. Étant donné un graphe $G = (V, E)$ et un sommet v de G. Le graphe $G_v = (V', E')$ avec $V' = V \setminus \{v\}$ et $E' = E \cap \{\{x, y\} \mid x \in V'\}$. Montrez que si G satisfait (K) et v est un sommet de G, alors G_v est une union disjointe finie de graphes connexes dont au moins l'un d'entre eux satisfait (K).

Question 6. Montrez que si G satisfait (K) alors il admet un rayon.

Application à la coloration de graphes Soit $G = (\mathbb{N}, E)$ un graphe. Une k-coloration de G est une fonction $f : \mathbb{N} \to \{0, \ldots, k-1\}$ tel que pour tout $\{x, y\} \in E$ on a $f(x) \neq f(y)$. On dit que G est k-coloriable s'il existe une k-coloration de G.

Question 7. Montrez qu'un graphe dénombrable G est k-coloriable si et seulement si tout ses sous-graphes finis sont k-coloriables.

Application aux mots infinis Soit Σ un alphabet. On note Σ^* l'ensemble des mots finis sur Σ et Σ^∞ l'ensemble des mots infinis. Dans la suite, on pose $L \subseteq \Sigma^*$ un langage arbitraire et $u \in \Sigma^\infty$ un mot infini. Un découpage de u est une suite infinie de mots finis $v_0, \ldots, v_n, \ldots$ telle que $u = v_0 v_1 \cdots v_n \cdots$.

Question 8. Montrez qu'il existe un découpage de u tel qu'à partir d'un certain rang, soit tous les mots appartiennent à L, soit aucun mot n'appartient à L.

Langages réguliers sans-étoiles et monoïde apériodique

Dans la suite, les alphabets sont considérés comme des ensembles **finis** de taille au moins deux.. Pour un alphabet Σ, on note Σ^* l'ensemble des mots finis et ε le mot vide. Pour $L \subseteq \Sigma^*$, on note L^c le complément de L, c'est à dire, l'ensemble des mots de Σ^* qui ne sont pas dans L. On définit une expression *sans-étoiles* par induction :

- Chaque lettre a est une expression de base.
- Pour E et F deux expressions sans-étoiles alors
  - E^c, $E \cap F$ et $E \cup F$ sont sans-étoiles.
  - EF est sans-étoiles

Pour E une expression sans-étoile, on lui associe un langage $\mathcal{L}(E)$ par induction de la manière suivante :

- $\mathcal{L}(a) = \{a\}$
- $\mathcal{L}(E^c) = \mathcal{L}(E)^c$, $\mathcal{L}(E \cap F) = \mathcal{L}(E) \cap \mathcal{L}(F)$ et $\mathcal{L}(E \cup F) = \mathcal{L}(E) \cup \mathcal{L}(F)$
- $\mathcal{L}(EF) = \mathcal{L}(E)\mathcal{L}(F) = \{uv \mid u \in E \text{ et } v \in F\}$

On remarque que pour un langage L donné, il existe possiblement deux expressions différente E et F tel que $\mathcal{L}(E) = L = \mathcal{L}(F)$. On pourra néanmoins, quand cela est opportun, confondre un langage et son expression par abus de notation.

Question 1. Montrez que les langages suivants admettent des expressions sans-étoiles :

- $\{\}$ (l'ensemble vide) et Σ^*
- B^* pour $B \subseteq \Sigma$,
- $\{\varepsilon\}$ (l'ensemble qui contient le mot vide)

Question 2. Pour cette question, on fixe $\Sigma = \{a, b, c\}$. Donnez sans justifier l'automate déterministe minimal pour le langage $\Sigma^* aa \Sigma^*$ en admettant qu'il a 3 états.

Question 3. Montrez que le langage $K := (ab)^*$ est sans-étoiles.

Dans la suite, pour $u \in \Sigma^*$ et $L \subseteq \Sigma^*$, on note $u^{-1}L = \{v \mid uv \in L\}$ et $Lu^{-1} = \{v \mid vu \in L\}$.
Pour E une expression, on définit $h(E)$ la hauteur de concaténation d'une expression sans-étoiles par induction.
Les expressions de base ont pour hauteur de concaténation 0.
Pour E et F, $h(E \cap F) = h(E \cup F) = \max(h(E), h(F))$ et $h(E^c) = h(E)$. Enfin, $h(E.F) = \max(h(E), h(F)) + 1$. Autrement dit, pour une expression sans-étoiles, sa profondeur de concaténation est le nombre maximal de concaténations emboîtées.
Par exemple, $(\{a\}\{a\})\{b\}$ est de profondeur 2.

Question 4. Montrez que si L est définissable par une expression sans-étoiles, alors pour tout $u \in \Sigma^*$, les langages $u^{-1}L$ et Lu^{-1} sont également définissables par des expressions sans-étoiles de même profondeur de concaténation.

Dans la suite, on dit que $\eta \colon \Gamma^* \to \Sigma^*$ est un morphisme (de monoïdes) si pour tout $u, v \in \Gamma^*$, on a $\eta(uv) = \eta(u)\eta(v)$.

Question 5. Montrez que si $L \subseteq \Sigma^*$ est définissable par une expression sans-étoiles et qu'on prend un morphisme $\eta \colon \Gamma^* \to \Sigma^*$, alors $\eta^{-1}(L)$ est définissable par une expression sans-étoiles.

L'objectif est maintenant de montrer que pour tout $r > 1$, le langage $(a^r)^*$ n'est pas exprimable par une expression sans-étoiles. Pour ce faire, nous allons montrer par induction sur les expressions sans-étoiles le résultat suivant :

Soit L un langage défini par une expression sans-étoiles. Il existe $N \in \mathbb{N}$ tel que pour tout $n > N$, pour tout $i < r$, on a $(a^r)^n \in L$ ssi $(a^r)^n a^i \in L$.

Question 6. Expliquez pourquoi les cas de base vérifient cette hypothèse d'induction.

Question 7. En supposant que deux langages réguliers L et K qui vérifient cette hypothèse d'induction, montrer que leur union, complément et intersection la vérifient également.

Question 8. En supposant que deux langages réguliers L et K vérifient cette hypothèse d'induction, montrer que leur concaténation la vérifie également.

Dans la suite, on dit qu'un langage vérifie la propriété (S) s'il existe N tel que pour tous mots u, x, y, et tout entier $n > N$, on a $xu^n y \in L$ si et seulement $xu^{n+1}y \in L$.

Question 9. En déduire qu'un langage régulier L exprimable par une expression sans-étoiles vérifie (S).

Nous avons ainsi prouvé le sens direct du théorème de Schützenberger qui dit qu'un langage est sans-étoiles si et seulement s'il satisfait (S).

Question 10. En admettant le sens retour du théorème de Schützenberger, proposez un algorithme pour vérifier cette propriété sur un langage régulier donné par un automate déterministe. Faites en l'analyse de complexité.

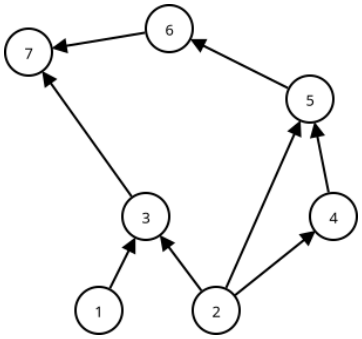
Jeu des jetons

Un graphe orienté G est défini par un ensemble fini de sommets V et d'arêtes $E \subseteq V \times V$. Pour un sommet $x \in V$, ses *prédécesseurs* forment l'ensemble des sommets $y \in V$ tels que $(y, x) \in E$ et ses *successeurs* forment l'ensemble des sommets $z \in V$ tels que $(x, z) \in E$. On suppose de plus que le graphe ne contient pas de cycle, y compris des arêtes d'un sommet vers lui-même.
On s'intéresse au problème suivant : on fixe un sommet v du graphe, appelé *sommet distingué*. On dispose d'un ensemble de *jetons* qui peuvent être placés sur les sommets du graphe, de sorte que chaque sommet ne comporte que zéro ou un jeton. S'il a un jeton, le sommet est dit *marqué*. On peut placer les jetons selon les règles suivantes :

1. Si x n'a aucun prédécesseur, on peut placer un jeton sur x
2. Si tous les prédécesseurs de x sont marqués, on peut placer un jeton sur x
3. On peut toujours retirer un jeton

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

Question 1. Soit le graphe représenté ci-contre, où "7" est le sommet distingué. Montrer qu'il existe une stratégie utilisant 4 jetons.



Solution : Comme pour le moment on peut enlever un jeton n'importe quand, il suffit de s'assurer qu'on arrive à marquer 7. Les étapes du marquage sont les suivantes.

Placer jeton	2	1	3		4	5			6	7
Enlever jeton				1			2	4		
Nombre de jetons	1	2	3	2	3	4	3	2	3	4

Question 2. Démontrer qu'un graphe orienté acyclique comporte au moins un sommet sans prédécesseur.

Solution : On peut procéder par l'absurde. Si tous les sommets ont au moins un prédécesseur, on peut former un cycle de la manière suivante : on sélectionne un sommet, puis un prédécesseur de ce sommet, et ainsi de suite. On finit forcément par trouver un sommet déjà rencontré : cela créée un cycle.

Question 3. En déduire que pour tout graphe orienté acyclique à n sommets, il existe toujours une stratégie pour résoudre le jeu avec n jetons.

Solution : Procéder par induction sur le nombre de sommets du graphe. Un graphe ne contenant aucun cycle contient toujours au moins un sommet qui n'a pas de prédécesseur. Placer un premier jeton sur ce sommet nous ramène au problème des jetons sur le graphe duquel on a enlevé ce sommet. Ce graphe est lui-même acyclique et de taille inférieure.

Question 4. Soit un arbre binaire (chaque sommet non-feuille a deux enfants) à ℓ feuilles, dans lequel les prédécesseurs d'un sommet sont ses enfants, et le sommet distingué est la racine r. Montrer qu'il existe une stratégie utilisant $\lceil \log_2 \ell \rceil + 2$ jetons, où $\lceil x \rceil$ est la partie entière supérieure de x.

Solution : On montre par récurrence que : tout arbre ayant *moins de* ℓ feuilles a une stratégie avec $\lceil \log_2 \ell \rceil + 2$ jetons.

Si l'arbre contient une seule feuille, il suffit d'un jeton (a fortiori 2). S'il en a 2, il faut 3 jetons. Si on prend maintenant un arbre à ℓ feuilles, la racine comporte deux sous-arbres, tous deux ayant $\ell - 1$ feuilles ou moins. L'un d'entre eux a plus de $\ell/2$ feuilles (disons ℓ'). On commence par celui-ci. On le marque en utilisant $\lceil \log_2 \ell' \rceil + 1 \leq \lceil \log_2 \ell \rceil + 1$ jetons (par hypothèse de récurrence). On laisse le jeton sur la racine de ce sous-arbre.

Ensuite, on s'occupe de l'autre sous-arbre, qui a moins de $\ell/2$ feuilles. On peut donc marquer sa racine en utilisant $\lceil \log_2(\ell/2) \rceil + 1 \leq \lceil \log_2 \ell \rceil$ jetons. Enfin, on marque la racine de l'arbre (à cette étape il n'y a plus que 3 jetons sur l'arbre). Le nombre de jetons utilisés est donc bien $\lceil \log_2 \ell \rceil + 1$.

On remplace maintenant la règle 3. par les deux règles suivantes :

3. On peut toujours retirer un jeton d'un sommet sans prédécesseurs
4. On ne peut retirer un jeton d'un sommet que si ses prédécesseurs sont tous marqués

À partir de maintenant, on considère uniquement le graphe-ligne $L_n := x_1 \to x_2 \to \ldots \to x_n$, où x_n sera toujours le sommet distingué.

Question 5. Montrer qu'avec ces nouvelles règles :

1. Il existe toujours une stratégie pour résoudre le jeu en utilisant n jetons et $O(n)$ étapes.
2. S'il existe une stratégie pour marquer x_n en k jetons et t étapes, alors il existe aussi une stratégie pour, en partant de la configuration finale (x_n est le seul sommet marqué), retrouver la configuration initiale (aucun sommet marqué) en k jetons et t étapes.

Solution : Pour la première question : on place des jetons sur tous les sommets x_1 jusqu'à x_n, et on les retire de x_{n-1} jusqu'à x_1 dans l'ordre inverse.

Pour la deuxième, on peut remarquer que les conditions pour placer ou pour enlever un jeton sont les mêmes (tous les prédecesseurs marqués). Par conséquent le jeu est réversible : en partant de la configuration finale on peut ré-appliquer les mêmes étapes dans l'ordre inverse. On n'attendait pas forcément une démonstration très formelle pour cette sous-question, qui est principalement utile pour les questions suivantes.

Question 6. Montrer qu'on peut marquer x_n en $O(n)$ étapes et $O(\sqrt{n})$ jetons.

Solution : Soit $m = \lfloor \sqrt{n} \rfloor$. La stratégie se découpe en plusieurs sous-étapes de la manière suivante :

— pour tout j de 1 à $m - 1$, marquer le sommet jm de sorte qu'aucun sommet entre $(j-1)m$ et jm n'est marqué : on utilise pour cela la stratégie naïve, qui utilise m jetons

— marquer le sommet n à l'aide de la stratégie naïve (qui utilise donc $\leq m$ jetons)

— pour tout j de $m - 1$ à 1, enlever le jeton placé sur le sommet jm. On utilise pour cela la stratégie naïve. En repartant du sommet $(j-1)m$, on marque tous les sommets jusqu'à jm. On retire le jeton du sommet jm, puis de $jm - 1$, et ainsi de suite jusqu'à arriver au seul $(j-1)m$ marqué.

La correction de cette stratégie est évidente : dans la configuration finale le sommet n est marqué et aucun sommet $1, \ldots, n-1$ ne l'est. Le nombre de jetons utilisés est de $O(\sqrt{n})$ pour les marquages intermédiaires et $O(\sqrt{n})$ pour les sous-chaînes. Le nombre d'étapes est $O(n)$.

Question 7. Montrer qu'on peut marquer x_n en utilisant au total $\lceil \log_2 n \rceil + 1$ jetons. Combien d'étapes comporte cette stratégie ?

Solution : On va décrire récursivement cette stratégie. Pour commencer, le sommet x_1 peut être marqué avec un seul jeton. Le sommet x_2 a besoin de deux jetons.

Comme dans la question précédente, on a besoin du fait que les stratégies de marquage sont réversibles, i.e., si on est capable d'arriver à la configuration où seul x_{2^k} est marqué, alors on est également capable, en partant de cette configuration, de retirer un jeton placé sur ce sommet (de sorte que tous les sommets de 1 à 2^k sont non marqués).

