

## Devoir (à rendre au plus tard le 1<sup>er</sup> décembre 2008)

### Exercice 1

Parmi les ensembles suivants, lesquels sont dénombrables ? (Justifier soigneusement chaque réponse.)

1. l'ensemble des entiers qui ne sont pas des nombres premiers D
2. l'ensemble des nombres décimaux (tels que 0,5 ou 3,14) D
3. l'ensemble des suites finies de nombres réels M
4. l'ensemble des fonctions partielles de  $\mathbb{N}$  dans {0,1} ND
5. l'ensemble des listes du type float D
6. l'ensemble des machines de Turing qui s'arrêtent sur la bande vierge D
7. l'ensemble des fonctions totales croissantes de  $\mathbb{N}$  dans {0,1} D

### Exercice 2

Parmi les problèmes suivants (dont la donnée est une machine de Turing  $M$ ), lesquels sont décidables ? (Justifier soigneusement chaque réponse.)

1.  $M$  est-elle déterministe ? D
2. En travaillant sur la bande vierge,  $M$  passe-t-elle plus d'une fois par son état initial ? ID
3.  $M$  s'arrête-t-elle sur la bande vierge après au plus 5 pas de calcul ? D
4.  $M$  s'arrête-t-elle sur tout mot d'entrée après au plus 5 pas de calcul ? D
5. Existe-t-il un mot d'entrée  $x$  tel qu'en travaillant sur  $x$ ,  $M$  s'arrête après au plus 5 pas de calcul ? D

### Exercice 3

On se donne une énumération  $M_0, M_1, \dots, M_n, \dots$  des machines de Turing. Parmi les fonctions suivantes de  $\mathbb{N}$  dans  $\mathbb{N}$ , lesquelles sont calculables ? (Justifier soigneusement chaque réponse.)

1.  $f(n) =$  le plus grand entier inférieur ou égal à  $\log(1 + n\sqrt{n})$  C
2.  $f(n) =$  le plus petit entier  $x$  tel que la machine  $M_n$  s'arrête sur  $x$  en au plus 5 pas de calcul C
3.  $f(n) = 1$  si la machine  $M_n$  s'arrête sur la bande vierge en au plus  $2^n$  pas de calcul, et 0 sinon C
4.  $f(n) = 1$  si la machine  $M_n$  s'arrête sur l'entrée  $n$  (non définie sinon) C

Un ensemble  $E$  est dénombrable s'il est en bijection avec  $\mathbb{N}$

$E \subseteq \mathbb{N}$  est décidable = récursif

$\Leftrightarrow$  il existe une MT s'arrêtant sur toute entrée

$\Leftrightarrow$  il existe un algorithme implementant  $d(m) = \begin{cases} 1 & \text{si } m \in E \\ 0 & \text{sinon} \end{cases}$

$E \subseteq \mathbb{N}$  est récursivement énumérable

$\Leftrightarrow$  il existe une MT telle que  $L(M)$  est l'ensemble des écritures de  $E$

$\Leftrightarrow$  il existe un algorithme de semi-décision pour  $d(m) = \begin{cases} 1 & \text{si } m \in E \\ \uparrow & \text{sinon} \end{cases}$

$f$  est une fonction calculable de  $\mathbb{N}$  dans  $\mathbb{N}$

$\Leftrightarrow$  il existe une MT qui calcule  $g$ ,  $g(p) = T(m, p) \quad \forall p \in \mathbb{N}$

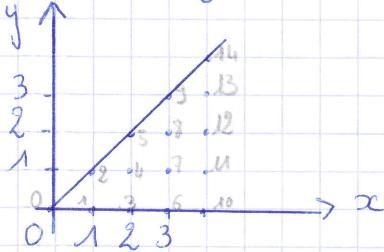
(les valeurs de  $g$  seront la  $n$ -ième ligne).

dicky@labri.fr

fonction calculable = il existe un algorithme qui fait ce traitement

### exercice 1.1 :

$$E = \{(x, y) \in \mathbb{N} \times \mathbb{N} \mid y \leq x\}$$



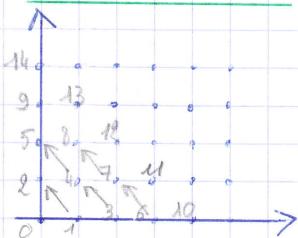
$$\begin{aligned}f(x, y) &= x + (x-1) + \dots + (x-(x-1)) + (x-x) + y \\&= \frac{x(x+1)}{2} + y\end{aligned}$$

Réciproque :

$$x = \text{le plus grand entier tel que } \frac{x(x+1)}{2} \leq m$$

$$y = m - \frac{x(x+1)}{2}$$

### exercice 1.2 :



$$f(x, y) = \frac{(x+y)(x+y+1)}{2} + y$$

$$f(x, y) = (2x+1)2^y$$

trouver  $x$  et  $y$  tels que  $f(x, y) = m$

$$6 \rightarrow (1, 1) \quad 6 = 3 \times 2^1$$

$$7 \rightarrow (3, 0) \quad 7 = 7 \times 2^0$$

$$16 \rightarrow (0, 4) \quad 16 = 31 \times 2^4$$

$y$  : le plus grand entier tel que  $2^y$  divise  $m$

$2x+1$  : le plus grand nombre impair qui divise  $m$ .

$$m = \underbrace{10110010}_{x} \underbrace{11000000}_{y \text{ Zéros}}$$

$f$  est une bijection de  $\mathbb{N} \times \mathbb{N}$  sur  $\mathbb{N} - \{0\}$

$f(x, y) = (2x+1)2^y - 1$  est une bijection de  $\mathbb{N} \times \mathbb{N}$  dans  $\mathbb{N}$ .

