Devoir Non Surveillé 3

À rendre le 25 février 2025

INFORMATIQUE MP2I

Tournez la page S.V.P.

1 Un peu de géométrie (types énumérés en OCaml)

On s'intéresse à des ensembles de points du plan $\mathbb{R}^2$. On se donne les types suivants en OCaml pour les représenter :

```
type point = {x : float; y : float}
type vector = {x : float; y : float}
type set =
  | Set of point list (* Ensemble fini de points *)
  | Line of point * point (* Droite, donnee par deux de ses points *)
  | Circle of point * float (* Cercle, donne par son centre et son rayon *)
type cardinal =
  | Finite of int
  | Infinite
```

De plus, comme nous travaillons avec des nombres flottants, nous définissons une fonction qui nous permet de déterminer si deux nombres sont assez proches pour que l'on considère qu'ils soient égaux. On se fixe aussi une précision ε à cet effet.

```
let epsilon = 0.001

let is_eq (f1 : float) (f2 : float) =
  let df = f1 -. f2 in df > -. epsilon && df < epsilon
```

☐ 1 – Écrire une fonction len (l : 'a list) : int qui calcule la taille d'une liste, et en déduire une fonction card (s : set) : cardinal qui calcule le cardinal d'un ensemble de points.

☐ 2 – En supposant que le type int peut représenter tous les entiers sans dépassement de capacité, proposer une relation d'ordre totale $\leq_c$ telle que l'ensemble des éléments de type cardinal munit de $\leq_c$ est un ensemble bien fondé.

On souhaite à présent calculer l'intersection de deux ensembles de points du type set.

☐ 3 – Écrire une fonction vect (a : point) (b : point) : vector qui renvoie le vecteur $\overrightarrow{AB}$, et une fonction det (u : vector) (v : vector) : float qui calcule le déterminant de deux vecteurs $\vec{u}$ et $\vec{v}$, et en déduire une fonction are_aligned (a : point) (b : point) (c : point) : bool qui détermine si trois points A, B, et C sont alignés.

☐ 4 – En déduire une fonction lines_intersect (a, b : point * point) (c, d : point * point) : set qui calcule l'intersection des droites (AB) et (CD). On prendra soin de justifier la correction de la fonction en détaillant les calculs qui y ont mené.

☐ 5 – Écrire une fonction filter (p : 'a -> bool) (l : 'a list) : 'a list qui prend en entrée un prédicat p une liste ℓ et qui renvoie la liste des éléments x de ℓ vérifiant $p(x)$.

☐ 6 – En déduire une fonction on_line (a, b : point * point) (l : point list) : point list qui renvoie la liste des points de ℓ appartenant à la droite (AB),
et une fonction common_elements (l1 : 'a list) (l2 : 'a list) : 'a list qui renvoie la liste des éléments en qui sont à la fois dans ℓ_1 et dans ℓ_2. Quelle est la complexité de common_elements ?

☐ 7 – Écrire une fonction dist (a : point) (b : point) : float qui calcule la distance AB, et en déduire une fonction on_circle (o, r : point * float) (l : point list) : point list qui renvoie la liste des points de ℓ appartenant au cercle de centre O de rayon r.

☐ 8 – Écrire des fonctions pour les cas restants,
et en déduire une fonction intersection (s1 : set) (s2 : set) : set qui calcule $S_1 \cap S_2$. [1]
On prendra soin de justifier la correction de ces fonctions en détaillant les calculs qui y ont mené.

On souhaite à présent étendre notre type set pour inclure les unions de droites, ou les unions de cercles, ou les unions d'unions de droites et d'unions de cercles, etc.
On modifie donc à présent notre déclaration de type :

```
type set =
  | Set of point list (* Ensemble fini de points *)
  | Line of point * point (* Droite, donnee par deux de ses points *)
  | Circle of point * float (* Cercle, donne par son centre et son rayon *)
  | Union of (set * set) (* Union de deux ensembles de points *)
```

☐ 9 – Proposer un ajout à la fonction cardinal pour gérer le nouveau cas.

