Exercice 1 [sur 10]

1- Le PCP est un problème *indécidable*.

2- Pour chacun des problèmes P de la question 2, nous donnons soit un algorithme qui le résout (cas 1), soit une réduction (many-one) de PCP au problème P (cas 2). Dans le cas 1 P est décidable, dans le cas 2, P est indécidable.

a-Problème PCP* :

Le mot $w := \varepsilon$ vérifie $\varphi(w) = \psi(w) = \varepsilon$. L'algorithme qui lit l'instance (A, B, φ, ψ) puis retourne la réponse "oui" résout ce problème.

b-Problème PCP-RAT :

Soit $\mathcal{A}_0$ un automate fini qui reconnait le langage A^+ (un automate déterministe à deux états convient).

Soit $\rho : (A, B, \varphi, \psi) \mapsto (A, B, \mathcal{A}_0, \varphi, \psi)$. La réponse est "oui" sur $\rho(A, B, \varphi, \psi)$ ssi

$$\exists w \in \mathrm{L}(\mathcal{A}_0), \varphi(w) = \psi(w),$$

ssi

$$\exists w \in A^+, \varphi(w) = \psi(w),$$

ssi la réponse au PCP est "oui" sur (A, B, φ, ψ). L'application ρ est donc une réduction many-one de PCP à PCP-RAT.

c-Problème PCP-PROD :

Si $w \in A^+$ vérifie que $\varphi(w) = \psi(w)$, alors $\varphi(w \cdot w) = \psi(w \cdot w)$, donc $u := w, v := w$ est une solution au PCP-PROD (A, B, φ, ψ).

Réciproquement, si (u, v) est une solution au PCP-PROD (A, B, φ, ψ) : $\varphi(u \cdot v) = \psi(u \cdot v)$ donc $w := u \cdot v$ est une solution au PCP (A, B, φ, ψ). L'application ρ qui est l'identité i.e. $\rho(A, B, \varphi, \psi) := (A, B, \varphi, \psi)$ est donc une réduction many-one de PCP à PCP-PROD.

d-Problème PCP-IND :

Les langages $\varphi(A^+), \psi(A^+)$ sont rationnels. A partir des ensembles finis de mots $\{\varphi(a) \mid a \in A\}, \{\psi(a) \mid a \in A\}$ on peut construire des automates finis déterministes $\mathcal{C}_\varphi, \mathcal{C}_\psi$ tels que

$$\mathrm{L}(\mathcal{C}_\varphi) = \varphi(A^+), \;\; \mathrm{L}(\mathcal{C}_\psi) = \psi(A^+).$$

Comme la famille des langages rationnels est effectivement close par intersection, on peut calculer un troisième automate fini $\mathcal{C}$ tel que

$$\mathrm{L}(\mathcal{C}) = \mathrm{L}(\mathcal{C}_\varphi) \cap \mathrm{L}(\mathcal{C}_\psi).$$

On teste alors si $\mathrm{L}(\mathcal{C}) \neq \emptyset$: on répond 'oui" (au PCP-IND) ssi ce langage est non-vide.

Exercice 2 [sur 20]

Partie 1

Fonctions à sens unique.

1- La fonction f^{-1} est calculée par l'algorihme suivant (en pseudo-code) :

```
def finv(y:X^*):X^*
(* returns the unique x s.t. f(x)== y *)
begin
1   FINI=false; x= ε
2   WHILE not FINI:
3       if f(x) == y then:
4           return x
5       else:
6           x=suc(x)
end
```

où `suc` designe la fonction successeur pour l'ordre hiérarchique sur X^*. Comme f est calculable : le calcul de la ligne 3 se termine.
Comme f est surjective : après un temps fini le test de la ligne 3 prend la valeur `true` ce qui fait sortir de la boucle 2.
Lorsque le programme se termine le test 3 vient d'avoir la valeur `true`, donc la valeur retournée est un mot x tel que $f(x) = y$.
2- Soit $f : X^* \to X^*$ totale. Définissons une fonction

$$\bar{f} : X^* \to X^*$$

par

$$\begin{aligned} \text{si } y \in \mathrm{Im}(f), \quad \bar{f}(y) &:= \min\{x \in X^* \mid f(x) = y\} \\ \text{sinon }, \quad \bar{f}(y) &:= \varepsilon. \end{aligned}$$