### exercice 1.3 :

$$\begin{array}{ccc} \mathbb{Z} \times \mathbb{Z} & \longrightarrow & \mathbb{N} \\ (x, y) & \downarrow & (2x' + 1)2^{y'} - 1 \\ \mathbb{N} \times \mathbb{N} & & \text{avec } x' = g(x) \\ (x', y') & & y' = g(y) \end{array}$$

où  $g(z) = \begin{cases} 2z & \text{si } z \geq 0 \\ -2z-1 & \text{si } z < 0 \end{cases}$

$g^{-1}(m) = \begin{cases} m/2 & \text{si } m \text{ est pair} \\ -(m+1)/2 & \text{si } m \text{ est impair} \end{cases}$

### exercice 1.4 :

$$\begin{array}{ccc} \mathbb{N} \times \mathbb{N} \times \mathbb{N} & \longrightarrow & \mathbb{N} \times \mathbb{N} \\ (x, y, z) & \xrightarrow{f} & (f(x, y), z) \xrightarrow{f_2} f_2(f(x, y), z) = f_3(x, y, z) \\ f_p(x_1, x_2, \dots, x_{p-1}, x_p) = f_p(f_{p-1}(x_1, x_2, \dots, x_{p-1}), x_p) \end{array}$$

### exercice 1.5 :

bijection donnée dans l'énoncé ?

$$m \geq 1 \rightarrow \text{liste}$$

$$m = 2^1 \rightarrow [1]$$

$$m = 3^1 \rightarrow [0, 1]$$

$$m = 4 \rightarrow [2]$$

$$m = 5 \rightarrow [0, 0, 1]$$

$$[m_1, m_2, \dots, m_l] \leftrightarrow 2^{m_1} \times 3^{m_2} \times \dots \times p^{m_l+1}$$

$$0 : ?$$

$$1 : ?$$

$$2 = 2^{0+1} \rightarrow [0]$$

$$3 = 2^0 \times 3^{0+1} \rightarrow [0, 0]$$

$$4 = 2^{1+1} \rightarrow [1]$$

problème pour la liste vide [] et les listes qui se terminent par un zéro [-, 0], [-, -, 0], [-, -, -, 0], ...

problème pour 0

numérotation de Gödel

problème pour la liste vide

pour 0 et 1

$$\Rightarrow [m_1, m_2, \dots, m_l] \leftrightarrow 2^{m_1} \times 3^{m_2} \times \dots \times p^{m_l+1} - 1$$

$$0 \leftrightarrow []$$

$$m \rightarrow m+1 = 2^{m_1} \times 3^{m_2} \times \dots \times p^{m_l} \rightarrow [m_1, m_2, \dots, m_l - 1]$$

avec  $m_l > 0$

## Modèles de Calcul : feuille 1

Bijections “calculables”

### Exercice 1.1

Donner une bijection de  $\{(x, y) \in \mathbb{N} \times \mathbb{N} \mid y \leq x\}$  dans  $\mathbb{N}$ , avec des algorithmes explicites de calcul pour la fonction directe et sa réciproque.

### Exercice 1.2

Donner trois exemples de bijections de  $\mathbb{N} \times \mathbb{N}$  dans  $\mathbb{N}$ .

La fonction  $f$  définie par  $f(x, y) = (2x + 1)2^y$  est-elle une solution correcte ? Sinon, pouvez-vous la corriger ?

### Exercice 1.3

Donner une bijection de  $\mathbb{Z} \times \mathbb{Z}$  dans  $\mathbb{N}$ , avec des algorithmes de calcul pour la fonction directe et sa réciproque.

### Exercice 1.4

Donner une bijection de  $\mathbb{N}^3$  dans  $\mathbb{N}$ , avec une formule explicite.

Plus généralement, donner un algorithme de calcul de  $\varphi(p, x_1, x_2, \dots, x_p)$  de façon que, pour tout entier ~~p > 1~~, la fonction  $(x_1, x_2, \dots, x_p) \rightarrow \varphi(p, x_1, x_2, \dots, x_p)$  soit une bijection de  $\mathbb{N}^p$  dans  $\mathbb{N}$ . ~~p > 2~~

### Exercice 1.5

Construire une bijection de l'ensemble  $\mathcal{L}$  des listes finies d'entiers naturels dans  $\mathbb{N}$ .

Indication : on sait que tout entier  $n > 1$  a une décomposition unique sous la forme  $n = 2^{m_1} \times 3^{m_2} \times \cdots \times p_k^{m_k} \times \cdots \times p_\ell^{m_\ell}$ , où  $p_k$  est le  $k$ -ième nombre premier et  $m_\ell \neq 0$  : ce qui donne une bijection de  $\mathbb{N} - \{0, 1\}$  sur l'ensemble des suites qui sont soit la liste  $(0)$ , soit une liste ne se terminant pas par  $0$ .

⚠️ Des listes infinies ne sont pas numérotables (diagonalisation)

### Exercice 1.6

1- Donner une bijection de l'ensemble des arbres finis (non étiquetés) dans  $\mathbb{N}$ .

Indication : un arbre  $A$  peut être vu comme la liste de ses sous-arbres. Autrement dit soit  $A = ()$  (arbre réduit à un seul nœud)

soit  $A = (A_1, A_2, \dots, A_\ell)$  où  $A_1, A_2, \dots, A_\ell$  sont des arbres planaires finis et  $\ell \geq 1$ .

2- Donner une bijection de l'ensemble des arbres étiquetés (arbres dont chaque nœud est étiqueté par un entier naturel) dans  $\mathbb{N}$ .

Indication : un arbre étiqueté  $A$  peut être vu comme la liste formée de l'étiquette de sa racine suivie de la liste (finie) de ses sous-arbres. Autrement dit,  $A = (r, A_1, A_2, \dots, A_\ell)$  où  $A_1, A_2, \dots, A_\ell$  sont des arbres et  $\ell \geq 0$ .