☐ 10 – Proposer un ajout à la fonction intersection pour gérer le nouveau cas. On pourra utiliser la distributivité de l'intersection par rapport à l'union.

1. Si vous n'arrivez pas à calculer l'intersection dans un ou plusieurs des cas restants, remplacez ce calcul par **assert false**

2 Introduction aux fonctions de hachages

Dans cette partie, on pose $\Sigma = [\![0, 255]\!]$ l'alphabet composé des valeurs de type **char**. On note Σ^* l'ensemble des chaînes de caractères, c'est-à-dire des suites finies d'éléments de Σ, ainsi que ε la chaîne de caractère vide "". Pour $u \in \Sigma^*$ et $a \in \Sigma$, on note $u \cdot a$ le mot composé des lettres de u suivi de la lettre a. On cherche à construire une fonction $h : \Sigma^* \mapsto \Sigma$ et à étudier ses propriétés.

☐ 11 – Dans un premier temps, nous posons h définit comme suit :

$$h(\varepsilon) = 0, \qquad h(u \cdot a) = (128 \times (h(u) + a)) \mod 256$$

Donner les valeurs possibles pour $h(u)$, avec $u \in \Sigma^*$.

On considère à présent la fonction h définit comme suit :

$$h(\varepsilon) = 0, \qquad h(u \cdot a) = (17 \times (h(u) + a)) \mod 256$$

☐ 12 – Écrire une fonction **int** hash(**char** u[]) qui prend en entrée une chaîne de caractères u et qui renvoie $h(u)$.

☐ 13 – Montrer que 17 est inversible dans $\mathbb{Z}/256\mathbb{Z}$ et donner son inverse.

☐ 14 – Montrer que pour tout mot $u \in \Sigma^*$, si a est tiré aléatoire uniformément dans Σ, alors $h(u \cdot a)$ est distribué uniformément dans $[\![0, 255]\!]$.
Reformulation : montrer que pour toute valeur $x \in [\![0, 255]\!]$ et tout mot $u \in \Sigma^*$, il existe un unique a tel que $h(u \cdot a) = x$.

☐ 15 – Donner deux chaînes de caractères u et v distinctes ($u \neq v$) telles que $h(u) = h(v)$.

☐ 16 – Soit $u_0, u_1, \ldots, u_n$ $n + 1$ lettres de Σ. Notons $u = \varepsilon \cdot u_0 \cdot u_1 \cdot \ldots \cdot u_{n-1}$ et $u' = \varepsilon \cdot u_1 \cdot u_2 \cdot \ldots \cdot u_n$. Exprimez $h(u')$ en fonction de $h(u)$, u_0 et u_n.

☐ 17 – On note a^n le mot défini par :

$$a^0 = \varepsilon, \qquad a^{n+1} = a^n \cdot a$$

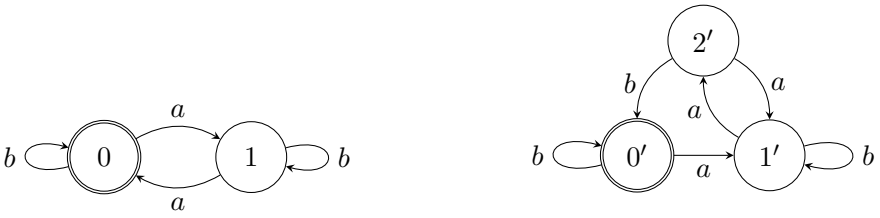
Déterminer en justifiant $\{h(a^n) \mid n \in \mathbb{Z}\}$ pour les valeurs de a suivantes :

1. $a = 0$ (remarque : impossible en C car le caractère 0 signe la fin de la chaîne de caractère)
2. $a = 1$
3. $a = 17$
4. a quelconque.

3 ENS - Info-Mathématiques 2022 - Section 2

La partie qui nous intéresse est la section 2. Ci-après sont inclues les pages correspondante. Le début de la première page et la fin de la dernière sont à ignorer.