On montre donc par induction que :

> Pour tout k, en partant d'une configuration où aucun sommet n'est marqué, on peut marquer tout sommet en position $\leq 2^k$ avec $k+1$ jetons et $T(2^k)$ étapes, de sorte qu'aucun autre sommet n'est marqué.

Supposons la propriété vraie pour k. Pour tout sommet $2^k < i \leq 2^{k+1}$, on procède de la manière suivante :

- marquer le sommet numéro 2^k en utilisant un appel récursif. On utilise $k+1$ jetons ; à ce moment seul le sommet numéro 2^k est marqué ;
- marquer le sommet i. Pour ce faire, on considère la séquence d'étapes dans la stratégie de marquage du sommet $(i - 2^k)$, et on répète ces étapes, décalées de 2^k positions. Au lieu de partir de la racine, on part du sommet $2^k + 1$ (puisque 2^k est marqué). Comme le sommet 2^k est marqué, cela revient au même (on peut partir du sommet $2^k + 1$ comme on partait précédemment de la racine).

  On utilise $k+1$ jetons d'après l'hypothèse de récurrence, et comme 2^k reste marqué durant ces étapes, on a au total $k+2$ jetons. À ce stade seuls les sommets 2^k et i sont marqués.
- on utilise une troisième fois l'hypothèse de récurrence et la réversibilité des stratégies pour enlever le jeton du sommet 2^k. On a donc bien que seul i est marqué.

Le nombre d'étapes nécessaires est : $T(2^{k+1}) = 3T(2^k)$. Comme on a $T(1) = 1$ cela implique $\implies T(2^k) = 3^k$ étapes. Pour un entier n donné, on utilise donc de l'ordre de $n^{\log_2 3}$ étapes.

Question 8. On cherche maintenant à obtenir un meilleur compromis entre le nombre de jetons utilisés et le nombre d'étapes. Montrer que pour tout ε, il existe une stratégie utilisant $O(\log n)$ jetons et $O(n^{1+\varepsilon})$ étapes.

Solution : On peut généraliser la construction précédente, qui consistait à faire une récursion en coupant le problème en deux.

On va couper en ℓ parties récursivement. Soient T_k et J_k le nombre d'étapes et de jetons pour marquer tous les sommets à distance ℓ^k : $T_k \leq 2\ell T_{k-1}$ et $J_k = J_{k-1} + \ell - 1$, donc $T_k = (2\ell-1)^k \leq \ell^k 2^k$ et $J_k = (\ell-1)k$. (Remarquer qu'on retrouve bien le cas $\ell = 2$).

En posant $n = \ell^k$ et en faisant varier k on a donc : $k = \log_\ell n$ et donc $T \leq n 2^{\log n / \log \ell} = n^{1+\log 2/\log \ell}$ et $J = (\ell-1)\log_\ell n = O(\log n)$.

Évaluation efficace d'infixes dans les semigroupes finis

Pour un alphabet Σ, on note Σ^* l'ensemble des mots finis de Σ et Σ^+ les mots finis non vides. On note également ε le mot vide. sur Σ. Pour $u = u_0 \cdots u_{n-1} \in \Sigma^+$ et $0 \leq i < j < n$, on note $u[i : j]$ le mot $u_i \cdots u_{j-1}$. Dans la suite, on considère des semigroupes $(S, \cdot)$, c'est-à-dire, un ensemble muni d'une loi de composition interne associative. L'ensemble Σ^+ muni de la concaténation est un semigroupe.
Pour deux semigroupes T et S, une fonction $\eta : T \to S$ est un morphisme de semigroupes si pour tout $u, v \in T$, $\eta(uv) = \eta(u) \cdot \eta(v)$.

Dans la suite, on fixe un morphisme **surjectif** $\eta \colon \Sigma^+ \to S$ avec S qui est un semigroupe fini. En particulier, sa taille $|S|$ peut être supposée constante dans la suite. On étudie le problème suivant.

Étant donné un mot $u \in \Sigma^+$, on veut construire une structure de données pour le *problème infixe* de η, c'est-à-dire, une structure de données qui retourne $\eta(u[i : j])$.

La complexité d'une telle structure de données est évaluée en fonction de deux paramètres :

— La complexité en temps du précalcul pour construire la structure de données.
— La complexité en temps d'évaluation d'une requête infixe.

Toutes les complexités sont évaluées en fonction de la taille du mot n (et **pas** du semigroupe S). Dans la suite la complexité des opérations arithmétiques classiques sur $\log(n)$ bits est supposée constante.

Question 1. En autorisant un précalcul polynomial, quelle complexité pour l'évaluation d'une requête infixe peut on espérer ?

Question 2. Montrer que quand $|\Sigma| = 1$, l'évaluation d'une requête peut être réalisée en temps constant. Quel est la complexité du précalcul ?

Un semigroupe est un groupe, s'il existe un élément neutre e et si tout élément admet un inverse. Plus formellement, si pour tout x, $xe = ex = x$ et si pour tout élément $x \in S$, il existe un unique y tel que $xy = yx = 1$. On utilise dans la suite la notation x^{-1} pour désigner un tel élément y quand le semigroupe est un groupe.

Question 3. Montrer que si S est un groupe, alors il est possible d'avoir une structure de données avec un précalcul linéaire et un temps d'évaluation constant.

Question 4. Proposer une structure de données avec un précalcul en $O(n\sqrt{n})$ et une évaluation de requête en $O(1)$. On pourra considérer un découpage du mot d'entrée en $\sqrt{n}$ morceaux de taille $\sqrt{n}$.

Question 5. En déduire que pour tout ε, le problème peut être résolu avec une structure de données nécessitant un précalcul $O(n^{1+\varepsilon})$ et des requêtes en $O(1)$.

Dans la suite, pour tout mot $u \in \Sigma^+$, on définit la fonction $\varphi_u \colon S \to S$ tel que $\varphi_u(x) = \eta(u) \cdot x$.

Question 6. Montrer que si pour tout $a \in \Sigma$, les fonctions φ_a sont des bijections, alors il existe une structure de données avec un précalcul en $O(n)$ et des requêtes en $O(1)$.

Dans la suite, pour $\Sigma' \subseteq \Sigma$; on note $\eta_{|\Sigma'}$ la restriction de η à Σ'^+

Question 7. Montrer que s'il existe $a \in \Sigma$ tel que φ_a n'est pas une bijection, on peut construire une structure de données pour η avec des requêtes en $O(1)$ et un précalcul linéaire à partir de :

— une structure de données pour $\eta_{|\Sigma \setminus \{a\}}$ avec des requêtes en $O(1)$ et un précalcul linéaire
— une structure de données pour $\eta' \colon \Gamma^+ \to S'$ avec des requêtes en $O(1)$ et un précalcul linéaire Γ un alphabet quelconque et S' un semigroupe tel que $|S'| < |S|$.

Question 8. En déduire une structure de données avec un précalcul $O(n)$ et des requêtes en $O(1)$ dans le cas général.

Arithmétique à virgule flottante

On définit un format de nombre à virgules flottantes φ par trois entiers :

1. Une précision $p \in \mathbb{N}^*$
2. Un exposant minimal $e_{\min} \in \mathbb{Z}$
3. Un exposant maximal $e_{\max} \in \mathbb{Z}$, avec $e_{\min} < 0 < e_{\max}$.

Étant donné un format $\varphi = (p, e_{\min}, e_{\max})$, l'ensemble des nombres flottants définis via ce format est :

$$\begin{aligned}\mathbb{FP}_\varphi = \{(-1)^s \cdot M \cdot 2^{e-p+1} | s \in \{0,1\}, e \in \mathbb{Z}, e_{\min} \le e \le e_{\max}, M \in \mathbb{N}, 2^{p-1} \le M < 2^p\} \\ \sqcup \{(-1)^s \cdot M \cdot 2^{e_{\min}-p+1} | s \in \{0,1\}, M \in \mathbb{N}, 0 \le M < 2^{p-1}, \}\end{aligned}$$

La précision p correspond ainsi au nombre maximal de chiffres dans l'écriture de M en base 2.
Par exemple, le format double précision utilisé par la plupart des machines est $\varphi_{64} = (53, -1022, 1023)$.
On suppose que φ est fixé.
On note Ω le plus grand nombre flottant représentable, et $u = 2^{-p}$

Question 1. On suppose dans cette question que $p = 3$ et $e_{\max} \ge e_{\min} + 3$. Représenter schématiquement les nombres flottants sur $[0, 2^{e_{\min}+3}]$

Pour tout réel x tel que $|x| < \Omega$, on note $\mathtt{RN}(x)$ le nombre flottant le plus proche de x.

Question 2. La définition de $\mathtt{RN}(x)$ est-elle bien construite ?

Question 3. Montrer qu'il existe $a, b \in \mathbb{FP}$ tels que $|a + b| < \Omega$ et $a + b \notin \mathbb{FP}$.

Question 4. Montrer que pour $2^{e_{\min}} \le |x| \le \Omega$, $\mathtt{RN}(x) = x \cdot (1 + \delta)$ avec $|\delta| \le u$.

On note $+_{\mathbb{FP}}$ l'addition sur les flottants, qui satisfait $x +_{\mathbb{FP}} y = \mathtt{RN}(x + y)$ lorsque $|x + y| < \Omega$.
Dans la suite, on admet que l'on peut toujours utiliser l'équation de la question 4.

Question 5. Somme de n termes.
Soit $(x_1, \ldots, x_n) \in \mathbb{FP}^n$. On considère l'algorithme suivant :

```
1: fonction IterativeSum(x_1, ..., x_n)
2:     s ← x_1
3:     pour i = 2 to n faire
4:         s ← s +_FP x_i
5:     fin pour
6:     renvoyer s
7: fin fonction
```

On suppose que $n \cdot u < 1$. Montrer que

$$\left| \textsc{IterativeSum}(x_1, \ldots, x_n) - \sum_{i=1}^{n} x_i \right| \le \gamma_{n-1} \sum_{i=1}^{n} |x_i| \qquad \text{avec } \gamma_n = \frac{nu}{1 - nu}$$

On suppose que l'on dispose d'une primitive $\texttt{2SUM} : \mathbb{FP}^2 \to \mathbb{FP}^2$ telle que pour tous flottants a et b : si $(s,t) = \texttt{2SUM}(a,b)$, alors

1. $s + t = a + b$,
2. $s = \texttt{RN}(a+b)$,
3. $|t| \leq u|s|$,
4. $|t| \leq u|a+b|$.

Question 6. Somme compensée de n termes.

On considère l'algorithme suivant :

```
1: fonction CompensatedSum(x_1, ..., x_n)
2:     π_1 ← x_1
3:     σ_1 ← 0
4:     pour i = 2 to n faire
5:         π_i, q_i ← 2SUM(π_{i-1}, x_i)
6:         σ_i ← σ_{i-1} +_FP q_i
7:     fin pour
8:     renvoyer π_n +_FP σ_n
9: fin fonction
```

On suppose que $n \cdot u < 1$. Montrer que

$$\left| \textsc{CompensatedSum}(x_1, \ldots, x_n) - \sum_{i=1}^{n} x_i \right| \leq u \left| \sum_{i=1}^{n} x_i \right| + \gamma_{n-1}^2 \sum_{i=1}^{n} |x_i|$$

Justifier ensuite de l'intérêt de l'algorithme des sommes compensées.

Question 7. On admet les résultats suivants :

Lemme (Sterbenz). Si $x, y \in \mathbb{FP}$ tels que $\frac{y}{2} \leq x \leq 2y$, alors $x - y$ est exactement représentable.

Lemme. Soient $a, b \in \mathbb{FP}$, si $s = \texttt{RN}(a+b) < \Omega$, alors $r = (a+b) - s \in \mathbb{FP}$

Soient a, b deux nombres flottants tels que $a \geq b \geq 0$. Comment calculer $a+b$ de manière exacte en trois opérations ?

Ensembles de mots infinis

Soient Γ un ensemble non vide fini ou infini (dénombrable ou non) dont les éléments sont appelés des lettres, Γ^* l'ensemble des mots finis, ε le mot vide, $\Gamma^+ := \Gamma^* \setminus \{\varepsilon\}$, et $\Gamma^{\mathbb{N}}$ l'ensemble des mots infinis. Pour tout $E \subseteq \Gamma^*$ et $W \subseteq \Gamma^*$ (ou $W \subseteq \Gamma^{\mathbb{N}}$), on pose $EW := \{uv \mid u \in E \wedge v \in W\}$, où uv est la concaténation des mots u et v. Pour tout $W \subseteq \Gamma^{\mathbb{N}}$ on pose $\overline{W} := \Gamma^{\mathbb{N}} \setminus W$. Pour tout u mot de Γ^+, on note $u^\omega := uuu\cdots \in \Gamma^{\mathbb{N}}$. Plus généralement, pour $E \subseteq \Gamma^+$, on note E^ω l'ensemble des mots infinis de la forme $u_0u_1u_2\ldots$ où pour tout $n \in \mathbb{N}$ u_n est un mot de E. Pour tout $W \subseteq \Gamma^{\mathbb{N}}$ on pose $W_f := \{u \in \Gamma^+ \mid u^\omega \in W\}$. Soit $W \subseteq \Gamma^{\mathbb{N}}$ pour tout le sujet.

Question 1.
1. Que peut-on dire de $\Gamma W \cup \Gamma\overline{W}$? Et de $\Gamma W \cap \Gamma\overline{W}$?
2. Comparer $\Gamma\overline{W}$ et $\overline{\Gamma W}$.
3. Montrer que $\overline{W} = \Gamma\overline{W}$ ssi $W = \Gamma W$.

Question 2. Montrer que $W = \Gamma W$ ssi $\forall u \in \Gamma^* \forall v \in \Gamma^{\mathbb{N}}, v \in W \Leftrightarrow uv \in W$.

Question 3.
1. Comparer $(\overline{W})_f$ et $\Gamma^+ \setminus W_f$.
2. Supposons que $\forall u, v \in \Gamma^+, uv \in W_f \Rightarrow vu \in W_f$. Montrer que $\forall u, v \in \Gamma^+, uv \in (\overline{W})_f \Leftrightarrow vu \in (\overline{W})_f$.

Hypothèse 1 pour toute la suite : $\Gamma W = W$. On dit alors que W est préfixe-indépendant.
Hypothèse 2 pour toute la suite : $(W_f)^\omega \subseteq W$. On dit alors que W_f se mélange bien.

Question 4.
1. Soient $u, v \in \Gamma^+$. Montrer que $uv \in W_f$ ssi $vu \in W_f$. On dit alors que W_f commute.
2. Soient $u, v \in \Gamma^+$. Montrer que $uv \in (\overline{W})_f$ ssi $vu \in (\overline{W})_f$.
3. Soient $u, v \in W_f$. Montrer que $uv \in W_f$. On dit alors que W_f est stable par concaténation.
4. Soient $u, v \in (\overline{W})_f$. Montrer qu'on n'a pas nécessairement $uv \in (\overline{W})_f$.