### Exercice 1.7

L'ordre hiérarchique sur les mots de  $\{a, b, c\}^*$  est l'ordre du dictionnaire des mots croisés :  $u \leq v$  si soit  $|u| < |v|$ , soit  $|u| = |v|$  et  $u \leq_{\text{lex}} v$ . Ainsi :

$$\varepsilon < a < b < c < aa < ab < ac < ba < \dots$$

On définit  $\varphi(u)$  comme le nombre de mots  $v \in \{a, b, c\}^*$  tels que  $v < u$ . Autrement dit,  $\varphi(u)$  est le numéro de  $u$  selon l'ordre hiérarchique.

- 1- Exprimer  $\varphi(ua)$  en fonction de  $\varphi(u)$ .
- 2- Donner une formule simple pour  $\varphi$ .

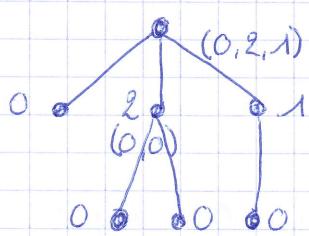
l'ensemble des mots sur un alphabet fini est dénombrable.

d'ensemble des fonctions de  $\mathbb{N}$  dans  $\mathbb{N}$  n'est pas dénombrable.

### exercice 1.6:

1. 0 si  $\Delta = ()$

ou  $(\underline{A_1}, \underline{A_2}, \dots, \underline{A_m})$  numéra de la liste  $[num(A_1), \dots, num(A_m)]$



0  $\rightarrow$  .

1  $\rightarrow$  ]

2  $\rightarrow$

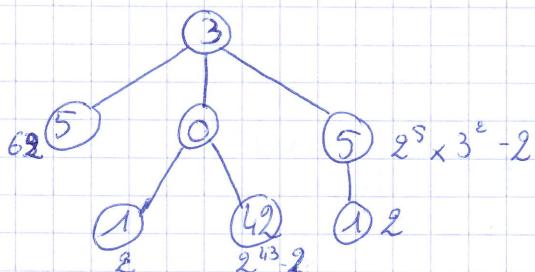
3  $\rightarrow$  [1]  $\rightarrow$

2.  $gödel([n, num(A_1), \dots, num(A_m)]) - 1$

0  $\rightarrow$  [0]  $\rightarrow$  ①

2  $\rightarrow$  [1]  $\rightarrow$  ②

1  $\rightarrow$  [0,0]  $\rightarrow$



### exercice 1.7:

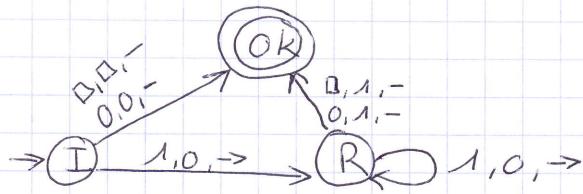
1. Soit  $m \in \mathbb{N}$  tel que  $m \mid u$

$$m = (\text{valeur entière } \varphi(u)) - 1$$

$$\varphi(ua) = \varphi(u) + 3^m \quad \ell(ua) = 3\ell(u) + 1$$

2.  $\varphi(u) = 3^{m-1} + 3^{m-2} + \dots + 3^{-(m-1)} + 3^{m-m} + m$  avec  $0 \leq m < 3^m$

### exercice 2.1:



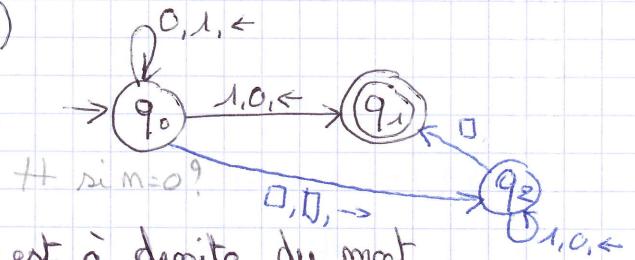
$$m \rightarrow \begin{cases} m+1 & \text{si } m \text{ est impair} \\ m & \text{si } m \text{ est pair} \end{cases}$$

### exercice 2.2:

$$M = (\{q_0, q_1\}, q_0, \{q_1\}, \{0, 1, \square\}, \{0, 1\}, \delta)$$

$$\delta = \begin{array}{l} q_0 \xrightarrow{0, 1, \leftarrow} q_0 \\ q_0 \xrightarrow{1, 0, \leftarrow} q_1 \end{array}$$

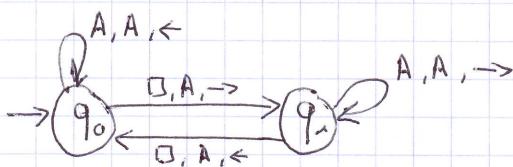
À la début du travail, la tête de lecture est à droite du mot.



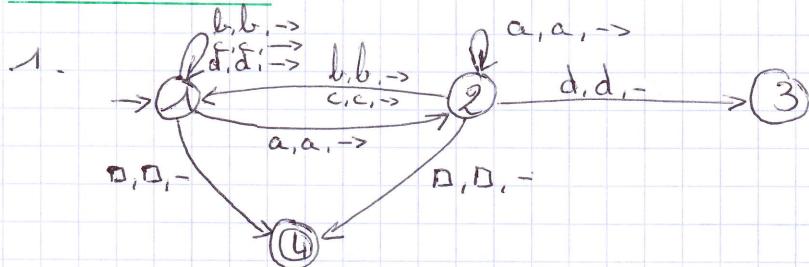
### exercice 2.3:

$$M = (\{q_0, q_1\}, q_0, \dots, \{A, \square\}, \{A\}, \delta)$$

$$\delta = \begin{array}{l} q_0 \xrightarrow{\square, A, \rightarrow} q_1 \\ q_0 \xrightarrow{A, A, \leftarrow} q_0 \\ q_1 \xrightarrow{\square, A, \leftarrow} q_0 \\ q_1 \xrightarrow{A, A, \rightarrow} q_1 \end{array}$$



### exercice 2.4:



2. On construit une machine de Turing reconnaissant les mots qui ne sont pas dans L puis on crée un état final vers lequel on peut accéder depuis les états non terminaux de la première machine.