Soit $x \in X^*$. Comme $f(x) \in \mathrm{Im}(f)$,$\bar{f}(f(x)) = x'$ où $f(x') = f(x)$. Donc $f(\bar{f}(f(x))) = f(x') = f(x)$.
3- 3.1 Soit f, $\bar{f} : X^* \to X^*$ totales, calculables, vérifiant les deux hypothèses :

$$\mathrm{Im}(f) \text{ n'est pas récursive} \tag{1}$$

$$f \circ \bar{f} \circ f = f \tag{2}$$

Considérons l'algorithme suivant :

```
def A(y:X^*):int
(* tests whether y is in Im f*)
begin
1   x=f̄(y)
2   if f(x) == y then:
3       return 1:
4   else:
5       return 0
end
```

Supposons que $y \in \mathrm{Im}f : y = f(x_0)$.
L'algorithme A teste que $f(\bar{f}(y)) == y$? (ligne 2).
Comme $f(\bar{f}(y)) == f(\bar{f}(f(x_0))) == f(x_0)$ (par l'hypothèse (2)) le test renvoie `true` et A renvoie 1.
Supposons que $y \notin \mathrm{Im}f$. le test de la ligne 2 renvoie `false`, donc A renvoie 0.
Donc on a :

$$\forall y \in X^*, y \in \mathrm{Im}f \Leftrightarrow A(y) = 1$$

donc $\mathrm{Im}f$ serait récursive, ce qui contredit l'hypoth'ese (1). Nous avons ainsi montré par l'absurde que , si $\mathrm{Im}(f)$ n'est pas récursive, alors f n'a pas de speudo-inverse calculable.
3.2 Construisons une application f totale, calculable, vérifiant l'hypothèse (1). Nous commencons par construire une fonction $f : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$.
On considère une numérotation effective $n \mapsto M_n$ des machines de Turing déterministes, à une bande.Posons :

$$\begin{aligned} f(n,m) &:= n \text{ si } M_n \text{ s'arrête en } \leq m \text{ étapes} \\ &:= 0 \text{ sinon .} \end{aligned}$$

On vérifie que : pour tout entier $n \neq 0$,

$$n \in \mathrm{Im}f \Leftrightarrow M_n \text{ s'arrête}$$

Donc $\mathrm{Im}f$ n'est pas récursive.
3.3 Soit $\gamma : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ définie par

$$\gamma(p,q) := \frac{(p+q)(p+q+1)}{2} + q.$$

L'application γ est une bijection calculable . Donc

$$f \circ \gamma^{-1} : \mathbb{N} \to \mathbb{N}$$

est une fonction totale, calculable, vérifiant l'hypothèse (1). Par conséquent elle n'admet aucun pseudo-inverse calculable (par 3.1).

Partie 2

Considérons une fonction $f : X^* \to X^*$ telle que

$$f \text{ est à sens unique} \tag{3}$$

4- Définissons un *témoin* pour (u, w) comme un mot

$$v \in X^* \text{ tel que } |v| \leq q(|w|) \text{ et } f(u \cdot v) = w.$$

On définit un vérifieur $V : (X^* \times X^*) \times X^* \to \{0, 1\}$ par :

$$V((u,w),v) = 1 \text{ si et seulement si } f(u \cdot v) = w.$$

Comme f est dans **FP** (C1), cette application V est aussi dans la classe **FP**. L'ensemble $L \subseteq X^* \times X^*$ des solutions (u, w) au problème PREFIX-INVERSE est l'ensemble des couples

(u, w) qui possèdent un témoin v vérifiant :

$$|v| \leq q(|w|) \text{ et } V((u, w), v) = 1.$$

Donc L est dans la classe NP.
5- Supposons que PREFIX-INVERSE est résolu par un algorithme A qui fonctionne en temps polynomial.

```
def g(w:X^*):X^*
(* returns an u such that f(u) =  w or ε in case such an u does not exist *)
begin
1   FINI=false
2   u = ε
3   WHILE not FINI:
4       if f(u)==w:
5           return u
6       for x in X:
7           found= false
8           if A(ux,w):
9               u=ux;found=true
10              break
11      if not found:
12          return ε
13      if |u|  >  q(|w|):
14          return ε
end
```

La fonction g ci-dessus s'appuie sur l'algorithme A pour calculer un antécédent u de w , selon un procédé glouton, jusqu'à :
- soit trouver un tel mot u (ligne 4)
- soit s'être assuré qu'un tel mot u n'existe pas (lignes 11,13).