Question 5. Soient $c \in \Gamma$ et $B \subseteq \Gamma$ tel que $cB \subseteq W_f$. Montrer que $\forall v \in B^+ \exists n > 0, c^n v \in W_f$.

Hypothèse 3 pour toute la suite : $\forall L, L' \subseteq \Gamma^+, (\forall v \in L \exists u \in L', uv \in W_f) \Rightarrow (\exists u \in L' \forall v \in L, uv \in W_f)$
Hypothèse 4 pour toute la suite : $((\overline{W})_f)^\omega \subseteq \overline{W}$. On dit alors que $(\overline{W})_f$ se mélange bien.

Question 6. Soient $c \in \Gamma$ et $B \subseteq \Gamma$ tel que $cB \subseteq W_f$. Montrer qu'il existe $n > 0$ tel que $c^n B^+ \subseteq W_f$.

Question 7. Soient $c \in \Gamma \cap W_f$ et $B \subseteq \Gamma$ tels que $cB \subseteq W_f$. Montrer que $cB^* \subseteq W_f$.

Pour tout $c \in \Gamma \cap W_f$, soit $g(c) := \{x \in \Gamma \setminus W_f \mid cx \in W_f\}$. Pour tout $c, c' \in \Gamma \cap W_f$, on pose $c \sqsubseteq c'$ si $g(c) \subseteq g(c')$, on pose $c \sqsubset c'$ si $c \sqsubseteq c'$ et $g(c) \neq g(c')$, et on pose $c \sim c'$ si $g(c) = g(c')$.

Question 8.
1. Montrer que $\sqsubseteq$ est un préordre total, i.e. une relation binaire réflexive, transitive et totale (i.e. $\forall x, y \in \Gamma \cap W_f, x \sqsubseteq y \vee y \sqsubseteq x$).
2. Montrer que $\sim$ est une relation d'équivalence avec un nombre fini de classes.

Question 9. Montrer qu'il existe $n \in N$ et $B_1, B_2, \ldots, B_{2n+1} \subseteq \Gamma$ tels que :

1. Les B_i sont disjoints deux à eux et leur union est Γ.
2. Pour tout $i \in \{1, \ldots, n\}$ on a $B_{2i} \subseteq W_f$
3. Pour tout $i \in \{0, \ldots, n\}$ on a $B_{2i+1} \subseteq (\overline{W})_f$.
4. Pour tout $i \in \{1, \ldots, n\}$ et $k \leq 2i$ on a $B_k B_{2i} \subseteq W_f$.
5. Pour tout $i \in \{0, \ldots, n\}$ et $k \leq 2i+1$ on a $B_k B_{2i+1} \subseteq (\overline{W})_f$.

Dualité de collections d'ensembles

Soit $(X, \leq)$ un ordre partiel, i.e. une relation réflexive, transitive et antisymétrique. On note $y < x$ si $y \leq x$ et $y \neq x$. L'ensemble des éléments minimaux pour $\leq$ de $A \subseteq X$ est noté $\min(A)$. On rappelle que $x \in \min(A)$ ssi $\forall y \in A, y \leq x \Rightarrow y = x$, ce qui équivaut aussi à $\forall y \in A, \neg(y < x)$. Une antichaîne de $(X, \leq)$ est une partie $Y \subseteq X$ d'éléments incomparables, i.e. $\forall x, y \in Y, \neg(x < y \vee y < x)$. Pour tout A, B tels que $B \subseteq A \subseteq X$, on dit que B est co-initial dans A, noté $B \leq_c A$, si $\forall x \in A \exists y \in B, y \leq x$.

Question 1.
1. La relation $\leq_c$ est-elle réflexive ? transitive ? antisymétrique ?
2. Soient A, B tels que $B \leq_c A$. Comparer pour l'inclusion B et $\min(A)$.
3. Soient A, B tels que $B \leq_c A$ et $b \in B \setminus \min(B)$. Que peut-on dire de $B \setminus \{b\}$?

Question 2. Soient A, B tels que $B \leq_c A$. Montrer que les assertions suivantes sont équivalentes. On propose l'ordre $1 \Rightarrow 2$ puis $2 \Rightarrow 3$ puis $3 \Rightarrow 4$ puis $4 \Rightarrow 1$.

1. Tout ensemble $C \subseteq A$ tel que $C \leq_c A$ vérifie $B \subseteq C$.
2. Aucun C strictement inclus dans B n'est co-initial dans A.
3. $B = \min(A)$.
4. B est une antichaîne.

Montrer que si B vérifie ces assertions, il est unique.

Soit E, I des ensembles non vides. On va considérer une famille $(R_i)_{i \in I}$ telle que $\forall i \in I, R_i \subseteq E$, et l'ensemble $R_I := \{R_i \mid i \in I\}$ qu'on appelle une collection. Le pré-dual de la collection R_I est défini comme l'ensemble des parties minimales (pour l'inclusion) de E qui intersectent tous les R_i.

Question 3. Dans cette question uniquement, on suppose que E est fini. Décrire dans les grandes lignes, à mi-chemin entre la langue naturelle et le langage algorithmique, un algorithme qui prenne en entrée une collection R_I et retourne en sortie son prédual C_J.

Soient deux collections R_I et C_J. Le couple (R_I, C_J) est dit faiblement déterminé, noté $\mathsf{FDet}(R_I, C_J)$, s'il vérifie l'assertion suivante : $\forall A, B \subseteq E, (A \cup B = E \Rightarrow P)$, où P est une notation pour $(\exists i \in I, R_i \subseteq A) \vee (\exists j \in J, C_j \subseteq B)$.

Question 4. Soient R_I et C_J deux collections. Montrer que les trois assertions suivantes sont équivalentes. Par exemple en montrant 1⇒2, puis 2⇒3, puis 3⇒1.

1. $\mathsf{FDet}(R_I, C_J)$
2. Pour tout $A, B \subseteq E$ tels que $A \cup B = E$ et $A \cap B = \emptyset$, on a P.
3. $\forall A \subseteq E, (\forall j \in J, A \cap C_j \neq \emptyset) \Rightarrow \exists i \in I, R_i \subseteq A$.

On dit que R_I et C_J s'intersectent complètement, noté $\mathsf{Inter}(R_I, C_J)$, si $\forall (i, j) \in I \times J, R_i \cap C_j \neq \emptyset$. On écrit $\mathsf{Inter}(R_I, K)$ au lieu de $\mathsf{Inter}(R_I, \{K\})$. Si $\mathsf{Inter}(R_I, C_J)$ et $\mathsf{FDet}(R_I, C_J)$, on dit que le couple (R_I, C_J) est déterminé, noté $\mathsf{Det}(R_I, C_J)$.

Question 5. Soient R_I et C_J deux collections. Montrer que les assertions suivantes sont équivalentes.

1. $\mathsf{Det}(R_I, C_J)$.
2. C_J est co-initial pour l'ordre $\subseteq$ parmi les parties de E intersectant tous les R_i.
3. R_I est co-initial pour l'ordre $\subseteq$ parmi les parties de E intersectant tous les C_j.

Soient R_I et C_J deux collections. On dit que R_I effleure C_J, dénoté $\mathsf{Ef}(R_I, C_J)$, si $\forall j \in J, \forall x \in C_j, \exists i \in I, R_i \cap C_j = \{x\}$. On écrit $\mathsf{Ef}(R_I, K)$ au lieu de $\mathsf{Ef}(R_I, \{K\})$.

Question 6. Soient R_I une collection. Les assertions suivantes sont-elles équivalentes ?

1. K est minimal parmi les ensembles intersectant tous les R_i.
2. $\mathsf{Ef}(R_I, K)$ et $\mathsf{Inter}(R_I, K)$.

Soit R_I une collection. Si le pré-dual C_J de R_I vérifie $\mathsf{FDet}(R_I, C_J)$, on dit que C_J est le dual de R_I.

Question 7. Montrer que les assertions suivantes sont équivalentes.

1. $\mathsf{Det}(R_I, C_J)$ et pour toute collection C'_L, si $\mathsf{Det}(R_I, C'_L)$ alors $C_J \subseteq C'_L$.
2. $\mathsf{Det}(R_I, C_J)$ et pour tout $L \subsetneq J$, on a $\neg\mathsf{Det}(R_I, C_L)$.
3. $\mathsf{Det}(R_I, C_J)$ et C_J est le pré-dual de R_I.
4. $\mathsf{Det}(R_I, C_J)$ et C_J est une antichaîne
5. $\mathsf{Det}(R_I, C_J)$ et $\mathsf{Ef}(R_I, C_J)$.
6. C_J est le dual de R_I.

Question 8.

1. Soit $I = \{1, \ldots, k\}$ un ensemble fini. Toute collection R_I a-t-elle un dual ?
2. Toute collection de parties finies de E a-t-elle un dual ?

Monoïdes et langages

Un monoïde est un triplet $(M, \cdot_M, e_M)$ où $e_M \in M$ et $\cdot_M : (M \times M) \to M$ vérifie $\forall x \in M, x \cdot_M e_M = e_M \cdot_M x = x$ et $\forall x, y, z \in M, (x \cdot_M y) \cdot_M z = x \cdot_M (y \cdot_M z)$. (Informellement, un monoïde est un groupe sans garantie d'existence d'inverses.) On abrège souvent $x \cdot_M y$ en xy et $(xy)z$ ou $x(yz)$ en xyz. Fixons un monoïde $(M, \cdot_M, e_M)$ pour l'ensemble du sujet, avec M fini.

Question 1. Soit $x \in M$.

1. Montrer que $j_x := \min E$, où $E := \{j \in \mathbb{N} \mid \exists i \in [0, j-1], x^i = x^j\}$, est bien défini.
2. Montrer que les éléments $1, x, \ldots, x^{j_x - 1}$ sont distincts deux à deux.
3. Montrer qu'il existe un unique $i_x \in [0, j_x - 1]$ tel que $x^{i_x} = x^{j_x}$.
4. Soit $p_x := j_x - i_x$. Montrer que pour tout $n \in \mathbb{N}$, $x^{i_x+n} = x^{i_x+r}$, où $r := (n \mod p_x)$ est le reste de la division euclidienne de n par p_x.

Question 2. Qn ne demande pas de justification pour cette question. Soit $x \in M$. Trouver $e_x \in M$ tel que $(G_x, \cdot_M, e_x)$ soit un groupe, où $G_x := \{x^{i_x}, \ldots, x^{i_x+p_x-1}\}$. On ne demande pas de vérifier l'existence d'inverses.

Un groupe $(G, \cdot_G, e_G)$ est dit contenu dans $(M, \cdot_M, e_M)$ si $G \subseteq M$ et $\forall x, y \in G, x \cdot_G y = x \cdot_M y$. On n'exige pas $e_G = e_M$, donc pour tout $x \in M$, $(\{x\}, \cdot_x, x)$ est un groupe, où $\cdot_x$ est une restriction de $\cdot_M$. Un monoïde est dit apériodique s'il ne contient que des groupes réduits à un élément.

Question 3. On rappelle que M est fini. Montrer que les trois assertions suivantes sont équivalentes, par exemple en montrant $1 \Rightarrow 2$ puis $2 \Rightarrow 3$ puis $3 \Rightarrow 1$.

1. M est un monoïde apériodique.
2. Pour tout $x \in M$, on a $p_x = 1$ (et donc $x^{i_x+1} = x^{i_x}$)
3. $\exists k \in \mathbb{N}, \forall x \in M, x^{k+1} = x^k$

On suppose dans la suite que $(M, \cdot_M, e_M)$ est apériodique et fini.

Question 4 (Lemme de simplification). Soient $p, m, q \in M$.

1. Montrer que si $m = pmq$, alors $m = pm = mq$.
2. En déduire que si $pq = e_M$, alors $p = q = e_M$.

Question 5 (Caractérisation de l'égalité). Soit $m, x \in M$. Montrer que $m = x$ ssi $x \in (mM \cap Mm)$ et $m \in MxM$. (Où $mM := \{my \mid y \in M\}$ et $MxM := \{yxz \mid y, z \in M\}$.)

Fixons un ensemble fini non vide Σ. Soit Σ^* l'ensemble des mots finis sur Σ, et ε le mot vide. Soit $\mu : \Sigma^* \to M$ un morphisme, i.e. $\mu(\varepsilon) = e_M$ et $\mu(uv) = \mu(u) \cdot_M \mu(v)$ pour tout $u, v \in \Sigma^*$, où uv est la concaténation des mots u et v.

Question 6. Soient $m \in M$ et $w \in \Sigma^*$. Montrer que les deux assertions suivantes sont équivalentes :

1. $m \notin M\mu(w)M$

2. w se décompose soit en $w = uav$ où $m \notin M\mu(a)M$, soit en $w = uavbt$ où $m \in M\mu(av)M \cap M\mu(vb)M$ et $m \notin M\mu(avb)M$. (Où $u, v, t \in \Sigma^*$ et $a, b \in \Sigma$.)

Question 7. Soient $m \in M \setminus \{e_M\}$ et $w \in \Sigma^*$. Montrer que les deux assertions suivantes sont équivalentes :

1. $\mu(w) \in mM$
2. Il existe $u, v \in \Sigma^*$ et $a \in \Sigma$ tels que $w = uav$ et $\mu(u) \notin mM$ et $\mu(ua)M \subseteq mM$.

Question 8. Soit $m \in M \setminus \{e_M\}$. Montrer que $\mu^{-1}(m) = (U\Sigma^* \cap \Sigma^* V) \setminus (\Sigma^* Y \Sigma^*)$, où $\mu^{-1}(m) := \{w \in \Sigma^* \mid \mu(w) = m\}$ et

$$U := \bigcup_{\substack{a \in \Sigma \\ x \in M \setminus mM \\ x\mu(a)M \subseteq mM}} \mu^{-1}(x)a \qquad V := \bigcup_{\substack{a \in \Sigma \\ x \in M \setminus Mm \\ M\mu(a)x}} a\mu^{-1}(x) \subseteq Mm$$

$$Y := \{a \in \Sigma \mid m \notin M\mu(a)M\} \cup Z \qquad Z := \bigcup_{\substack{a,b \in \Sigma \\ m \in M\mu(a)xM \cap Mx\mu(b)M \\ m \notin M\mu(a)x\mu(b)M}} a\mu^{-1}(x)b$$

Correspondance de Galois

On considère l'ensemble des intervalles fermés sur les entiers relatifs, où $\perp$ représente l'intervalle vide.

$$\mathcal{I} = \{[a, b] \mid a \in \mathbb{Z} \cup \{-\infty\}, b \in \mathbb{Z} \cup \{+\infty\}, a \leq b\} \cup \perp$$

L'ordre sur les entiers relatifs est étendu sur $\mathbb{Z} \cup \{+\infty, -\infty\}$ via :

$$\forall n \in \mathbb{Z}, -\infty < n < +\infty \qquad -\infty < +\infty$$

Question 1. Décrire l'ordre partiel $\sqsubseteq$ sur $\mathcal{I}$ tel que $i_1 \sqsubseteq i_2$ si et seulement si i_1 est inclus dans i_2. Justifier. Pourquoi l'ordre n'est-il pas total ?