### Test 1

Utiliser la feuille elle-même pour répondre aux questions.

(13)  
20

Nom : Gonnaud

#### Exercice 1

L'ordre *hiérarchique* sur les mots de  $\{a, b, c\}^*$  est l'ordre du dictionnaire des mots croisés :  $u \leq v$  si soit  $|u| < |v|$ , soit  $|u| = |v|$  et  $u \leq_{lex} v$ . Ainsi :

$$\varepsilon < a < b < c < aa < ab < ac < ba < \dots$$

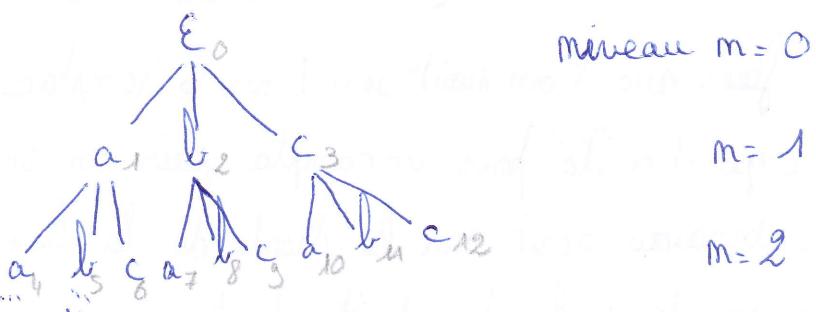
On définit  $\varphi(u)$  comme le nombre de mots  $v \in \{a, b, c\}^*$  tels que  $v < u$ . Autrement dit,  $\varphi(u)$  est le numéro de  $u$  selon l'ordre hiérarchique.

1- Exprimer  $\varphi(ua)$  en fonction de  $\varphi(u)$ . (Indication : évaluer le nombre de mots  $< ua$  suivant qu'ils sont de la forme  $\varepsilon$ ,  $va$ ,  $vb$  ou  $vc$ .)

2- Soient  $x_0, x_1, \dots, x_n$  des lettres de  $\{a, b, c\}$ . Donner une formule pour calculer  $\varphi(x_n x_{n-1} \dots x_0)$  en fonction des  $\varphi(x_k)$ .

1.  $\varphi(ua) = 3\varphi(u) + 1$  à prouver

$\frac{u}{10}$



des mots sont numérotés en effectuant un parcours en largeur sur cet arbre.

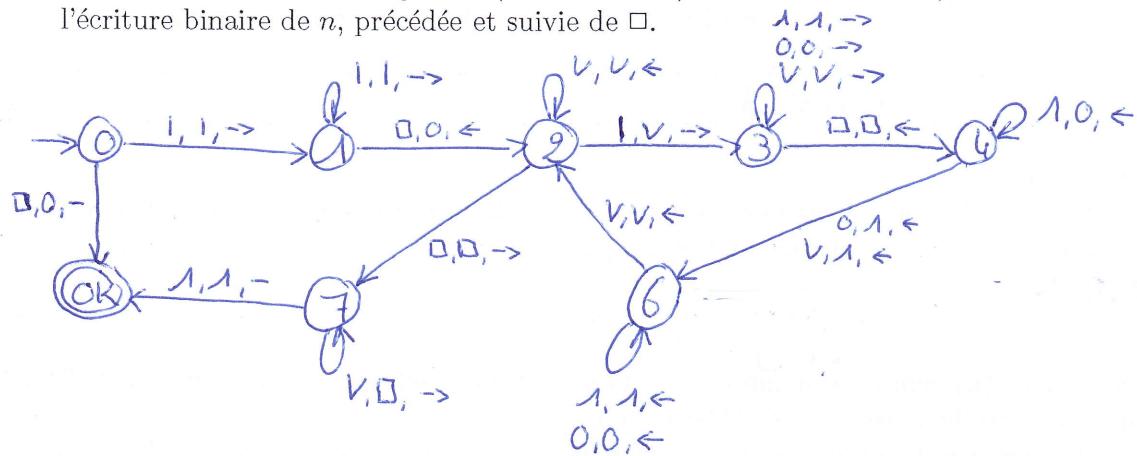
A chaque niveau  $m$ , il y a  $3^m$  éléments

2. Le mot  $x_m x_{m-1} \dots x_0$  est écrit au niveau  $m+1$  de l'arbre.

## Exercice 2

Donner une machine de Turing qui calcule la représentation en binaire d'un entier donné en représentation unaire.

On supposera qu'à l'origine la bande contient le mot  $\square a^n \square$ , la tête de lecture étant placée sur le symbole  $a$  le plus à gauche (ou  $\square$  si  $n = 0$ ). À la fin du travail, la bande doit contenir l'écriture binaire de  $n$ , précédée et suivie de  $\square$ .



?

On commence par écrire un 0 à la fin de la série de 1 représentant le nombre en binaire puis on remet la tête de lecture sur ce nombre (mais on sommes dans l'état 2).

9  
10

Chaque fois que l'on voit un 1 on le remplace par un V pour montrer qu'il a été pris en compte puis on incrémente le nombre binaire écrit sur la droite de la bande (état 4).

