

Exercice 1.

1. Soit L un langage reconnaissable en temps polynomial : *i.e.* il existe un algorithme $\mathcal{A}$ et une constante r tels que pour tout mot w , l'algorithme traite w en temps $O(|w|^r)$ et $w \in L$ si l'algorithme retourne "oui" et $w \notin L$ si l'algorithme retourne "non".

Montrer qu'il existe un algorithme polynomial qui reconnaît L^* et donner son temps d'exécution en fonction de $|w|$ et r .

2. Soit un mot $w = xy$ avec $|x| = |y|$. Pour un tel mot, nous définissons $\text{permute}(w) = yx$. Pour simplifier, nous définissons $\text{permute}(w) = \varepsilon$ si $|w|$ est impair. Si L est un langage, nous définissons $\text{permute}(L) = \{\text{permute}(w) | w \in L\}$.

- (a) Montrer que si L est régulier, alors $\text{permute}(L)$ n'est pas nécessairement régulier.
- (b) Montrer que si L est régulier, alors $\text{permute}(L)$ est algébrique.
- (c) Montrer que si L est algébrique, alors $\text{permute}(L)$ n'est pas nécessairement algébrique.

Exercice 2. La logique de Hoare manipule des triplets. Un triplet $\{P\} \mathbb{C} \{Q\}$ décrit les conséquences Q de l'exécution de la ligne de code $\mathbb{C}$, à condition que la propriété P soit vérifiée avant cette exécution. Par exemple, il est clair que le triplet $\{y = 2\} x := y \{x = 2\}$ est valide. La logique de Hoare fournit ces règles pour chacune des constructions d'un langage impératif simple ¹:

$\boxed{\text{L1}} \frac{}{\{P[E/x]\} \ x:=E \ \{P\}}$	$\boxed{\text{L2}} \frac{\{P\} \ S \ \{Q\} \ , \ \{Q\} \ T \ \{R\}}{\{P\} \ S;T \ \{R\}}$
$\boxed{\text{L3}} \frac{P' \Rightarrow P \ , \ \{P\} \ S \ \{Q\} \ , \ Q \Rightarrow Q'}{\{P'\} \ S \ \{Q'\}}$	$\boxed{\text{L4}} \frac{\{P \wedge B\} \ S \ \{P\}}{\{P\} \ \text{while } B \ \text{do } S \ \text{done} \ \{\neg B \wedge P\}}$

Voici un exemple de l'utilisation de la logique de Hoare pour prouver le triplet $\{x > 0\} x := x + 1; x := x + 1 \{x > 2\}$:

$$\boxed{\text{L2}} \frac{\boxed{\text{L3}} \frac{x > 0 \Rightarrow x + 1 > 1 \quad \boxed{\text{L1}} \frac{}{\{x + 1 > 1\} x := x + 1 \{x > 1\}} \quad x > 1 \Rightarrow x > 1 \quad \boxed{\text{L3}} \frac{x > 1 \Rightarrow x + 1 > 2 \quad \boxed{\text{L1}} \frac{}{\{x + 1 > 2\} x := x + 1 \{x > 2\}} \quad x > 2 \Rightarrow x > 2}{\{x > 0\} x := x + 1 \{x > 1\}} \quad \{x > 1\} x := x + 1 \{x > 2\}}{\{x > 0\} x := x + 1; x := x + 1 \{x > 2\}}$$

¹La construction $\frac{A}{B}$ signifie que si A est valide alors B l'est aussi, et la construction $P[E/x]$ représente l'élément P modifié de telle sorte que chaque occurrence de x dans P a été remplacée par l'expression E

1. Que fait le programme suivant, pour lequel n est supposé initialisé à une valeur entière strictement positive ?

```
y := 1 ;  
i := n ;  
while i > 1 do  
  y := y * i ;  
  i := i - 1 ;  
done
```

2. Prouver ce comportement de la même manière que dans l'exemple cité plus haut (c'est à dire en utilisant un arbre dont chaque branchement correspond à une règle parmi $\boxed{\text{L1}}$, $\boxed{\text{L2}}$, $\boxed{\text{L3}}$ et $\boxed{\text{L4}}$).