Soient $(C, \leq)$ et $(A, \sqsubseteq)$ deux ensembles partiellements ordonnés. La paire $(\alpha : C \to A, \gamma : A \to C)$ est dite une correspondance de Galois si :

$$\forall a \in A, \forall c \in C, c \leq \gamma(a) \Leftrightarrow \alpha(c) \sqsubseteq a$$

On note dans ce cas $(C, \leq) \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} (A, \sqsubseteq)$.
On dit qu'un élément $a \in A$ est une abstraction sûre de $c \in C$ si $c \leq \gamma(a)$.
On dit que $\bar{f} : A \to A$ est une abstraction sûre de $f : C \to C$ si $\forall a \in A, f(\gamma(a)) \leq \gamma(\bar{f}(a))$.
On considère $\gamma_{\mathcal{I}} : \mathcal{I} \to \mathcal{P}(\mathbb{Z})$ telle que

$$\gamma_{\mathcal{I}}(\perp) = \emptyset$$
$$\gamma_{\mathcal{I}}([a, b]) = \{z \in \mathbb{Z} \cup \{-\infty, +\infty\} \mid a \leq z \leq b\}$$

Question 2. Établir $\alpha_{\mathcal{I}}$ tel que $(\mathcal{P}(\mathbb{Z}), \subseteq) \underset{\alpha_{\mathcal{I}}}{\overset{\gamma_{\mathcal{I}}}{\leftrightarrows}} (\mathcal{I}, \sqsubseteq)$.

Question 3. Donner deux abstractions sûres de $\{1, 3, 5\}$ dans les intervalles. Quelle abstraction vous semble la plus précise ?

Question 4. Montrer que $(C, \leq) \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} (A, \sqsubseteq)$ si et seulement si les conditions suivantes sont satisfaites :

1. $\forall c \in C, c \leq \gamma(\alpha(c))$
2. $\forall a \in A, \alpha(\gamma(a)) \sqsubseteq a$
3. γ est croissante
4. α est croissante

Question 5. Calculer les meilleures abstractions sûres de

— $f(X \in \mathcal{P}(\mathbb{Z})) = \{2x \mid x \in X\}$,
— $g(X \in \mathcal{P}(\mathbb{Z})) = \{x \in X \mid x \leq 1\}$.

Question 6. Soient $\bar{f}_1, \bar{f}_2 : A \to A$ des abstractions sûres de $f_1, f_2 : C \to C$. Sous quelle condition sur f_1 est-ce que la composition des abstractions $\bar{f}_1 \circ \bar{f}_2$ est une abstraction sûre de $f_1 \circ f_2$?

Question 7. La composition de deux meilleures abstractions est-elle une meilleure abstraction ?

On considère le programme Python suivant :

```
x = randrange(0, 10) # génère un entier entre 0 inclus et 10 exclus
y = 2 * x
if(y < 6): print(x, y)
```

On cherche à caractériser intuitivement l'ensemble des états de programmes possibles lors de l'affichage, i.e., l'ensemble des valeurs possibles pour toutes les variables. On pose $\mathcal{V} = \{x, y\}$.

Question 8. Caractériser l'ensemble $\sigma \in \mathcal{P}(\mathcal{V} \to \mathbb{Z})$ d'états atteignables avant la conditionnelle, puis lors de l'affichage. Si l'on utilise maintenant des intervalles, quel est l'état atteignable avant la conditionnelle ? Et lors de l'affichage ? Quelles sont les sources d'imprécision ?

Question 9. Établir une correspondance de Galois entre les deux types d'états de programmes mentionnés dans la question précédente.

Réductibilité pour la Terminaison de λ_{ST}

Le λ-calcul, langage formel modélisant la programmation fonctionnelle, est défini par la syntaxe :

$$M, N ::= x \mid M\ N \mid \lambda x.M$$

où x est issu d'un ensemble infini de *variables de termes*.
Ainsi, un terme M est :

- soit une *variable* x,
- soit une *abstraction* $\lambda x.M$ (qu'il faut comprendre comme la fonction qui au paramètre x associe le terme M), on dit dans ce cas que la variable x est *liée* dans M,
- soit une *application* $M\ N$ (qu'il faut comprendre comme le terme - fonction - M appliqué au terme - argument - N)

On note $M[N/x]$ le terme obtenu en remplaçant toutes les occurences (non-liées) de la variable x dans M par le terme N.
Une relation d'α-*conversion* $\equiv_\alpha$ autorise le renommage des variables liées : si $y \notin M$, alors $\lambda x.M \equiv_\alpha \lambda y.(M[x/y])$. Dans la suite, on considèrera les termes *modulo* α-*conversion* (on s'autorisera à renommer les variables liées dans les sous-termes), et on utilisera cette opération pour présenter des termes dans lesquels les variables liées sont distinctes deux à deux, et distinctes des variables non-liées.
La sémantique (le comportement d'un terme) est donnée par une relation de réduction $\longrightarrow$ donnée par :

1. $(\lambda x.M)\ N \longrightarrow M[N/x]$ (on applique la fonction qui à x associe M à N, et on récupère le corps de la fonction M dans lequel l'argument N remplace le paramètre x.)
2. si $M \longrightarrow M'$ alors $M\ N \longrightarrow M'\ N$ (on peut réduire du côté « fonction » d'une application)
3. si $N \longrightarrow N'$ alors $M\ N \longrightarrow M\ N'$ (on peut réduire du côté « argument » d'une applicaton)
4. si $M \longrightarrow M'$ alors $\lambda x.M \longrightarrow \lambda x.M'$ (on peut réduire dans une abstraction).

L'application est parenthésée à gauche, ainsi $M_1\ M_2\ M_3$ désigne $(M_1\ M_2)\ M_3$ et on écrit $\lambda xy.M$ pour $\lambda x.(\lambda y.M)$
$\longrightarrow^*$ désigne la clôture réflexive et transitive de $\longrightarrow$. Les *réduits* d'un terme M sont les éléments de l'ensemble $\{M' \mid M \longrightarrow^* M'\}$. Un *graphe de réduction* d'un terme M est un graphe avec boucles où les sommets sont les réduits de M et les arêtes sont les couples (M', M'') tels que $M' \longrightarrow M''$.
On définit $I = \lambda x.x$, $K = \lambda xy.x$, et $\Omega = (\lambda x.(x\ x))\ (\lambda y.(y\ y))$.
À titre d'exemple, on donne le graphe de réduction de $(\lambda xy.(x\ z))\ I\ I$ dans la Figure 1.

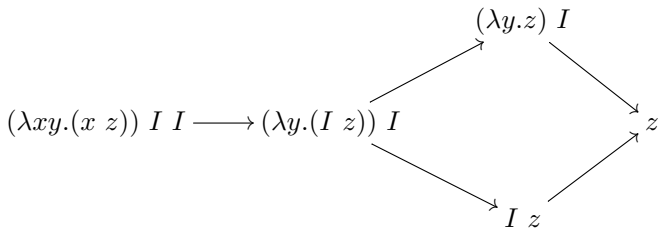


FIGURE 1 – Graphe de réduction de $(\lambda xy.(x\ z))\ I\ I$

Question 1. Donner les graphes de réduction des termes suivants :

1. $(I\ I)\ (I\ I)$
2. $K\ I\ \Omega$

Question 2. Donner un terme dont le graphe de réduction est infini.

On donne un système de *types simples* au λ-calcul. Les types sont donnés par :

$$S, T ::= \tau \mid S \to T$$

Ainsi un type est soit une variable τ (issue d'un ensemble infini de variables de types, disjoint de l'ensemble des variables de termes) soit le type fonctionnel $S \to T$ des fonctions de S dans T.
Une *hypothèse* $x : T$ associe une variable de terme à un type. Un *contexte* Γ est un ensemble d'hypothèses et on note $\Gamma, x : T$ le contexte $\Gamma \cup \{x : T\}$ quand x n'est pas dans Γ. Un *jugement* de typage $\Gamma \vdash M : T$ indique que le terme M a le type T dans le contexte Γ (on dit qu'un terme est *typable* s'il existe Γ, T tel que $\Gamma \vdash M : T$). Les règles de typage permettant de déduire un jugement sont données par :

1. $\Gamma, x : T \vdash x : T$
2. si $\Gamma, x : T_1 \vdash M : T_2$ alors $\Gamma \vdash \lambda x.M : T_1 \to T_2$
3. si $\Gamma \vdash M : T_1 \to T_2$ et $\Gamma \vdash N : T_1$, alors $\Gamma \vdash M\ N : T_2$

Question 3. Montrer que $\emptyset \vdash I : \tau \to \tau$.

Puis montrer que K, puis $K\ I\ I$ sont typables.

Question 4. Montrer que Ω n'est pas typable.

Question 5. Montrer la propriété de *substitution assujettie* :

Pour tout M, N, Γ, T, T_0, x, si $\Gamma, x : T_0 \vdash M : T$ et $\Gamma \vdash N : T_0$, alors $\Gamma \vdash M[N/x] : T$

Question 6. Montrer la propriété de *réduction assujettie* :

Pour tout M, M', Γ, T, si $\Gamma \vdash M : T$ et $M \longrightarrow M'$, alors $\Gamma \vdash M' : T$

Un terme M est *terminant* quand il n'existe pas de suite $(M_n)_{n\in\mathbb{N}}$ telle que
$M_0 = M$ et $\forall n \in \mathbb{N}, M_n \longrightarrow M_{n+1}$.

On cherche à montrer que tous les termes typables sont terminants.

Question 7. Montrer qu'il existe un terme terminant qui n'est pas typable.

On définit $\texttt{RED}_T$, les *termes réductibles de type* T, par induction sur T :

— $\texttt{RED}_\tau$ est l'ensemble des termes terminants de type τ.

— $\texttt{RED}_{T_1 \to T_2}$ est l'ensemble des termes M de type $T_1 \to T_2$ tels que pour tout $N \in \texttt{RED}_{T_1}$, $M\ N \in \texttt{RED}_{T_2}$.

Question 8. Montrer que :

1. si $M \in \texttt{RED}_T$, alors M est terminant.
2. si $M \in \texttt{RED}_T$ et $M \longrightarrow M'$, alors $M' \in \texttt{RED}_T$
3. si $\Gamma \vdash M : T$ et M n'est pas une abstraction, et si $\forall M', M \longrightarrow M' \Rightarrow M' \in \texttt{RED}_T$, alors $M \in \texttt{RED}_T$

Question 9. Montrer que si pour tout $N \in \texttt{RED}_{T_1}$ et $M[N/x] \in \texttt{RED}_{T_2}$, alors $\lambda x.M \in \texttt{RED}_{T_1 \to T_2}$

Question 10. Montrer que si M est typable, M est terminant.

Énumération rapide pour les équations booléennes

On considère dans cet exercice un système de m équations booléennes sur n variables $\mathbf{x} = (x_0, \ldots, x_{n-1}) \in (\mathbb{Z}/2\mathbb{Z})^n$, de la forme : $\forall 1 \leq k \leq m, f_k(\mathbf{x}) = 0$. Chaque $f_k : (\mathbb{Z}/2\mathbb{Z})^n \to \mathbb{Z}/2\mathbb{Z}$ est une fonction booléenne *quadratique*, c'est-à-dire un polynôme de degré 2 :

$$f_k(\mathbf{x}) = f_k(0) + \sum_{i=0}^{n-1}\sum_{j=0}^{n-1} a_{ij}x_i x_j \ ,$$

où les a_{ij} sont des coefficients booléens, et l'addition et le produit se font dans $\mathbb{Z}/2\mathbb{Z}$. Pour deux vecteurs $\mathbf{x}, \mathbf{y} \in (\mathbb{Z}/2\mathbb{Z})^n$, on note $\mathbf{x} + \mathbf{y}$ leur addition bit à bit dans $(\mathbb{Z}/2\mathbb{Z})^n$.
Notons que le monôme $x_i x_i$ est équivalent à x_i puisque pour tout Booléen $x^2 = x$. On suppose que les équations sont aléatoires et indépendantes : chaque a_{ij} vaut 1 avec probabilité $\frac{1}{2}$. De plus, on pose $m \leq n$ afin de garantir l'existence (en moyenne) de solutions. On veut énumérer toutes les solutions du système.
On mesure la complexité des algorithmes en opérations binaires (additions et multiplications dans $\mathbb{Z}/2\mathbb{Z}$), ainsi que d'autres opérations de base qui seront détaillées par la suite. Pour tout $0 \leq j \leq n-1$, on note $\mathbf{e}_j^n \in (\mathbb{Z}/2\mathbb{Z})^n$ le vecteur booléen $(0, \ldots, 0, 1, 0, \ldots, 0)$, ne comportant qu'un 1 en position j. On pourra simplifier cette notation en $\mathbf{e}_j$ lorsque n est fixé et connu.
Pour tout entier $0 < \ell \leq 2^n - 1$, on écrit ℓ en base 2 **en plaçant le bit de poids faible à gauche**. On note alors $\nu(\ell)$ la position du premier bit à 1 dans cette écriture binaire, en partant de 0 ; et $\nu'(\ell)$ la position du deuxième bit à 1. Par exemple :

$$3 \text{ s'écrit } 11 \implies \nu(3) = 0, \nu'(3) = 1$$

$$26 \text{ s'écrit } 01011 \implies \nu(26) = 1, \nu'(26) = 3$$

Notons que $\nu(0)$ n'est pas défini, de même que $\nu'(\ell)$ si ℓ est une puissance de 2 (et n'a donc qu'un seul bit à 1).

Question 1. Soit un système de $m = n$ équations à n variables. Montrer qu'on peut énumérer (exactement) toutes ses solutions en $O(n^3 2^n)$ opérations.

Le code de Gray à n bits est une liste de 2^n vecteurs générée récursivement de la manière suivante :

- Pour $n = 1$: renvoyer la liste à deux éléments $\big[(0), (1)\big]$
- Pour $n > 1$: soit $\big[y_0, \ldots, y_{2^{n-1}-1}\big]$ le code de Gray à $n-1$ bits. Renvoyer la liste de vecteurs :

$$\big[y_0 \| 0, \ \ y_1 \| 0, \ \ \ldots, \ \ y_{2^{n-1}-1} \| 0, \ \ y_{2^{n-1}-1} \| 1, \ \ \ldots, \ \ y_0 \| 1\big]$$

  où $\|$ est une concaténation (par exemple $(0,1,1)\|0 = (0,1,1,0)$)

Lorsque n est fixé, on note $\mathsf{Gray}(n, t)$ le t-ème élément de la liste, compté à partir de 0.