Une fois que tous les 1 ont été traités (on arrive au premier 0 à gauche du mot, état 7), on parcourt tous les V pour les effacer et on arrive dans l'état OK.

d'accord

Exercice 1

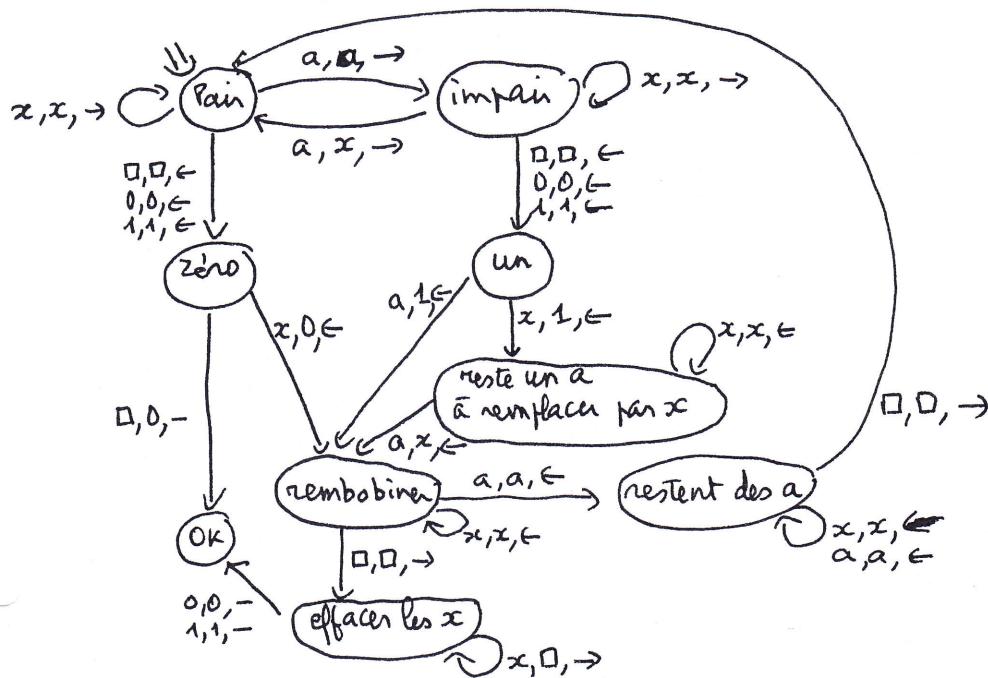
1) Soit  $u \in \{a, b, c\}^*$ . Les mots inférieurs à  $ua$  dans l'ordre hiérarchique sont d'une part  $\epsilon$ , d'autre part les mots de la forme  $vb$  ou  $vc$  avec  $v < u$ . On a donc  $\varphi(ua) = 3\varphi(u) + 1$ .

2) les mots inférieurs à  $ub$  sont  $ua$  et les mots inférieurs à  $uc$ , d'où  $\varphi(ub) = 3\varphi(u) + 2$  et de même  $\varphi(uc) = 3\varphi(u) + 3$ .

Le calcul récursif donné par les formules peut se faire itérativement :  $\varphi(ux) = 3\varphi(u) + \varphi(x)$  si  $x \in \{a, b, c\}$

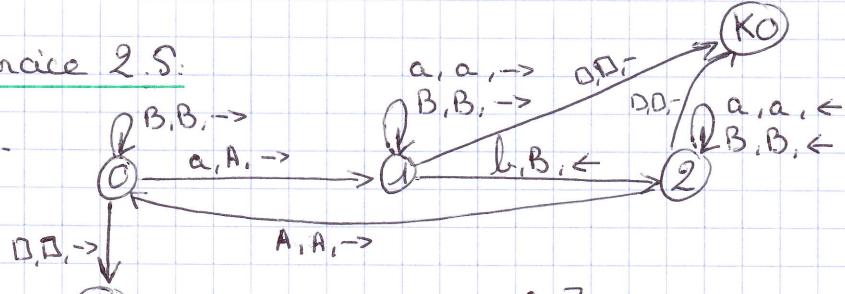
$$\varphi(x_n x_{n-1} \dots x_0) = \sum_{k=0}^n 3^k \varphi(x_k)$$
Exercice 2

Une version complète :



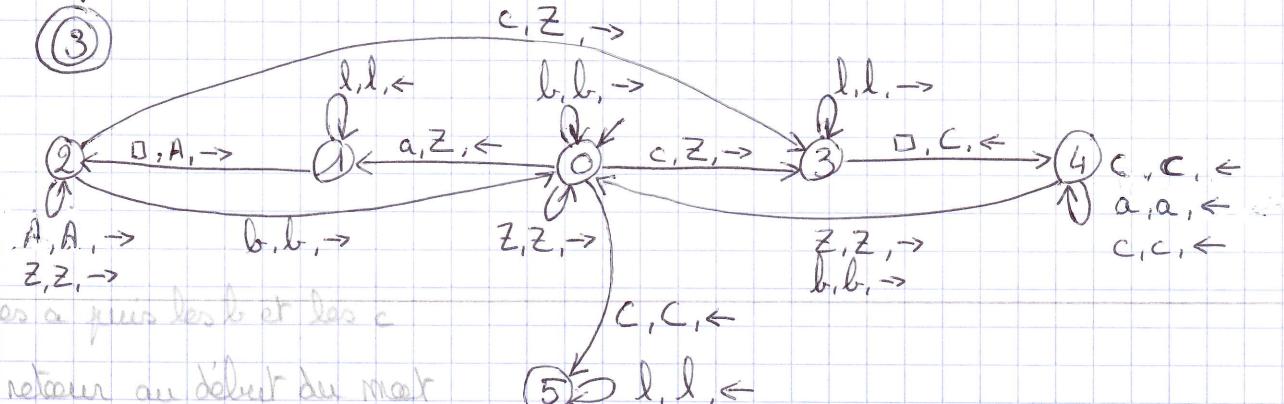
### exercice 2.5:

1.