Nous allons exécuter cet algorithme en partant de la paire d'états $\langle 0, 0' \rangle$ de l'automate suivant :



On représente sur une ligne les valeurs des variables R, $\langle x, y \rangle$, et F au point situé entre les lignes 2.1 et 2.2, en soulignant la paire en cours de traitement, $\langle x, y \rangle$. À la première itération, R est vide et on extrait la paire $\langle 0, 0' \rangle$ de F, qui est donc vide :

$$\underline{\langle 0, 0' \rangle}$$

On vérifie que $o(0) = o(0')$, et on insère les successeurs de la paire $\langle 0, 0' \rangle$ à la fin de F : $\langle 1, 1' \rangle$ selon la lettre a, puis $\langle 0, 0' \rangle$ selon la lettre b. La paire soulignée passe dans R et on souligne la nouvelle paire à traiter :

$$\langle 0, 0' \rangle \ \underline{\langle 1, 1' \rangle} \ \langle 0, 0' \rangle$$

on vérifie que $o(1) = o(1')$, on insère les successeurs de la paire $\langle 1, 1' \rangle$ ($\langle 0, 2' \rangle$ puis $\langle 1, 1' \rangle$) ; la paire soulignée passe dans R et on souligne la suivante :

$$\langle 0, 0' \rangle, \langle 1, 1' \rangle \ \underline{\langle 0, 0' \rangle} \ \langle 0, 2' \rangle, \langle 1, 1' \rangle$$

cette paire est déjà dans R, on la barre et on passe à la suivante :

$$\langle 0, 0' \rangle, \langle 1, 1' \rangle, \cancel{\langle 0, 0' \rangle}, \ \underline{\langle 0, 2' \rangle} \ \langle 1, 1' \rangle$$

on a $o(0) \neq o(2')$, l'algorithme renvoie faux.

Question 1.5 *Exécuter l'algorithme 1 pour la paire d'états $\langle 0, 3 \rangle$ de l'automate de la question 1.3. Utiliser la représentation précédente, et ne donner que la dernière ligne. Faire de même en partant de la paire $\langle 2, 5 \rangle$.*

Nous démontrerons la correction de cet algorithme dans la partie 3.

2 Fonctions d'ensembles, de relations

Soit E un ensemble, potentiellement infini. On note $\mathcal{P}(E)$ l'ensemble des parties de E. Notons que pour toutes parties $X, Y \subseteq E$, on a $X \subseteq Y$ si et seulement si $X \cup Y = Y$.

Une suite $(X_i)_{i \in \mathbb{N}}$ de parties de E est dite *croissante* (resp. *décroissante*) si pour tout indice i, $X_i \subseteq X_{i+1}$ (resp. $X_{i+1} \subseteq X_i$).

On s'intéresse aux fonctions de $\mathcal{P}(E)$ dans lui-même. On note id la fonction identité, et $f \circ g$ la composition de deux fonctions :

$$\mathrm{id} : X \mapsto X \qquad\qquad f \circ g : X \mapsto f(g(X))$$

On définit également la suite $(f^i)_{i\in\mathbb{N}}$ d'itérées d'une fonction f, par récurrence sur i :

$$f^0 \stackrel{\text{def}}{=} \mathrm{id}$$
$$f^{i+1} \stackrel{\text{def}}{=} f \circ f^i$$

Étant données deux fonctions $f, g : \mathcal{P}(E) \to \mathcal{P}(E)$, on note $f \cup g$ et $f \cap g$ les fonctions

$$f \cup g : X \mapsto f(X) \cup g(X) \qquad\qquad f \cap g : X \mapsto f(X) \cap g(X)$$

Plus généralement, étant donnée une famille $(f_i)_{i\in I}$ de fonctions, on note $\bigcup_{i\in I} f_i$ et $\bigcap_{i\in I} f_i$ les fonctions

$$\bigcup_{i\in I} f_i : X \mapsto \bigcup_{i\in I} f_i(X) \qquad\qquad \bigcap_{i\in I} f_i : X \mapsto \bigcap_{i\in I} f_i(X)$$