Question 2.
1. Donner le code de Gray pour $n = 3$
2. Montrer que pour tout n, le code de Gray à n bits contient chaque vecteur de $(\mathbb{Z}/2\mathbb{Z})^n$ exactement une fois
3. Montrer que pour tout $1 \leq t < 2^n$, $\mathsf{Gray}(n, t) = \mathsf{Gray}(n, t-1) + \mathbf{e}^n_{\nu(t)}$

Indice : on utilisera que $\nu(t) = v$ si et seulement si t s'écrit sous la forme $k2^v$ avec k impair.

Si $f(\mathbf{x})$ est une fonction quadratique booléenne en n variables, sa *dérivée en* x_i, notée $\frac{\partial f}{\partial i}$, est la fonction :

$$\frac{\partial f}{\partial i} : \mathbf{x} \mapsto f(\mathbf{x} + \mathbf{e}_i^n) + f(\mathbf{x})$$

On suppose dans la suite que calculer ν, ν' ou incrémenter un compteur coûte une seule opération.

Question 3. 1. Montrer que $\frac{\partial f}{\partial i}$ est une fonction affine sur $(\mathbb{Z}/2\mathbb{Z})^n$.

2. En déduire un algorithme qui énumère tous les zéros de f en $O(n2^n)$ opérations au lieu de $O(n^2 2^n)$.
   Indice : on utilisera un code de Gray pour énumérer les vecteurs de $(\mathbb{Z}/2\mathbb{Z})^n$.

Question 4. Soit $t \geq 1$ une entrée du code de Gray. Soit s le plus petit entier strictement supérieur à t tel que $\nu(s) = \nu(t)$. Montrer que :

$$\mathsf{Gray}(n, t) + \mathsf{Gray}(n, s-1) = e^n_{\nu'(s)}$$

Comment interpréter ce résultat ?

Indice : on utilisera que si t s'écrit sous la forme $k2^v$ avec k impair, alors $\nu'(t) = \nu(t - 2^v)$.

Question 5. En déduire une modification de l'algorithme précédent pour énumérer tous les zéros de f en $O(2^n)$ opérations.

Question 6. En utilisant la question précédente, donner un algorithme pour résoudre un système de $m = n$ équations en n variables, qui utilise $O((\log n)2^n)$ opérations en moyenne.

A Annexe : sujets proposés pour la filière MPI

Certains sujets sont accompagnés d'*ébauches* de solution.

Inférence de Types

Soit $\mathcal{A} = \{A, B, C, \dots\}$ un ensemble infini de *variables de types*.
On considère $\mathcal{T}$ l'ensemble des *types* définis inductivement par :

- $\mathbb{N} \in \mathcal{T}$. ($\mathbb{N}$ est un type)
- $\mathcal{A} \subseteq \mathcal{T}$. (une variable de types est un type)
- si $T_1 \in \mathcal{T}$ et $T_2 \in \mathcal{T}$ alors $T_1 \to T_2 \in \mathcal{T}$. (la "flèche" de deux types est un type)

Une *substitution de types* $\sigma : \mathcal{A} \longrightarrow \mathcal{T}$ est une fonction qui vaut l'identité presque partout, c'est-à-dire telle que $\{A \in \mathcal{A} \mid \sigma(A) \neq A\}$ est fini.
Si σ est une substitution de types et $T \in \mathcal{T}$ un type, on note $\sigma_\uparrow(T)$ (par abus de notation, on s'autorisera à écrire $\sigma(T)$) le type obtenu inductivement par :

- $\sigma_\uparrow(\mathbb{N}) = \mathbb{N}$.
- pour $\mathcal{A}$, $\sigma_\uparrow(A) = \sigma(A)$.
- pour tout $T_1, T_2 \in \mathcal{T}^2$, $\sigma_\uparrow(T_1 \to T_2) = \sigma_\uparrow(T_1) \to \sigma_\uparrow(T_2)$

Un *système d'équations de types* (ou juste *système*) est un ensemble fini $\{(T_k, T'_k)\}_{1 \leq k \leq n}$ de couples de types.
Un *unificateur* d'un système $\{(T_k, T'_k)\}_{1 \leq k \leq n}$ est une substitution de types σ telle que $\forall k, \sigma_\uparrow(T_k) = \sigma_\uparrow(T'_k)$.

Par exemple, on pose $S_0 = \{(B \to A, C),\ (\mathbb{N}, B)\}$

et σ_0 définie par $\begin{cases} \sigma_0(B) & = & \mathbb{N} \\ \sigma_0(C) & = & \mathbb{N} \to A \\ \sigma_0(X) & = & X \text{ pour tout} X \notin \{B, C\} \end{cases}$

Alors σ_0 est un unificateur de S_0 car :

1. $\sigma_0(B \to A) = \mathbb{N} \to A$ et $\sigma_0(C) = \mathbb{N} \to A$
2. $\sigma_0(\mathbb{N}) = \mathbb{N}$ et $\sigma_0(B) = \mathbb{N}$

Question 1. Trouver si possible un unificateur des systèmes suivants :

1. $\{(\mathbb{N} \to A, B \to \mathbb{N})\}$
2. $\{(\mathbb{N} \to A, B \to (C \to \mathbb{N})),\ (\mathbb{N} \to \mathbb{N}, C)\}$
3. $\{(A \to B), (B \to A)\}$
4. $\{(\mathbb{N} \to (A \to \mathbb{N}), \mathbb{N} \to B),\ (B, A)\}$

Question 2. Un système S admet-il toujours un unificateur ?

Quand un unificateur pour S est-il unique ?

Question 3. En considérant les différentes possibilités pour les formes des types T_1 et T'_1, exprimer l'existence d'un unificateur pour $\{(T_k, T'_k)\}_{1 \leq k \leq n}$ en fonction de l'existence d'un unificateur pour un autre système, bien choisi.

Question 4. Rédiger l'algorithme induit par la question précédente, étudier sa terminaison.

Soit $\mathcal{X} = \{x, y, z, \dots\}$ un ensemble de *variables de termes*.
On considère un langage d'expressions fonctionnelles $\mathcal{E}$ défini inductivement par :

- $\mathbb{N} \subseteq \mathcal{E}$, (les entiers sont des expressions)
- $\mathcal{X} \subseteq \mathcal{E}$ (les variables de termes sont des expressions)
- Si $(E_1, E_2) \in \mathcal{E}^2$, $E_1 + E_2 \in \mathcal{E}$ (la somme de deux expressions est une expression)
- Si $(E_1, E_2) \in \mathcal{E}^2$, $E_1(E_2) \in \mathcal{E}$ (l'application d'une expression à une expression est une expression)
- Si $x \in \mathcal{X}$ et $E \in \mathcal{E}$ alors $(x \mapsto E) \in \mathcal{E}$ (si E est une expression et x une variable, l'expression fonctionnelle $x \mapsto E$ est une expression)

On supposera que les variables x apparaissant dans des sous-expressions $x \mapsto E'$ d'une expression E sont toutes distinctes et distinctes deux à deux des variables de E qui n'apparaissent jamais directement à gauche d'un $\mapsto$. On note $FV(E)$ ce dernier ensemble de variables.

Par exemple, $f_0 = (x \mapsto (y \mapsto x(y) + 1))$ est un élément de $\mathcal{E}$.

Les contextes de types sont des fonctions de $D \subseteq \mathcal{X}$ dans T.
On note $\emptyset$ le contexte de domaine vide et $(\Gamma, x : T)$, le contexte Γ' défini par $\Gamma'(x) = T$ et pour tout y dans le domaine de Γ, $y \neq x \Rightarrow \Gamma'(y) = \Gamma(y)$.

Dans la suite on s'intéresse au problème de donner un type de $\mathcal{T}$ à une expression de $E \in \mathcal{E}$ avec un contexte Γ qui sert d'oracle donnant le type des variables de $FV(E)$, quand c'est possible. On dit qu'on *infère un type de E dans le contexte Γ*.

Question 5. Donner deux types différents qu'il semble légitime de donner à f_0 dans le contexte $\emptyset$.

Question 6. Donner des règles qui formalisent quand il est légitime de donner un type T à E dans le contexte Γ.

Question 7. En déduire un algorithme qui infère un type d'une expression E.

Question 8. Utiliser cet algorithme pour trouver un type de $\mathbf{S} = x \mapsto (\ y \mapsto (\ z \mapsto x(z)(y(z))\ )\ )$ dans le contexte $\emptyset$.

Zones

Soit $\mathcal{V} = \{V_1, \ldots, V_n\}$ un ensemble de variables, et $\mathbb{I} \in \{\mathbb{Z}, \mathbb{Q}, \mathbb{R}\}$.
Une contrainte de potentiel est une inégalité de la forme $V_i - V_j \leq c$, avec $c \in \mathbb{I}$.
Une conjonction de contraintes de potentiel sur $\mathcal{V}$ est représentée par une matrice M de taille $n \times n$ telle que :

- $M_{ij} = c$ si la contrainte $V_j - V_i \leq c$ est présente dans la conjonction
- $M_{ij} = +\infty$ sinon

Dans la suite, on appelle ces matrices des matrices de potentiel.
Les solutions des contraintes de potentiel définies par une matrice M sont :

$$\textsc{sols}(M) = \{(v_1, \ldots, v_n) \in \mathbb{I}^n | v_j - v_i \leq M_{i,j}\}$$

Question 1. Soit M une matrice de potentiel de taille n, et $(v_1, \ldots, v_n) \in \textsc{sols}(M)$ une solution à une conjonction de contraintes de potentiel. Montrer que $\forall c \in \mathbb{I}$, $(v_1 + c, \ldots, v_n + c) \in \textsc{sols}(M)$.

Solution : Les $+c$ vont s'annuler dans toutes les inégalités.

Question 2. Expliquer comment encoder des contraintes de la forme $V_i \leq c$ et $V_i \geq c$ en étendant $\mathcal{V}$.

Solution : Ajouter une variable $V_0 \in \mathcal{V}$. On a ensuite $V_i \leq c \rightsquigarrow V_i - V_0 \leq c$, et $V_i \geq c \Leftrightarrow -V_i \leq -c \rightsquigarrow V_0 - V_i \leq -c$.
Il faut ensuite définir une variante des solutions sur ces contraintes étendues.

Question 3. On suppose que $M_{i_1,i_2} = c_1, \ldots, M_{i_{n-1},i_n} = c_n$. Peut-on en déduire une nouvelle contrainte ?

Solution : On a donc $V_{i_2} - V_{i_1} \leq c_1, \ldots, V_{i_n} - V_{i_{n-1}} \leq c_n$ On obtient $V_{i_n} - V_{i_1} \leq c_1 + \ldots + c_n$.

Question 4. On considère les contraintes suivantes :

$$1 \leq V_1; V_1 \leq 5; 1 \leq V_2; V_2 \leq 3; V_1 - V_2 \leq 1$$

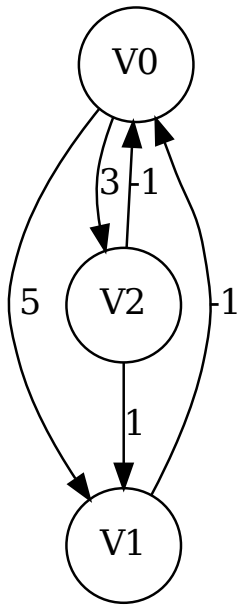
1. Donner la représentation matricielle de cet ensemble de contraintes.
2. Montrer qu'il est possible de déduire une contrainte sur V_1 qui est plus précise que les contraintes initiales.

Solution : On encode en :

$$\begin{aligned} V_0 - V_1 &\leq -1 \\ V_1 - V_0 &\leq 5 \\ V_0 - V_2 &\leq -1 \\ V_2 - V_0 &\leq 3 \\ V_1 - V_2 &\leq 1 \end{aligned}$$

La matrice est donc : $\begin{pmatrix} +\infty & 5 & 3 \\ -1 & +\infty & +\infty \\ -1 & 1 & +\infty \end{pmatrix}$

Pour la question suivante, il est important de noter que la matrice représente un graphe orienté pondéré



Oui, on peut sommer $V_2 - V_0 \leq 3$ et $V_1 - V_2 \leq 1$ pour obtenir $V_1 - V_0 \leq 4$. Graphiquement, cela correspond à trouver le chemin de plus faible poids entre V_0 et V_1

> Une matrice de potentiel est dite close s'il n'est pas possible de déduire de contraintes plus précises. La clôture d'une matrice M, est la matrice M^* telle que $\text{SOLS}(M) = \text{SOLS}(M^*)$ et M^* est close. On admet que lorsque le système de contraintes est satisfiable, M^* existe et est unique.

Question 5. On suppose que le système de contraintes est satisfiable. Donner un algorithme calculant M^*. Quelle est sa complexité ?

Solution : Il faut donc utiliser l'algorithme de Floyd Warshall (au programme de MPI), dans une version en place ici. Il n'y a pas de problèmes de cycles de poids négatif grâce à l'hypothèse de cette question.

```
1: fonction FW(M  matrice de taille n × n)
2:     pour k = 1 à n faire
3:         pour i = 1 à n faire
4:             pour j = 1 à n faire
5:                 M_{i,j} = min(M_{i,j}, M_{i,k} + M_{k,j})
6:             fin pour
7:         fin pour
8:     fin pour
9:     renvoyer M
10: fin fonction
```

On peut parler de complexité temporelle (cubique) et spatiale (ici, on a calculé en place la transformation).

Question 6. Montrer que le système de contraintes est insatisfiable si et seulement si $\exists i, M^*_{i,i} < 0$.

Solution : On montre que le système de contraintes est insatisfiable ssi le graphe possède un cycle de poids strictement négatif, par double implication. Étant donné une matrice de potentiel M, on note

$$\textsc{sols}(M) = \{(v_0, \ldots, v_n) \in \mathcal{I}^{n+1} | v_j - v_i \leq M_{i,j}\}$$

- $\Leftarrow$ Supposons que G possède un cycle de poids strictement négatif. On note $v_{i_1}, \ldots, v_{i_l}$ les arêtes de ce cycle. Les poids correspondent à $M_{i_2,i_1}, ..., M_{i_l,i_1}$, et leur somme, notée c, est strictement négative. Par l'absurde, supposons qu'il existe $(v_0, \ldots, v_n) \in \textsc{sols}(M)$. Par définition, $v_{i_2} - v_{i_1} \leq M_{i_2,i_1}$, ..., $v_{i_1} - v_{i_l} \leq M_{i_l,i_1}$. En sommant, on obtient $0 \leq c$, mais $c < 0$, contradiction.
- $\Rightarrow$ Par contrapposée, supposons qu'il n'existe pas de cycle de poids strictement négatif. On note $G = (V, E)$ le graphe initial, $G' = (V', E')$, avec $V' = V \sqcup \{x\}, E' = E \cup \{(x, v, 0) | v \in E\}$. On note s_i le plus court chemin de x à v_i, qui est bien défini car G n'a pas de cycles de poids négatif. Par définition, $s_j \leq s_i + M_{j,i}$. Donc $s_j - s_i \leq M_{j,i}$. Ainsi le point $(s_0, \ldots, s_n)$ est une solution des contraintes de potentiel.