On vérifie ainsi que, si $w \in \text{Im}f$, alors $f(g(w)) = w$. Donc $f \circ g \circ f = f$ i.e. g est un pseudo-inverse de f.
De plus, comme A est dans **FP**, g est aussi dans **FP**.
-6 Supposons que f est une fonction à sens unique.Considérons le problème PREFIX-INVERSE associé à cette fonction f. On a vu à la question 5 que, si PREFIX-INVERSE est dans **P**, alors f admet un pseudo-inverse $\bar{f}$ dans la classe **FP**, ce qui est contraire à la condition (C3) sur f.
Donc PREFIX-INVERSE est un problème dans la classe $\mathbf{NP} \setminus \mathbf{P}$.

Partie 3

Considérons maintenant un langage $L \subseteq X^*$ tel que

$$L \text{ appartient à la classe de complexité } \mathbf{NP} \setminus \mathbf{P}. \tag{4}$$

7- Le calcul de $f(w)$ consiste à :
a- tester si $w \notin X^*\#X^*$?
b- extraire de w les mots u, v
c- tester si $|v| \leq p_1(|u|)$
d- calculer $u\#V(u,v)$
e- ou calculer $\#\#w$.
a,b,e, prennent un temps linéaire. Le test c prend un temps polynomial. Le calcul d prend un temps $\leq p_2(|w|)$. Le temps de calcul total et donc majoré par une fonction polynomiale.
8- La définition de f distingue 3 cas.
Dans le cas 1 :

$$|f(u\#v) = |u| + 2 \leq |u\#v| + 2.$$

Dans les cas 2,3

$$|f(u\#v) = |\#\#u\#v| = |u\#v| + 2.$$

Donc la longueur de l'image est bien majorée par un polynôme de la longueur de l'argument.
Inversement :
Dans le cas 1 :

$$|u\#v| \leq |u| + 1 + p_1(|u|) \leq p_3(|u\#v|).$$

en posant $p_3(z) := z + 1 + p_1(z)$.
Dans les cas 2,3 :

$$|w| \leq |f(w)|.$$

Donc la longueur de w est bien majorée par un polynôme de la longueur de $f(w)$.
*9-Supposons que $\bar{f} : X^* \to X^*$ est totale et $f \circ \bar{f} \circ f = f$. Montrons que $\bar{f} \notin \mathbf{FP}$. Nous raisonnons par l'absurde : nous supposons que

$$\bar{f} \in \mathbf{FP}. \tag{5}$$

Considérons l'algorithme :

```
def A(u:X^*):bool
(* returns true iff u in L *)
begin
1   b=f̄(u#1)
2   if f(b) == u#1
3       return true
4   else:
5       return false
end
```

Si $u \in L$:

$$\exists v \in X^*, |v| \leq p_1(|u|) \text{ tel que } V(u,v) = 1.$$

Donc $f(u\#v) = u\#1$,
donc, puisque $\bar{f}$ est pseudo-inverse de f :

$$f(\bar{f}(f(u\#v))) = u\#1$$

c'est à dire

$$f(\bar{f}(u\#1)) = u\#1$$

donc le test de la ligne 2 est positif, ce qui entraîne que

$$A(u) = true.$$

Si $A(u) = true$:
Autrement dit :

$$f(\bar{f}(u\#1)) = u\#1 \tag{6}$$

Comme $u\#1$ n'est pas de la forme $\#\#w$ pour un mot $w \in Y^*$, le calcul de f sur l'argument $\bar{f}(u\#1)$ suit le cas 1 : il existe $v \in X^*$ tel que

$$\bar{f}(u\#1) = u\#v,\ |v| \leq p_1(|u|) \text{ et } f(\bar{f}(u\#1)) = u\#V(u,v)$$

ce qui, en présence de l'égalité (6) donne :

$$u\#1 = u\#V(u,v)$$

d'où

$$V(u,v) = 1 \text{ et } |v| \leq p_1(|u|)$$

ce qui montre que

$$u \in L.$$

Nous avons montré que A est un algorithme de reconnaissance de L. De plus. comme $f, \bar{f}$ sont dans **FP**, A est aussi dans **FP**. Ceci contredit l'hypothèse que $L \notin \mathbf{P}$ (4). Donc un pseucdo-inverse de f vérifiant l'ypothèse (5) *n'existe pas* : (C3) est vraie.
10- Si $L \in \mathbf{NP} \setminus \mathbf{P}$, alors la fonction f étudiée dans la partie 3 est à sens unique.