3. Prouver que le triplet $\{n \geq 0\} \ C \ \{z = n^3\}$ est valide, où C est le programme suivant :

```
x := 0 ;  
y := 0 ;  
z := 0 ;  
while not (x=n) do  
  z := z+3y+3x+1 ;  
  y := y+2x+1 ;  
  x := x+1  
done
```

4. Proposer une règle $\boxed{\text{L5}}$, similaire à $\boxed{\text{L1}}$, $\boxed{\text{L2}}$, $\boxed{\text{L3}}$, $\boxed{\text{L4}}$ pour la construction usuelle `if  $B$  then  $S$  else  $T$` , où B est une condition booléenne and S and T sont des propriétés.

Exercice 3. (Exercice pour les candidats de la discipline secondaire)

Soit A un anneau commutatif unitaire.

PARTIE I : DIVISION EUCLIDIENNE RAPIDE PAR LA MÉTHODE DE NEWTON

Soient $S, T \in A[X]$ avec $\deg(S) = n$, $\deg(T) = m$ et T unitaire.

1. Montrer que l'algorithme classique de division euclidienne de S par T a une complexité arithmétique en $O(n^2)$.
2. Pour $P \in A[X]$ et $k \geq \deg(P)$, nous notons $\text{Rec}_k(P(X)) = X^k P(1/X)$. Montrer que

$$\text{Rec}_{n-m}(Q) = \text{Rec}_n(S) \text{Rec}_m(T)^{-1} \bmod X^{n-m+1},$$

où Q est le quotient de la division euclidienne de S par T .

3. Soit $F \in A[X]$ avec $F(0) = 1$. Considérons la suite de polynômes $G_i \in A[X]$ définie par $G_0 = 1$ et

$$G_{i+1} = 2G_i - F \cdot G_i^2 \bmod X^{2^{i+1}}$$

pour $i \geq 0$. Montrer que pour tout entier $i \geq 0$, nous avons

$$F \cdot G_i \equiv 1 \bmod X^{2^i}.$$

4. En déduire un algorithme pour calculer Q et le reste R de la division euclidienne de S par T .
5. Montrer que la complexité arithmétique de ce nouvel algorithme de division euclidienne appliqué à deux polynômes de degré $< n$ est en $O(M(n))$ où $M(n)$ est la complexité arithmétique du produit de deux polynômes de degré $< n$ de $A[X]$ (avec $M(n+m) \geq M(n) + M(m)$ pour $m, n \in \mathbb{N}$).

PARTIE II : ÉVALUATION RAPIDE DE POLYNÔMES

Soit $P \in A[X]$ unitaire avec $\deg(P) = n$. Soient $a_1, \dots, a_n \in A$.

1. Supposons que $n = 2^k$ est une puissance de 2 et considérons un l'arbre binaire complet T à n feuilles défini par :
 - chacune des n feuilles est associée à un polynôme $X - a_j$ pour $j \in \{1, \dots, n\}$;
 - pour chaque nœud interne u ayant les fils v et w associés aux polynômes $M_v(X)$ et $M_w(X)$ respectivement, u est associé au polynôme $M_u(X) = M_v(X) \cdot M_w(X)$.
 - (a) Donner un algorithme pour construire l'arbre T avec une complexité arithmétique en $O(M(n) \log n)$.
 - (b) Donner un algorithme qui prenant en entrée P , $(a_1, \dots, a_n)$ et T , calcule $P(X) \bmod M_u(X)$ pour tout $u \in T$, avec une complexité arithmétique en $O(M(n) \log n)$.
2. Dédire des questions précédentes un algorithme qui calcule $P(a_1), \dots, P(a_n)$ pour tout $n \in \mathbb{N}$ avec une complexité arithmétique en $O(M(n) \log n)$.