Question 7. Comment adapter l'algorithme de la question 5 à une implémentation en langage C sur des entiers ?

Solution : Il faut réfléchir à deux choses :

- La représentation de $+\infty$ comme poids des arêtes non connectées (enum ou `MAX_INT`).
- Les cycles de poids négatif peuvent créer des entiers négatifs qui grandissent très vite. Il vaut mieux changer l'algorithme pour que si $i = j$ et $M_{i,i} < 0$ alors l'algorithme s'arrête, vu qu'il n'y a de toute façon pas unicité de la clôture dans ce cas.

  Il est possible d'exhiber des matrices d'adjacence dont les coefficients de la matrice d'adjacence lors des différentes itérations de l'algorithme croissent très rapidement. On peut par exemple prendre la matrice $M_{i,j} = -1$.

  On considère l'algorithme de Floyd Warshall qui n'est pas en place, définit sous la forme de récurrence suivante :

$$M^{(0)} = M$$
$$M^{(n+1)}_{i,j} = \min(M^{(n)}_{i,j}, M^{(n)}_{i,n} + M^{(n)}_{n,j})$$

  On montre par récurrence que $M^{(k)}_{i,j} = -2^k$.

Question 8. Comment transformer les contraintes de la forme $\varepsilon_i V_i + \varepsilon_j V_j \leq c$, avec $\varepsilon_i, \varepsilon_j \in \{-1, 1\}$ en contraintes de potentiel ?

Solution : Cette fois-ci, on duplique les variables, (V_i^+ représente V_i et V_i^- représente $-V_i$)

- $V_i - V_j \leq c \rightsquigarrow V_i^+ - V_j^+ \leq c \wedge V_j^- - V_i^- \leq c$
- $V_i + V_j \leq c \rightsquigarrow V_i^+ - V_j^- \leq c \wedge V_j^+ - V_i^- \leq c$
- $-V_i - V_j \leq c \rightsquigarrow V_i^- - V_j^+ \leq c \wedge V_j^- - V_i^+ \leq c$
- $V_i \leq c \rightsquigarrow V_i^+ - V_i^- \leq 2c$
- $V_i \geq c \rightsquigarrow V_i^- - V_i^+ \leq -2c$

$\textsc{SolsOct}(M) = \{(v_1, \ldots, v_n) \in \mathbb{I}^n | (v_1, -v_1, \ldots, v_n, -v_n) \in \textsc{Sols}(M)\}$

Elimination des coupures dans MLL et MALL

La logique linéaire additive-multiplicative $MALL$ a pour syntaxe :
$A, B ::= p \mid p^\perp \mid A \otimes B \mid A \parr B \mid A \oplus B \mid A \& B$
avec p appartenant à un ensemble infini de *formules atomiques*.
L'opération de *dualité* $\cdot^\perp$ est définie inductivement sur les formules :
$(A \otimes B)^\perp = A^\perp \parr B^\perp \qquad (A \oplus B)^\perp = A^\perp \& B^\perp$
$(A \parr B)^\perp = A^\perp \otimes B^\perp \qquad (A \& B)^\perp = A^\perp \oplus B^\perp \qquad (p^\perp)^\perp = p$
On note Γ, Δ des multi-ensembles $A_1, \ldots, A_n$ de formules (c'est-à-dire que les A_i ne sont pas ordonnées, mais on tient compte de leur multiplicité). Un *séquent linéaire* (ou seulement *séquent*, dans ce sujet) $\vdash \Gamma$ est prouvable quand on peut le déduire des règles suivantes (c'est-à-dire construire avec ces règles une *preuve* dont $\vdash \Gamma$ est la conclusion) :

$$\frac{}{\vdash A, A^\perp}(\texttt{Ax}) \qquad \frac{\vdash \Gamma, A^\perp \qquad \vdash A, \Delta}{\vdash \Gamma, \Delta}(\texttt{Cut}) \qquad \frac{\vdash \Gamma, A \qquad \vdash B, \Delta}{\vdash \Gamma, A \otimes B, \Delta}(\otimes) \qquad \frac{\vdash \Gamma, A, B}{\vdash \Gamma, A \parr B}(\parr)$$

$$\frac{\vdash \Gamma, A \qquad \vdash \Gamma, B}{\vdash \Gamma, A \& B}(\&) \qquad \frac{\vdash \Gamma, A}{\vdash \Gamma, A \oplus B}(\oplus_1) \qquad \frac{\vdash \Gamma, B}{\vdash \Gamma, A \oplus B}(\oplus_2)$$

Une occurence de (`Cut`) dans une preuve qui ajoute les formules A et $A^\perp$ dans ses séquents prémisses est appelée une *coupure sur* A.

Usuellement on appelle "tenseur" le connecteur $\otimes$, "par" le $\parr$, "plus" le $\oplus$ et "avec" le $\&$.
On définit la formule $A \multimap B$ comme $A^\perp \parr B$.

Question 1. Montrer que $\vdash A \otimes (B \oplus C) \multimap (A \otimes B) \oplus (A \otimes C)$ est prouvable

Question 2. Montrer qu'il n'existe pas de preuve de $\vdash A \multimap (A \otimes A)$ sans coupure.

Question 3. Informellement, on interprète la formule A comme "un exemplaire de la ressource A".
Donner une interprétation informelle des symboles $\cdot^\perp$, $\otimes$, $\oplus$, $\&$, $\multimap$.

On dit qu'une règle *élimine* la formule A, si A est une formule qui apparaît dans le séquent conclusion de la règle mais dans aucun des séquents prémisses. Par exemple, la règle $(\otimes)$ élimine la formule $A \otimes B$.
Dans une preuve, on dit qu'une coupure sur la formule C est *principale* si $C^\perp$ et C sont toutes les deux éliminées par (respectivement) la première règle appliqué à la prémisse gauche et la première règle appliquée à la prémisse droite de la coupure.

Question 4. Soit Π une preuve de $\vdash \Gamma_0$ contenant une coupure principale sur la formule C.
Montrer qu'on peut supprimer la coupure sur C de Π, c'est-à-dire obtenir une preuve de $\vdash \Gamma_0$ sans coupure sur C.

Question 5. Soit Π une preuve de $\vdash \Gamma_0$ contenant une coupure non-principale sur la formule C. Montrer qu'il existe une preuve Π' de $\vdash \Gamma_0$ sans coupure sur C.

Le théorème d'*élimination des coupures* dans un système de preuve (un ensemble de règles) $\mathcal{L}$ contenant (`Cut`) est donné par :
Tout séquent $\vdash \Gamma$ *prouvable dans* $\mathcal{L}$ *est prouvable dans* $\mathcal{L}$ *privé de la règle* (`Cut`)

Question 6. Les transformations induites par les questions 4. et 5. forment-elle un bon algorithme pour prouver l'élimination des coupures dans MALL ?

> MLL est la logique obtenue à partir de MALL en n'utilisant que les formules atomiques, la dualité $\cdot^{\perp}$ et les connecteurs $\otimes$ et $⅋$, et seulement les règles (`Ax`), (`Cut`), ($\otimes$) et ($⅋$).
> Un *réseau de MLL* (ou simplment *réseau*) est un graphe orienté avec arêtes pendantes (des arêtes sans destination dont la source est un sommet du graphe), dont les sommets (appelés *noeuds*) sont étiquetés par (`Ax`), (`Cut`), ($⅋$), ($\otimes$). Chaque noeud étiqueté par ($\otimes$) ou ($⅋$) a autant d'arêtes entrantes que le nombre de prémisses de la règle associée à son étiquette. Les noeuds étiquetés par (`Ax`) n'ont pas d'arêtes entrante et deux arêtes sortantes. Les noeuds étiquetés par (`Cut`) n'ont pas d'arête sortante et deux arêtes entrantes.

Question 7. Donner un algorithme qui prend une preuve Π de MLL et la transforme en un réseau correspondant.

> Un *réseau de preuve* est un réseau dans l'image de l'algorithme précédent (obtenu à partir d'une preuve d'un séquent de MLL).
> Un *interrupteur* d'un réseau $\mathcal{R}$ est un graphe non-orienté obtenu en supprimant les arêtes pendantes de $\mathcal{R}$, en supprimant, pour chaque noeud étiqueté par $⅋$ dans $\mathcal{R}$, une des deux arêtes entrantes dans ce noeud, et en désorientant les arêtes restantes.
> On admet le *théorème de Danos-Régnier* :
> *Un réseau $\mathcal{R}$ est un réseau de preuve si et seulement si tout interrupteur de $\mathcal{R}$ est connexe et acyclique.*

Question 8.

1. Donner deux preuves différentes d'un même séquent linéaire envoyées par l'algorithme de la question 7. sur le même réseau.
2. Donner un réseau qui n'est pas un réseau de preuve.
3. Donner un algorithme prend un réseau de preuve obtenu à partir d'une preuve prouvant le séquent $\vdash \Gamma$ et renvoie une preuve de $\vdash \Gamma$.

Question 9. Montrer le théorème d'élimination des coupures dans MLL.

Question 10. Peut-on appliquer cette technique à MALL ?

Structures pliables et traversables

On se place dans un langage récursif, fonctionnel, avec des définitions de types inductifs et un système de types polymorphes similaire à OCaml. On pourra utiliser indifféremment du pseudocode fonctionnel ou la syntaxe OCaml.
On suppose l'existence :

- d'une fonction identité `id` de type $A \to A$;
- d'un opérateur $(\circ) : (A \to B) \to (B \to C) \to (A \to C)$ de composition de fonctions, noté $\circ$ dans sa version infixe ; et
- de types de bases usuels, comme `int` le type des entiers et `unit` le type de l'unique élément ().

On donne la définition d'un type polymorphe `liste` A (où A est une variable de type) et, à titre d'exemple, de la fonction longueur de type `liste` $A \to$ `int` en pseudocode (par exemple) :

`liste` $A =$ `Nil` | `Cons` A (`liste` A)
`longueur` : `liste` $A \to$ `int`
  `longueur` `Nil` $= 0$
  `longueur` (`Cons` _ qu) $= 1 +$ (`longueur` qu)

---

et en OCaml :

```
type 'a liste = Nil | Cons of ('a * 'a liste)
let rec longueur : 'a liste -> int = function
        Nil -> 0
      | Cons (_,qu) -> 1 + (longueur qu)
```

Un type T est un *monoïde*, s'il existe (c'est-à-dire, si on peut définir) ε de type T et $(\diamond)$ de type $T \to T \to T$, noté $\diamond$ dans sa version infixe, tels que pour tous a, b, c de type T, $(a \diamond b) \diamond c = a \diamond (b \diamond c)$ et $\varepsilon \diamond a = a \diamond \varepsilon = a$.

Question 1. Montrer que `liste` A est un monoïde pour tout A.

Un *constructeur de type* est une fonction mathématique C qui associe à un type paramètre T un type résultat $C\ T$. Par exemple `liste` est un constructeur de type.
On dit qu'un constructeur de type C est *pliable* quand il existe un `fold` de type

$$(A \to B \to B) \to B \to C\ A \to B$$

pour tous types A et B, tel que `fold` $f\ e\ t$ applique de manière répétée la fonction f à chacun des éléments de type A de la structure t (elle-même de type CA) pour produire une valeur de type B, en partant initialement de l'élément e de type B.

Question 2. Montrer que `liste` est pliable.

Un `fold` pour `liste` est-il unique ?

Question 3.

1. Donner la définition d'un constructeur de type `barbre` tel que les éléments de `barbre` A sont des arbres binaires dont les nœuds sont des éléments de A.
2. Montrer que `barbre` est pliable.
3. Expliquer comment différentes définitions de `fold` permettent de générer les parcours infixe et préfixe d'un arbre.

Question 4. Utiliser `fold` pour écrire `arbreTaille` : `int` $\to$ `liste` (`barbre` `unit`), la fonction générant une liste des arbres binaires de taille n (c'est-à-dire, à n nœuds) avec les nœuds prenant leur valeur dans `unit`.

Une définition alternative d'un constructeur de type pliable est l'existence d'un `foldmap` de type

$$(A \to M) \to C\ A \to M$$

pour tout type A et pour tout monoïde M, tel que `foldmap` f t parcourt la structure t en accumulant une valeur de type M.

Question 5. Montrer que `liste` et `barbre` sont pliables pour cette définition.

Donner deux définitions de `foldmap` pour les `barbre` permettant de générer les parcours infixes et préfixes.

Question 6. Montrer que les deux définitions de pliabilité sont équivalentes.

Un *foncteur applicatif* est un constructeur de type F pour lequel il existe pour tous types A, B :

1. `fmap` de type $(A \to B) \to F\ A \to F\ B$
2. `pure` de type $A \to F\ A$
3. $(\odot)$ de type $F\ (A \to B) \to F\ A \to F\ B$, noté $\odot$ dans sa version infixe.

qui vérifie (entre autres) les lois suivantes :
`fmap id` $=$ `id` `fmap` $(f \circ g) =$ (`fmap` f) $\circ$ (`fmap` g)
(`pure id`) $\odot\ v = v$

Question 7. Sans vérifier formellement les lois, donner deux manières différentes de montrer que `liste` est un foncteur applicatif.

Un constructeur de types C est *traversable* s'il existe `traverse` de type

$$(A \to F\ B) \to C\ A \to F\ (C\ B)$$

pour tout foncteur applicatif F et types A et B.

Question 8. Montrer que `barbre` est traversable.

Question 9. On dispose d'un arbre t de type `barbre` A et une fonction `futurs` qui à un élément A associe un une liste de type `liste` A qui correspond aux « futurs possibles » (dans un sens non-déterministe) d'un élément.

A quoi correspond `traverse futurs` t ?