Une fonction $f : \mathcal{P}(E) \to \mathcal{P}(E)$ est dite

- *croissante* si pour tous $X, Y \subseteq E$ tels que $X \subseteq Y$, on a $f(X) \subseteq f(Y)$.
- *continue* si pour toute suite croissante $(X_i)_{i\in\mathbb{N}}$ de parties de E, on a $f\left(\bigcup_{i\in\mathbb{N}} X_i\right) = \bigcup_{i\in\mathbb{N}} f(X_i)$.
- *co-continue* si pour toute suite décroissante $(X_i)_{i\in\mathbb{N}}$ de parties de E, on a $f\left(\bigcap_{i\in\mathbb{N}} X_i\right) = \bigcap_{i\in\mathbb{N}} f(X_i)$.

La fonction identité est trivialement croissante, continue, et co-continue. De même, la composée de deux fonctions croissantes (resp. continues, co-continues) est croissante (resp. continue, co-continue), et l'union d'une famille de fonctions croissantes (resp. continues, co-continues) est croissante (resp. continue, co-continue).

À partir de la question suivante, toutes les réponses doivent être soigneusement justifiées.

Question 2.1 *Montrer que toute fonction continue est croissante.*

Question 2.2 *Montrer que toute fonction co-continue est croissante.*

Question 2.3 *Une fonction croissante est-elle toujours continue ? co-continue ? Justifier vos réponses.*

Etant donnée une fonction f, une partie $X \subseteq E$ est :

- un *post-point fixe (de f)* si $X \subseteq f(X)$,
- un *pré-point fixe (de f)* si $f(X) \subseteq X$,
- un *point fixe (de f)* si $X = f(X)$.

Question 2.4 *(i) Montrer que toute fonction admet un post-point fixe et un pré-point fixe.*
(ii) Donner une fonction qui n'admette pas de point fixe.

2.1 Plus grand point fixe

On va démontrer que toute fonction croissante admet un plus grand point fixe. On fixe pour cela une fonction croissante $f : \mathcal{P}(E) \to \mathcal{P}(E)$.

Question 2.5 *Montrer que l'union d'une famille de post-points fixes est un post-point fixe.*

Soit νf l'union de tous les post-points fixes de f :

$$\nu f \stackrel{\text{def}}{=} \bigcup_{X \subseteq f(X)} X$$

Par la question précédente, νf est un post-point fixe : $\nu f \subseteq f(\nu f)$.

Question 2.6 *(i) Montrer que νf est aussi un pré-point fixe.*
(ii) En déduire que c'est le plus grand point fixe (au sens de l'inclusion).

Question 2.7 *Montrer que f admet aussi un plus petit point fixe.*

Si l'on suppose de plus que la fonction f est co-continue, on peut caractériser son plus grand point fixe autrement. Remarquons tout d'abord que $(f^i(E))_{i\in\mathbb{N}}$ est une suite de parties de E.

Question 2.8 *(i) Montrer que la suite $(f^i(E))_{i\in\mathbb{N}}$ est décroissante.*
(ii) Montrer que si f est co-continue, alors $\nu f = \bigcap_{i\in\mathbb{N}} f^i(E)$.

2.2 Plus petite clôture

On ordonne les fonctions point à point : *f est contenue dans g*, noté $f \subseteq g$, si pour tout $X \subseteq E$, $f(X) \subseteq g(X)$.

Une fonction f est :

- *extensive* si $\mathrm{id} \subseteq f$ (c'est à dire, $\forall X, X \subseteq f(X)$) ;
- *saturante* si $f \circ f \subseteq f$ (c'est à dire, $\forall X, f(f(X)) \subseteq f(X)$) ;
- une *clôture* si elle est croissante, extensive, et saturante.