++ rajouter l'état KO

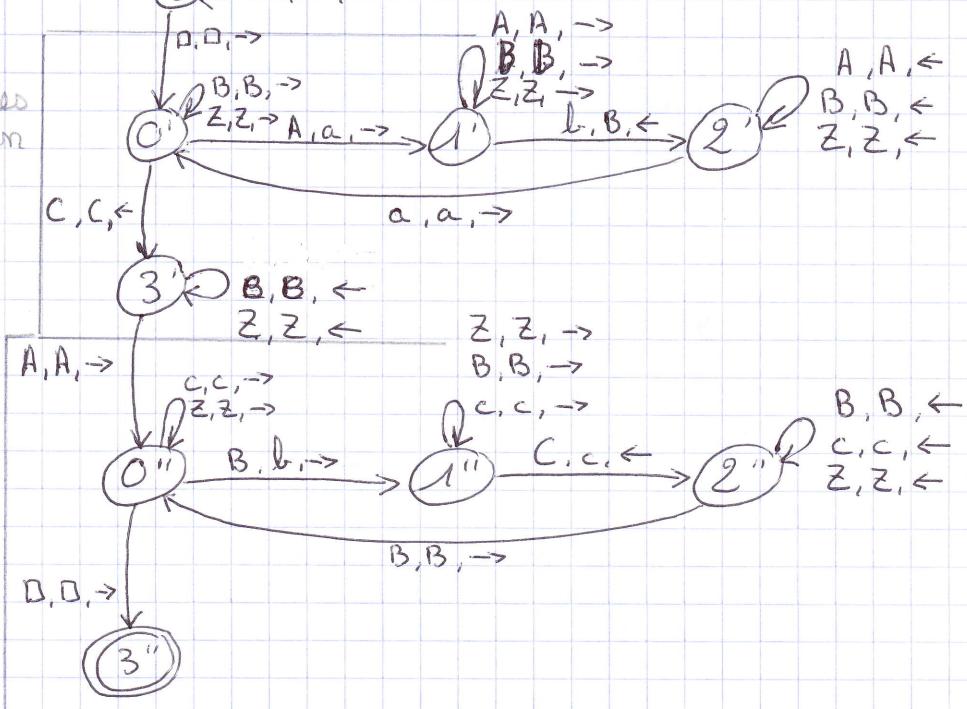
2.



retour au début du mot

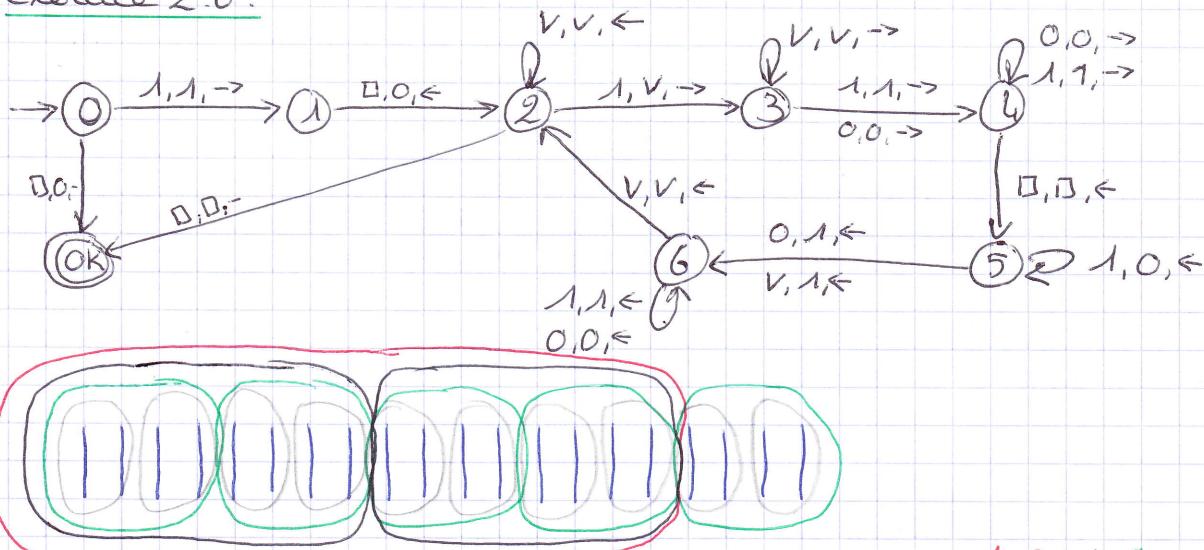
on compte les A et les B jusqu'à trouver un C : dans l'état 3'.  
 $|u|_a = |u|_b$

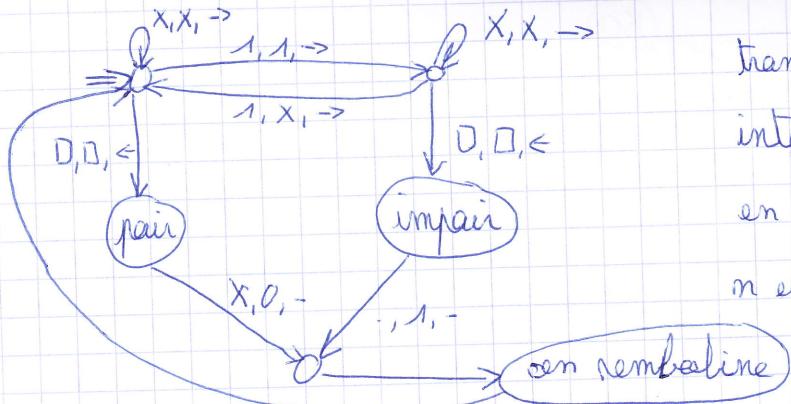
on compte les B et les C dans l'état 3".  
 $|u|_b = |u|_c$   
donc  $|u|_a = |u|_b = |u|_c$



l correspond à n'importe quelle lettre de Σ.