Graphes parfaits

Pour S un ensemble, on note $\mathcal{P}_2(S) = \{\{x, y\} : x, y \in S, x \neq y\}$ les sous-ensembles de S de cardinalité 2. Dans tout le sujet, on considère des graphes non-orientés $G = (V, E)$, avec un ensemble fini de sommets V, et un ensemble d'arêtes $E \subseteq \mathcal{P}_2(V)$.
Coloriage. Un coloriage d'un graphe $G = (V, E)$ est une application $V \to \mathbb{N}$. Dans ce contexte, on appelle les entiers des *couleurs*. Un coloriage est *valide* si toute paire de sommets reliés par une même arête a des couleurs différentes. Le *nombre chromatique* de G, noté $\chi(G)$, est le nombre minimal de couleurs nécessaires pour créer un coloriage valide de G.
Sous-graphe induit. Étant donné un graphe $G = (V, E)$ et un sous-ensemble de sommets $W \subseteq V$, le *sous-graphe induit* par W est le graphe $G[W] = (W, E \cap \mathcal{P}_2(W))$. On dit que c'est un sous-graphe induit *propre* si W est inclus strictement dans V.
Cliques. Une *clique* d'un graphe $G = (V, E)$ est un sous-ensemble de sommets $W \subseteq V$ tel que le sous-graphe $G[W]$ induit par W est un graphe complet, c'est-à-dire : $G[W] = (W, \mathcal{P}_2(W))$. On note $\omega(G)$ la cardinalité de la plus grande clique de G.
Anticliques. Une *anticlique* d'un graphe $G = (V, E)$ est un sous-ensemble de sommets $W \subseteq V$ tel que le sous-graphe $G[W]$ induit par W ne contient pas d'arête, c'est-à-dire : $G[W] = (W, \varnothing)$. On note $\alpha(G)$ la cardinalité de la plus grande anticlique de G.

Question 1. Soit G un graphe quelconque. Montrer $\chi(G) \geq \omega(G)$.

Question 2. Soit $G = (V, E)$ un graphe quelconque.

a. Montrer qu'un coloriage valide de G avec c couleurs existe si et seulement si il existe une partition $\{A_1, \ldots, A_c\}$ de V en c anticliques.

b. Montrer $\chi(G)\alpha(G) \geq |V|$.

Graphe parfait. Un graphe G est dit *parfait* si tous ses sous-graphes induits $G[W]$ satisfont : $\chi(G[W]) = \omega(G[W])$.
Graphe imparfait minimal. Un graphe G est dit *imparfait minimal* s'il n'est pas parfait, et que tous ses sous-graphes induits propres sont parfaits.

Question 3. Donner un exemple de graphe parfait, et de graphe imparfait minimal.

Pour simplifier les notations, dans les questions 4 à 8, on fixe G un graphe imparfait minimal quelconque. Sans perte de généralité, on pose $V = \{1, \ldots, n\}$. On note $\alpha = \alpha(G)$, $\omega = \omega(G)$, $\chi = \chi(G)$.

Question 4. Soit A une anticlique de G. Montrer $\omega(G[V \setminus A]) = \omega$.

Question 5. Soit A_0 une anticlique de G de cardinalité α. Montrer qu'il existe $\alpha\omega$ anticliques $A_1, \ldots, A_{\alpha\omega}$, telles que pour chaque sommet $v \in V$, v fait partie d'exactement α anticliques parmi $A_0, \ldots, A_{\alpha\omega}$. (Formellement : $\forall v \in V, |\{i \in \{0, \ldots, \alpha\omega\} : v \in A_i\}| = \alpha$.)

Question 6. On considère une suite d'anticliques $A_0, \ldots, A_{\alpha\omega}$ définie comme dans la question précédente. Montrer que pour tout i dans $\{0, \ldots, \alpha\omega\}$, il existe une clique C_i telle que $C_i \cap A_i = \varnothing$, et $\forall j \neq i, |C_i \cap A_j| = 1$.
Indication : utiliser la question 4

Matrice d'incidence. Étant donné une suite $V_0, \ldots, V_{\alpha\omega}$ de sous-ensembles de $V = \{1, \ldots, n\}$, on définit la *matrice d'incidence* de la suite (V_i) comme la matrice $M = (M_{i,j})_{1 \leq i \leq n, 0 \leq j \leq \alpha\omega} \in \{0,1\}^{n \times (\alpha\omega+1)}$ définie par $M_{i,j} = 1$ si $i \in V_j$, 0 sinon.

Question 7. Soit M_A la matrice d'incidence de la suite $A_0, \ldots, A_{\alpha\omega}$ de la question 5, et M_C la matrice d'incidence de la suite $C_0, \ldots, C_{\alpha\omega}$ de la question 6. On note $M_A^\intercal$ la transposée de M_A.

Montrer que $M_A^\intercal M_C$ est de rang $\alpha\omega + 1$, où les matrices sont vues comme à coefficient dans $\mathbb{Q}$.

Question 8. Montrer $n \geq \alpha\omega + 1$.

Dans les questions suivantes, $G = (V, E)$ est un graphe quelconque.

Question 9. Montrer que G est parfait si et seulement si $\omega(G[W])\alpha(G[W]) \geq |W|$ pour tout sous-graphe induit $G[W]$ de G.

Question 10. Soit $\overline{G} = (V, \mathcal{P}_2(V) \setminus E)$ le graphe *complémentaire* de G. Montrer que G est parfait si et seulement si $\overline{G}$ est parfait.

Théorème des amis

L'objectif du sujet est de montrer que dans tout groupe de personnes, si chaque paire de personnes a exactement un ami en commun, alors il existe quelqu'un (le diplomate) qui est ami de tout le monde. Le problème est modélisé par des graphes.

Graphe. Pour S un ensemble, on note $\mathcal{P}_2(S) = \{\{x, y\} \subseteq S : x \neq y\}$ l'ensemble des paires d'éléments distincts de S. Dans tout le sujet, on considère des graphes non-orientés $G = (V, E)$, composés d'un ensemble fini de sommets V, et d'un ensemble d'arêtes $E \subseteq \mathcal{P}_2(S)$.

Voisins. Étant donné un graphe $G = (V, E)$ et un sommet $x \in V$, l'ensemble des *voisins* de x est $N(x) = \{y \in V : \{x, y\} \in E\}$. Le *degré* de x est le nombre de voisins de x.

Graphe d'amis. Un *graphe d'amis* est un graphe $G = (V, E)$ tel que $|V| \geq 2$ et pour tout $x, y \in V$ avec $x \neq y$, il existe un unique sommet, noté $x \star y$, tel que : $\{x, x \star y\} \in E$ et $\{y, x \star y\} \in E$.

Diplomate. Étant donné un graphe $G = (V, E)$, un *diplomate* est un sommet de degré $|V| - 1$.

Exemple. Le graphe complet à trois sommets $G = (\{1, 2, 3\}, \{\{1, 2\}, \{2, 3\}, \{1, 3\}\})$ est un graphe d'amis contenant 3 diplomates.

Question 1.

a. Montrer qu'un graphe d'amis $G = (V, E)$ ne contient pas de 4-cycle, c'est-à-dire :

$$\neg\exists\{a, b, c, d\} \subseteq V, \{\{a, b\}, \{b, c\}, \{c, d\}, \{d, a\}\} \in E.$$

b. Donner un exemple de graphe d'amis à 5 sommets.

Question 2. Dans cette question, on suppose que G est un graphe d'amis contenant un sommet x de degré 2. On note y, z les deux voisins de x.

a. Montrer $\{y, z\} \in E$.

b. Montrer $V = N(y) \cup N(z)$.

c. Montrer que y ou z est un diplomate.

Dans les questions 3 à 5, on fixe $G = (V, E)$ un graphe d'amis ne contenant pas de sommet de degré 2. (On suppose donc pour l'instant qu'un tel graphe existe ; on verra plus tard si cela amène une contradiction.) Soit $n = |V|$. Sans perte de généralité, on pose $V = \{1, \ldots, n\}$.

Question 3. Soit $\{x, y\}$ une arête de G et $z = x \star y$. Soit $X = N(x) \setminus \{y, z\}$, $Y = N(y) \setminus \{x, z\}$, $Z = N(z) \setminus \{x, y\}$, et $W = V \setminus (X \cup Y \cup Z \cup \{x, y, z\})$.

a. Montrer que l'application suivante est bien définie et bijective :

$$\begin{aligned} f_\star : X \times Y &\to W \\ (a, b) &\mapsto a \star b. \end{aligned}$$

b. Montrer que tous les sommets de G on le même degré (on dit que G est *régulier*).

c. On note δ le degré d'un sommet de G. Montrer $n = \delta^2 - \delta + 1$.

Matrice d'adjacence. La *matrice d'adjacence* de G est la matrice $M_{i,j \in \{1,\ldots,n\}} \in \mathbb{R}^{n \times n}$ définie par $M_{i,j} = 1$ si $\{i, j\} \in E$, 0 sinon.

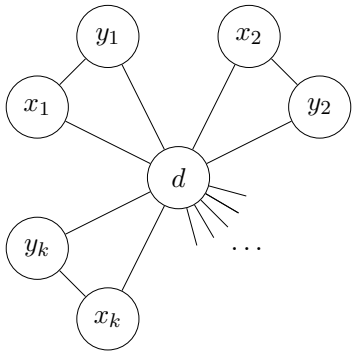
Question 4. Soit M la matrice d'adjacence du graphe G. On a vu dans la question 3 que G est régulier ; et on note δ le degré d'un sommet de G. Montrer $M^2 = \mathbb{1}_n + (\delta - 1)\mathsf{Id}_n$, où $\mathbb{1}_n \in \mathbb{R}^{n\times n}$ est la matrice ne contenant que des 1, et $\mathsf{Id}_n \in \mathbb{R}^{n\times n}$ est la matrice identité.

On admet (ou rappelle) les faits suivants.

- Toute matrice symétrique réelle est diagonalisable.
- La trace d'une matrice est invariante par changement de base.
- Si $k \in \mathbb{N}$ et $\sqrt{k} \in \mathbb{Q}$, alors $\sqrt{k} \in \mathbb{N}$ (lemme de Dirichlet).
- Les valeurs propres de $M^2 = \mathbb{1}_n + (\delta - 1)\mathsf{Id}_n$ sont δ^2 (multiplicité 1) et $\delta - 1$ (multiplicité $n - 1$).

Question 5. Montrer que la trace de M doit être nulle, et aboutir à une contradiction.

Question 6. Montrer que pour tout graphe d'amis $G = (V, E)$, il existe un diplomate $d \in V$, et une partition de $V \setminus \{d\}$ en paires $\{\{x_1, y_1\}, \ldots, \{x_k, y_k\}\}$, telle que $E = \bigcup_{i=1}^{k} \{\{x_i, y_i\}, \{d, x_i\}, \{d, y_i\}\}$.



Graphe moulin

Raisonnements ensemblistes

L'ensemble des entiers naturels est noté $\mathbb{N}$, et l'ensemble des parties de $\mathbb{N}$ est noté $\mathcal{P}(\mathbb{N})$.
Soit $\mathcal{T}$ un ensemble fini dont les éléments sont appelés des *variables ensemblistes*. Une *inclusion* est une formule s'écrivant : $(X \subseteq Y)$ où X (resp. Y) est une variable ensembliste, ou $\emptyset$, ou $\mathbb{N}$. Par abus de notation, $\emptyset$ et $\mathbb{N}$ représentent ici respectivement les ensembles vide et plein.
Attention : dans la suite, par convention, les lettres $A, B, C, D \ldots$ désigneront des éléments de $\mathcal{T}$, tandis que les lettres X, Y, Z désigneront des éléments de $\mathcal{T} \cup \{\emptyset, \mathbb{N}\}$.
Une conjonction d'inclusions sur les variables de $\mathcal{T}$ est dite *satisfiable* s'il existe une *valuation* $f : \mathcal{T} \to \mathcal{P}(\mathbb{N})$ des variables qui vérifie toutes les inclusions.
Pour deux formules Φ et Ψ, $\Phi \vDash \Psi$ signifie que toute valuation satisfaisant Φ satisfait également Ψ.

Question 1. 1. Montrer que la conjonction d'inclusions : $(\mathbb{N} \subseteq A_1) \wedge (A_2 \subseteq A_3) \wedge (A_3 \subseteq \emptyset)$ est satisfiable.
2. Montrer qu'en ajoutant l'inclusion : $\mathbb{N} \subseteq A_2$, ce n'est plus satisfiable.

Solution : Prendre $A_1 = \mathbb{N}$, $A_2 = A_3 = \emptyset$. Dans le deuxième cas on a une contradiction entre $A_3 = \mathbb{N}$ et $A_3 = \emptyset$.

Question 2. Soit une conjonction d'inclusions ne contenant aucune inclusion de la forme $\mathbb{N} \subseteq X$. Montrer que cette conjonction est satisfiable.

Solution : Il suffit d'évaluer toutes les variables à $\emptyset$.

Soit une conjonction d'inclusions $\Phi = (X_1 \subseteq Y_1) \wedge \ldots \wedge (X_k \subseteq Y_k)$. On définit un graphe orienté G_Φ tel que :

- Les sommets de G_Φ sont les éléments de $\mathcal{T} \cup \{\emptyset, \mathbb{N}\}$
- Pour toute inclusion $X_i \subseteq Y_i$ de Φ, on crée une arête $X_i \to Y_i$ dans G_Φ
- Pour tout sommet $A \neq \mathbb{N}$, on crée une arête $A \to \mathbb{N}$, et pour tout sommet $A \neq \emptyset$, une arête $\emptyset \to A$

Un *chemin* dans G_Φ est une séquence de sommets $Z_1, \ldots, Z_\ell$ reliés par des arêtes $Z_1 \to Z_2, Z_2 \to Z_3, \ldots, Z_{\ell-1} \to Z_\ell$. Par convention, on autorise des chemins ne comportant qu'un seul sommet (et aucune arête).

Question 3. Montrer l'équivalence entre les propriétés suivantes :

1. Φ est satisfiable
2. Il n'existe pas de chemin dans G_Φ menant de $\mathbb{N}$ à $\emptyset$
3. Φ est satisfiable à valeurs dans $\{\emptyset, \mathbb{N}\}$

Solution : Pour $3 \implies 1$: si Φ est satisfiable à valeurs dans $\{\emptyset, \mathbb{N}\}$, en particulier elle est satisfiable.

Pour $1 \implies 2$: tout chemin $Z_1 \to Z_2 \to \ldots \to Z_k$ dans G se traduit en une chaîne d'inclusions $Z_1 \subseteq \ldots \subseteq Z_k$. Par conséquent, s'il existe un chemin de $\mathbb{N}$ à $\emptyset$, par transitivité de l'inclusion (et récurrence immédiate sur la longueur du chemin) Φ implique $\mathbb{N} \subseteq \emptyset$ ce qui est une contradiction, et elle n'est pas satisfiable.

Pour $2 \implies 3$: Pour un sommet X de G_Φ, soit $s(X)$ l'ensemble des sommets Y tels que $X \to^* Y$ (successeurs, par clôture transitive), et $p(X)$ l'ensemble des sommets Y tels que $X \to^* Y$ par des chemins dans le graphe (prédécesseurs).