Etant donnée une fonction f, on pose $f^\omega \stackrel{\text{def}}{=} \bigcup_{i\in\mathbb{N}} (f \cup \mathrm{id})^i$.

Question 2.9 *(i) Montrer que f^ω est extensive et contient f.*
(ii) Montrer que si f est continue, alors f^ω est une clôture.

Question 2.10 *Montrer que si f est continue, alors pour tout X, $f^\omega(X)$ est le plus petit pré-point fixe de f contenant X.*

Question 2.11 *Déduire des questions précédentes que lorsque f est continue, f^ω est la plus petite clôture contenant f.*

2.3 Relations et fonctions de relations

On fixe dans cette partie un ensemble I. Une *relation* est une partie de $I \times I$. On s'intéresse aux fonctions des relations dans les relations. Autrement dit, on spécialise la partie précédente au cas $E = I \times I$. On note 1 la relation identité, $R \cdot S$ la composition de deux relations R, S, et $R^\intercal$ la transposée d'une relation R :

$$\begin{aligned} 1 &\stackrel{\text{def}}{=} \{\langle i, i\rangle \ \mid \ i \in I\} \\ R \cdot S &\stackrel{\text{def}}{=} \{\langle i, k\rangle \ \mid \ \exists j \in I,\ \langle i, j\rangle \in R \ \wedge \ \langle j, k\rangle \in S\} \\ R^\intercal &\stackrel{\text{def}}{=} \{\langle j, i\rangle \ \mid \ \langle i, j\rangle \in R\} \end{aligned}$$

Etant donnée une relation R, on notera parfois $x \ R \ y$ pour $\langle x, y\rangle \in R$.

Question 2.12 *Comment qualifie-t-on usuellement une relation R telle que $1 \subseteq R$? telle que $R^\intercal \subseteq R$? telle que $R \cdot R \subseteq R$?*

Soient $r, s, t : \mathcal{P}(I \times I) \to \mathcal{P}(I \times I)$ les trois fonctions suivantes :

$$r : R \mapsto 1 \qquad\qquad s : R \mapsto R^\intercal \qquad\qquad t : R \mapsto R \cdot R$$

Question 2.13 *Qu'est-ce qu'un pré-point fixe pour r ? pour s ? pour t ? pour $r \cup t$? pour $r \cup s \cup t$?*

Question 2.14 *Montrer que les trois fonctions r, s et t sont continues.*

Au vu des questions 2.10 et 2.11, on appelle usuellement la fonction t^ω *clôture transitive* : c'est une clôture, et $t(R)$ est la plus petite relation transitive contenant R. De manière similaire, les fonctions $(r \cup t)^\omega$ et $(r \cup s \cup t)^\omega$ sont respectivement les fonctions de clôture réflexive-transitive et réflexive-symétrique-transitive.

3 Equivalence de deux états

On cherche désormais des algorithmes permettant de tester l'équivalence de deux états dans un automate fini donné. Nous allons tout d'abord montrer que l'algorithme proposé en partie 1 est correct, et de complexité quadratique. On fixe dans toute cette partie un automate $\langle X, o, \delta\rangle$, dont l'ensemble d'états X est fini, de taille n.

3.1 Un algorithme quadratique

Soit $p : \mathcal{P}(X \times X) \to \mathcal{P}(X \times X)$ la fonction suivante entre relations sur l'ensemble X des états :

$$p : R \mapsto \{\langle x, y\rangle \ \mid \ o(x) = o(y) \ \wedge \ \forall a \in A,\ \delta(x, a) \ R \ \delta(y, a)\}$$

Pour toutes relations R, S, en déroulant cette définition, on a

$$S \subseteq p(R) \quad \text{si et seulement si} \quad \forall x, y \in X,\ x \ S \ y \ \Rightarrow \ o(x) = o(y) \ \wedge \ \forall a \in A,\ \delta(x, a) \ R \ \delta(y, a)$$

On appelle *bisimulation* tout post-point fixe de p : toute relation R telle que $R \subseteq p(R)$.