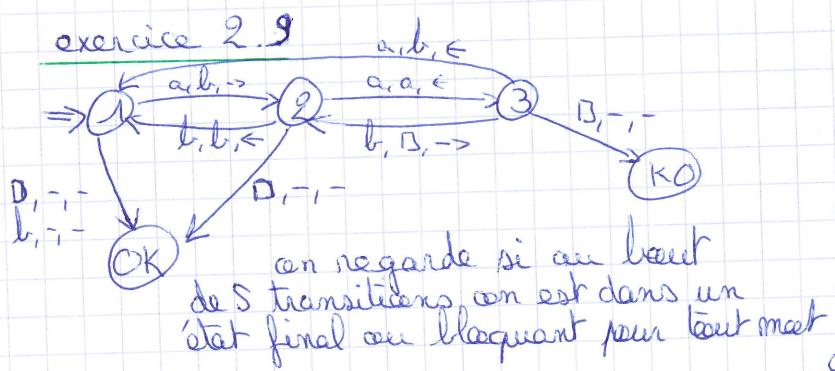
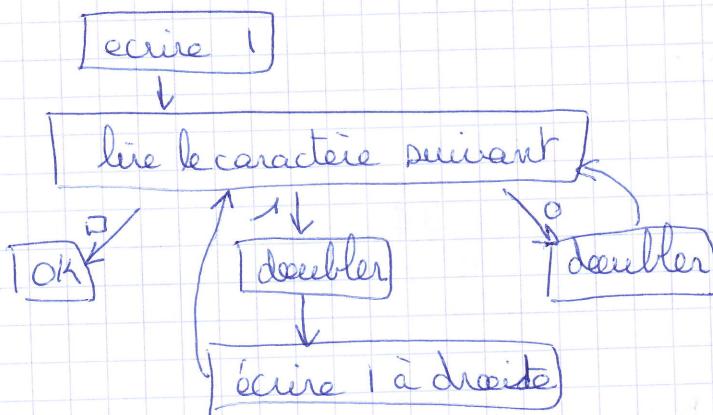
### exercice 2.6:





transforme  $m$  en  $m/2$  (avec des  $x$  intermédiaires) et le dernier caractère en 0 si  $m$  est pair et en 1 si  $m$  est impair.

$m$  en binaire  $\rightarrow m$  en ternaire



$b \dots$   
 $D \dots$   
 $aD \dots$   
 $ab \dots$   
 $aa \dots \rightarrow 5 \text{ pas}$

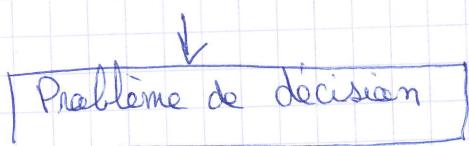
On fait tourner la machine sur tous les mots ( $9$  lettres suffisent)  $m^9$  mots  
 m = nombre de lettres de l'alphabet  
 nombre fini de cas  $\rightarrow$  problème décidable.  
 $\hookrightarrow$  algorithme polynomial sur la taille de l'alphabet de travail

exercice 2.10:

Données:  $M$  MT

$x$  mot

$m$  entier



$\left\{ \begin{array}{l} \text{Oui si } M \text{ en travaillant sur } x \text{ sort de } \\ \text{m cases} \\ \text{NON si } x \text{ reste dans } \end{array} \right.$



## Modèles de Calcul : feuille 2

Machines de Turing

### Exercice 2.1

Que calcule la machine de Turing suivante ? (les nombres sont écrits *de droite à gauche* ; au début du travail la tête de lecture est à gauche du mot).

	0	1	□
I	OK	$R, 0, \rightarrow$	OK
R	$OK, 1, -$	$R, 0, \rightarrow$	$OK, 1, -$

### Exercice 2.2

Écrire une machine de Turing qui calcule la fonction

$$n \longrightarrow \begin{cases} 0 & \text{si } n = 0 \\ n - 1 & \text{sinon} \end{cases}$$

les entiers étant codés en binaire, et écrits *de droite à gauche*.

### Exercice 2.3

Construire une machine de Turing dont la tête visite toutes les cases de la bande.

### Exercice 2.4

- Soit  $A = \{a, b, c, d\}$ . Construire une machine de Turing acceptant le langage  $L$  des mots  $u \in A^*$  ne contenant pas le facteur  $ad$ .
- De façon générale, étant donné un langage rationnel  $L$ , comment construire une machine de Turing reconnaissant les mots de  $L$  ?

### Exercice 2.5

- Construire une machine de Turing reconnaissant les mots sur  $\{a, b\}$  ayant le même nombre d'occurrences de  $a$  et de  $b$ .
- Construire une machine de Turing reconnaissant  $\{u \in \{a, b, c\}^* / |u|_a = |u|_b = |u|_c\}$ .

### **Exercice 2.6**

Donner une machine de Turing qui calcule la représentation en binaire d'un entier donné en représentation unaire.

### **Exercice 2.7**

Montrer que l'on peut coder toute machine de Turing par une autre machine de Turing qui n'utilise pour son alphabet de travail que les lettres 0, 1 et  $\square$ .

### **Exercice 2.8**

**Le problème de l'arrêt universel.** Montrer que le problème de déterminer si une machine de Turing s'arrête pour tout mot d'entrée est indécidable.

### **Exercice 2.9**

Le problème de déterminer si une machine de Turing s'arrête sur tout mot d'entrée après au plus 5 pas de calcul est-il décidable ? Justifier.

### **Exercice 2.10**

Montrer que le problème suivant est décidable : étant donné une machine de Turing  $M$ , un mot d'entrée  $x$ , et un entier  $n$ , déterminer si  $M$ , en travaillant sur  $x$ , sort du segment de bande contenant le mot  $x$ ,  $n$  cases à droite de  $x$  et  $n$  cases à gauche de  $x$ . On suppose qu'au début du travail la tête de lecture est au début du mot.

### **Exercice 2.11**

**Le problème de l'inversion de marche.** On considère le problème suivant : étant donnée une machine de Turing  $M$  qui se déplace (à gauche ou à droite) lors de chacune de ses transitions, et un mot d'entrée  $x$ , déterminer si  $M$ , en travaillant sur l'entrée  $x$ , se déplace toujours à droite ou au moins une fois vers la gauche. On suppose qu'au début du travail la tête de lecture est à gauche du mot. Ce problème est-il décidable ou indécidable ?

idée d'algo :

faire tourner  $M$  sur  $x$  et regarder si on sort de la bande  
ces : et si  $M$  ne s'arrête pas ?