Supposons qu'il n'existe pas de chemin de $\mathbb{N}$ à $\emptyset$. On montre qu'il existe une valuation qui fonctionne, qui sera à valeurs dans $\{\emptyset, \mathbb{N}\}$. On propose d'évaluer à $\mathbb{N}$ pour $V_N = s(\mathbb{N})$ et $\emptyset$ pour $V_0 = p(\emptyset)$. Par hypothèse V_0 et V_N sont disjoints. Quant aux sommets restants, on les évalue tous à $\emptyset$. Pour vérifier que cette valuation fonctionne, il faut vérifier qu'elle satisfait toutes les inclusions de Φ.

Soit $X \subseteq Y$ une inclusion dans Φ. On étudie les cas possibles : si $X \in V_N$, alors $Y \in V_N$ et l'inclusion est satisfaite. Si $Y \in V_0$, alors $X \in V_0$ et de même. Si $X \in V_0$, alors $X \subseteq D$ sera toujours satisfait. Si $Y \in V_N$, de même. Reste le cas où : $X \notin (V_0 \cup V_N)$ et $Y \notin (V_0 \cup V_N)$. Comme on a évalué ces deux variables à $\emptyset$ cette inclusion est satisfaite.

Question 4. Montrer que si Φ est satisfiable et X et Y sont des sommets de G_Φ, alors $\Phi \vDash (X \subseteq Y)$ si et seulement s'il existe un chemin dans G_Φ menant de X à Y.

Solution : Une des deux implications est facile : s'il existe un chemin de X à Y, on a $X \subseteq Y$.

Supposons qu'il n'existe pas de chemin de X à Y (en particulier $X \neq Y$, car $X \to^* X$ est aussi un chemin). Par hypothèse Φ est satisfiable, donc il n'y a pas de chemin de $\mathbb{N}$ à $\emptyset$. Cette fois, on va trouver une valuation de Φ qui ne satisfait pas $X \subseteq Y$.

L'idée est d'assigner $\emptyset$ à Y et $\mathbb{N}$ à X, et de voir si on peut encore satisfaire toutes les inclusions de Φ. (Notons qu'on ne peut pas avoir $Y = \mathbb{N}$ ni $X = \emptyset$ ici puisque cela contredirait l'hypothèse selon laquelle il n'y a aucun chemin). On propose la valuation suivante : $\mathbb{N}$ pour $V_N = s(X)$ (qui contient aussi les successeurs de $\mathbb{N}$ puisque $\mathbb{N}$ est successeur de X) ; $\emptyset$ pour $V_0 = p(Y)$ (qui contient aussi les prédécesseurs de $\emptyset$) ; $\emptyset$ pour les sommets restants. Par hypothèse V_0 et V_N sont disjoints, sinon on aurait un chemin de X à Y. On réutilise le raisonnement ci-dessus pour montrer que toutes les inclusions de Φ sont satisfaites.

On peut remarquer que par exemple une formule Φ force $\emptyset \subseteq A$ pour toute variable A, même si A n'apparaît pas dans Φ.

Question 5. En déduire un algorithme Résolution qui prend en entrée une conjonction d'inclusions Φ et une inclusion $(X \subseteq Y)$, et détermine si $\Phi \vDash (X \subseteq Y)$. Montrer qu'avec la bonne structure de données, cet algorithme est de complexité linéaire en le nombre d'inclusions de Φ et de variables de $\mathcal{T}$.

Solution : Il faut d'abord déterminer si Φ est satisfiable (dans ce cas on renvoie Vrai tout le temps).

Ensuite, il faut déterminer s'il existe un chemin entre X et Y dans le graphe G_Φ. On fait simplement un parcours en profondeur à partir de X (il n'est pas demandé de le détailler, puisqu'ils sont censés le connaître). Quand le graphe est représenté comme une liste d'adjacence, la complexité est linéaire en le nombre d'arêtes. Ce nombre d'arêtes dépend du nombre d'inclusions dans Φ et de variables dans $\mathcal{T}$.

On considère maintenant des formules logiques dont les littéraux sont des inclusions, par exemple :

$$(X_1 \subseteq X_2) \vee \neg(X_2 \subseteq X_3) \vee \neg(X_4 \subseteq X_3)$$

Ces formules sont écrites en forme normale conjonctive. On suppose que toutes les clauses sont *Horn*, c'est-à-dire qu'elles contiennent au plus un seul littéral positif (sans négation). On les interprète alors comme des implications. La formule ci-dessus devient ainsi :

$$\Big((X_2 \subseteq X_3) \wedge (X_4 \subseteq X_3)\Big) \implies (X_1 \subseteq X_2)$$

On appelle une telle formule *inclusion-Horn* et on lui étend la notion de satisfiabilité ci-dessus.

Question 6. Montrer que la formule inclusion-Horn avec les clauses suivantes :

$$\neg(A_3 \subseteq \emptyset)$$
$$\neg(A_1 \subseteq A_3)$$
$$\neg(A_2 \subseteq A_3) \vee \neg(A_1 \subseteq A_2)$$
$$(\mathbb{N} \subseteq A_1)$$
$$(A_2 \subseteq \emptyset)$$

est satisfiable, mais qu'aucune de ses valuations satisfaisantes n'est à valeurs dans $\{\emptyset, \mathbb{N}\}$.

Solution : On est obligé de prendre $A_1 = \mathbb{N}$, $A_2 = \emptyset$. Ensuite, comme $A_1 \subseteq A_2$ est faux, il ne reste plus que les contraintes : $\neg(A_1 \subseteq A_3) \implies A_3 \neq \mathbb{N}$ et $\neg(A_3 \subseteq \emptyset) \implies A_3 \neq \emptyset$. Pour A_3 on peut prendre tout ensemble non vide qui n'est ni $\emptyset$ ni $\mathbb{N}$.

On admet la propriété (P) :

Soit Ψ une conjonction d'inclusions. Si Ψ est satisfiable, il existe une valuation f telle que pour tous sommets X, Y dans le graphe G_Ψ, s'il n'existe pas de chemin entre X et Y, alors $f(X)$ et $f(Y)$ sont incomparables (i.e., $f(X) \not\subseteq f(Y)$ et $f(Y) \not\subseteq f(X)$).

Question 7. Soit Φ une formule inclusion-Horn, et soit Ψ la conjonction de toutes les clauses de Φ ne contenant aucun littéral négatif. C'est donc une conjonction d'inclusions de la forme : $\Psi = (X_1 \subseteq Y_1) \wedge \ldots \wedge (X_k \subseteq Y_k)$.

Montrer que si Ψ est satisfiable, et si pour tout littéral négatif $\neg(X \subseteq Y)$ dans les clauses de Φ, $\Psi \not\models (X \subseteq Y)$, alors Φ est satisfiable.

Solution : Noter que, puisque le graphe G_Ψ contient toutes les variables de $\mathcal{T}$ (y compris les variables qui ne sont pas dans Ψ), la valuation donne bien des valeurs à toutes ces variables.

Supposons que Ψ est satisfiable. Utilisons la valuation ci-dessus. Pour toutes variables X, Y, si $\Psi \not\models (X \subseteq Y)$ alors il n'existe pas de chemin entre X et Y par la question 4. Donc $f(X)$ et $f(Y)$ sont incomparables, donc le terme $\neg(X \subseteq Y)$ est vrai.

Par conséquent cette valuation rend tous les termes négatifs dans les clauses de Φ vrais. Toutes les clauses sans termes négatifs étant vraies (puisque Ψ l'est), toutes les clauses de Φ sont vraies. Elle est donc satisfiable.

Question 8. En déduire un algorithme déterminant, en temps polynomial, si une formule inclusion-Horn est satisfiable.

Solution : On définit un algorithme récursif.

Soit Φ une formule inclusion-Horn et soit Ψ l'ensemble de ses clauses positives. On effectue une disjonction de cas. On voit avec la question 7 qu'on va avoir par exemple le cas où Ψ est satisfiable et n'implique aucun littéral négatif, dans ce cas on peut s'arrêter et la formule est satisfiable. Sinon il reste d'autres cas :

1. Si Φ contient une clause vide, on renvoie « Non satisfiable »
2. Si Ψ n'est pas satisfiable, on renvoie « Non satisfiable »
3. Sinon, il existe un littéral négatif $\neg(X \subseteq Y)$ dans une des clauses de Φ tel que $\Psi \vDash X \subseteq Y$. Il faut observer que la formule obtenue en enlevant $\neg(X \subseteq Y)$ de la clause correspondante est équivalente à Φ.

Cela nous donne un algorithme récursif qui transforme Φ, Ψ en Φ', Ψ' où Φ' est plus petite que Φ (quand on ne s'arrête pas) d'un littéral négatif. Chaque appel récursif réduit la taille de la formule, donc le temps est polynomial. La correction de l'algorithme découle des questions précédentes. On peut raffiner le temps d'exécution de quadratique à linéaire en utilisant des structures de données adaptées.

Question 9. Prouver la propriété (P).

Solution : La valuation se construit en résolvant d'abord les cycles de G_Ψ, de façon à enlever toutes les inclusions triviales. On enlève aussi les successeurs de $\mathbb{N}$ et les prédécesseurs de $\emptyset$ (dont les valuations sont évidentes).

On fait ensuite un tri topologique sur les sommets restants, de façon à les parcourir dans l'ordre suivant : on doit s'assurer que pour chaque sommet, on a d'abord étudié tous ses prédécesseurs. On démarre de $\emptyset$. Soit $A_0, \ldots, A_k, \ldots$ la séquence des sommets rencontrés, alors l'ensemble $S(A_k)$ correspondant à A_k est calculé de la manière suivante :

$$S(A_k) = \{k\} \cup \bigcup_{X, X \to A_k} S(X)$$

Cette valuation satisfait bien toutes les inclusions, et de plus, elle a la propriété demandée par construction.

Permutations triables par pile

On s'intéresse aux permutations $\mathfrak{P}(n)$ de $\{1,\dots,n\}$. Une permutation $\pi \in \mathfrak{P}(n)$ est assimilée à la suite $(\pi(1),\pi(2),\dots,\pi(n))$.

Machine à pile. Une *machine à pile* maintient en interne une structure de pile, initialement vide. Elle prend en entrée une permutation $(\pi(1),\dots,\pi(n))$, et effectue une suite fixée d'opérations empile et dépile.

- empile : lit le premier élément non encore lu de la permutation, et l'ajoute à la pile. La i-ième opération empile ajoute donc $\pi(i)$ à la pile.
- dépile : enlève le dernier élément ajouté à la pile, et le renvoie en sortie.

La sortie de la machine est une suite d'éléments de $\{1,\dots,n\}$ renvoyés par les opérations dépile, pris dans l'ordre où ils sont renvoyés. *Exemple :* la machine EEDEDD qui effectue (empile, empile, dépile, empile, dépile, dépile) avec en entrée la permutation $(3,1,2)$ renvoie $(1,2,3)$:

$$\mathsf{EEDEDD}(3,1,2) = (1,2,3).$$

On remarque que le comportement d'une machine à pile est entièrement défini par la suite (fixe) d'opérations empile et dépile qu'elle effectue.

Suite valide. Une telle suite d'opérations est dite *valide* si elle contient exactement n opérations empile et n opérations dépile, et si à tout instant, le nombre d'opérations dépile effectuées est inférieur ou égal au nombre d'opérations empile effectuées. Dans tout le sujet, on ne considère que des machines effectuant des suites valides d'opération.

Permutation triable par pile. Une permutation $\pi \in \mathfrak{P}(n)$ est dite *triable par pile* s'il existe une machine à pile qui prend en entrée $(\pi(1),\dots,\pi(n))$, et qui renvoie $(1,\dots,n)$.

Exemple : l'égalité $\mathsf{EEDEDD}(3,1,2) = (1,2,3)$ plus haut montre que $(3,1,2)$ est triable par pile.

Question 1. Montrer que les permutations $(1,2,3,4)$, $(4,3,2,1)$, $(1,3,2,4)$ sont triables par pile.

Motif. On dit que $\pi \in \mathfrak{P}(n)$ *contient le motif* $\rho \in \mathfrak{P}(k)$ pour $k \leq n$ s'il existe $x_1 < \cdots < x_k$ dans $\{1,\dots,n\}$ tel que $(\pi(x_1),\dots,\pi(x_k)) = (x_{\rho(1)},\dots,x_{\rho(k)})$ (ou plus informellement : tel que $(\pi(x_1),\dots,\pi(x_k))$ et $(\rho(1),\dots,\rho(k))$ sont « dans le même ordre »).

Question 2. Montrer que si π est triable par pile, alors elle ne contient pas le motif $(2,3,1)$.

Question 3. Montrer que si π ne contient pas le motif $(2,3,1)$, alors elle est triable par pile.

Question 4. L'ensemble des permutations triables par pile est-il clos par composition ? Par inverse ?

Question 5. Proposer un algorithme qui détermine en temps linéaire en n si une permutation $\pi \in \mathfrak{P}(n)$ est triable par pile.

Indication : on peut utiliser les questions 2 et 3, mais ce n'est pas obligatoire.

Graphe associé à une permutation. À une permutation $\pi \in \mathfrak{P}(n)$, on associe le graphe non-orienté $G(\pi) = (V, E)$ de sommets $V = \{1, \ldots, n\}$, et d'arêtes $E = \{\{a, b\} \in V : a < b \text{ et } \pi(a) > \pi(b)\}$.
Graphe triable par pile. On dit qu'un graphe $G = (V, E)$ avec $V = \{1, \ldots, n\}$ est *triable par pile* s'il existe $\pi \in \mathfrak{P}(n)$ tel que $G = G(\pi)$ et π est triable par pile.
Graphe réalisable. On dit qu'un graphe $G = (V, E)$ est réalisable s'il est possible de renommer ses sommets V avec les entiers $\{1, \ldots, n\}$, de telle sorte que le graphe obtenu est triable par pile.

Question 6. Montrer que $\pi \mapsto G(\pi)$ est une injection de $\mathfrak{P}(n)$ vers l'ensemble des graphes de sommets $\{1, \ldots, n\}$. Est-ce une bijection ?

Question 7. Donner un exemple de graphe qui n'est pas réalisable.

Question 8. On considère l'ensemble des graphes formés en partant du graphe vide $(\varnothing, \varnothing)$, et en utilisant uniquement les deux opérations suivantes : ajout d'un sommet universel à un graphe de l'ensemble (un sommet est dit *universel* s'il est relié à tous les autres sommets), union disjointe de deux graphes de l'ensemble. Montrer que l'ensemble obtenu est exactement l'ensemble des graphes réalisables.

Question 9. En s'inspirant de la question précédente, proposer un algorithme qui détermine si un graphe $G = (V, E)$ est réalisable, en temps $O(|V|^3)$.