à tout moment, on connaît

- la position de la tête de lecture
- l'état de la machine
- mot écrit sur le segment de bande

algo :

faire tourner  $M$  sur  $x$  en mémorisant à chaque étape la configuration

- si on sort du segment : Oui
- si on s'arrête avant d'être sorti : Non
- si on retombe sur une configuration déjà vue : Non.

Nombre de configurations possibles :  $(2n + |x|) \times |S_M| \times (\text{taille alpha})^{2n + |x|}$

positions de la tête  
de lecture

états possibles

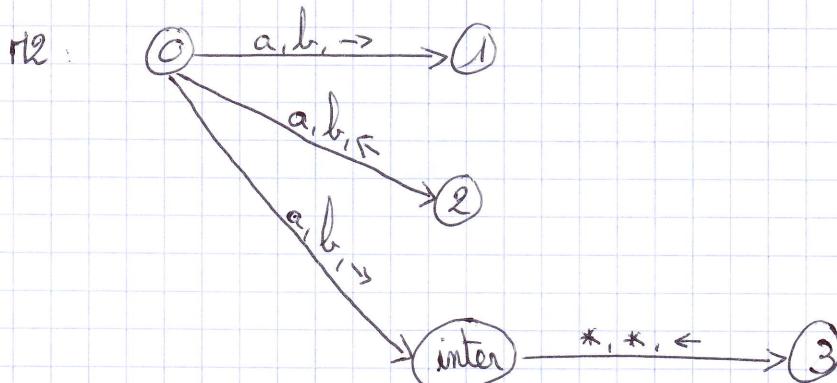
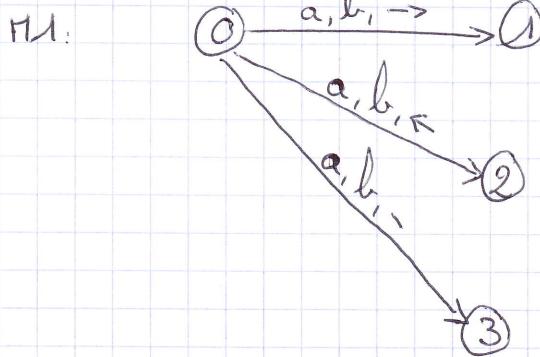
### exercice 2.7

Soient  $M_1$  une machine de Turing utilisant un alphabet de travail  
contenant tous les caractères ASCII et  $M_2$  une machine de Turing  
qui n'utilise pour son alphabet de travail que les lettres  $1, 0$  et  $\square$ .

On peut coder  $M_1$  par  $M_2$  en remplaçant chaque lettre de  $M_1$  par son  
code ASCII sur  $M_2$ . On sépare les lettres de  $M_1$  par un  $\square$  et on  
représente un  $\square$  de  $M_1$  par deux  $\square$  sur  $M_2$ .

(un codage est une injection)

### exercice 3.1



M2 simule M1 et sa tête de lecture se déplace à chaque transition.

### exercice 3.2:

soit  $S$  l'ensemble des transitions de la machine de Turing  $M$ :

$$S: \begin{array}{l} q_0 \xrightarrow{B, b, -} q_1 \\ q_0 \xrightarrow{a, b, -} q_0 \\ q_1 \xrightarrow{a, b, <} q_1 \end{array}$$

la machine RAM simulant la machine de Turing  $M'$ :

init: 0 load #1      ( $M'$  est la machine de Turing déterministe qui simule  $M$  mais qui n'utilise dans son alphabet de travail que les lettres 0, 1 et  $\square$ ). On considère ici qu'en 'a' est codé par un '0' et en 'b' par un '1'.)

1 store 0

2 read

2.1 if  $\square$  main

3 store (0)

4 load 0

5 incr

5.1 main: if load #1  
6 jump 2      8 store 0

7 : load (0)

decr

8 if  $q_0$

9 : write store (0)

jump  $q_1$

$q_0, a: \begin{array}{l} \text{load } \#1 \\ \text{write store } (0) \end{array}$

load 0

incr

jump  $q_0$

$q_1: \text{load } (0)$

$q_1, b: \text{if } q_1, a \text{ sortir les résultats}$

$q_1, a: \text{load } \#1$

write store (0)

load 0

decr

jump  $q_1$

### exercice 3.1:

M

états : S

règles : R

état initial :  $s_0$

états finaux : F

alphabet :  $\Sigma$



$s'$

pour chaque règle

$$s_1 \xrightarrow{x,y} s_2$$

de R

on ajoute un état  $s_n$

$$s_1 \xrightarrow{x,y} s_n$$

$$s_n \xrightarrow{z,z} s_2$$

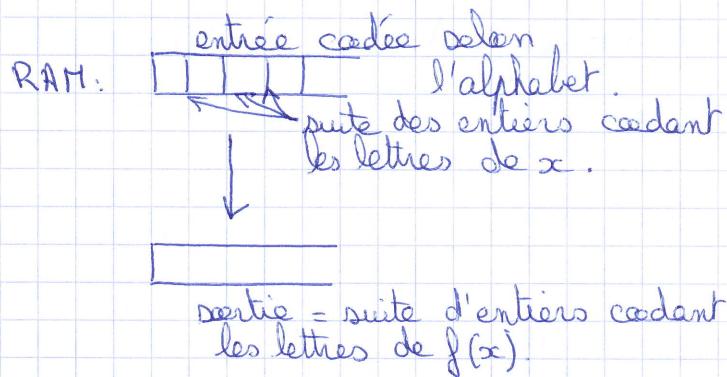
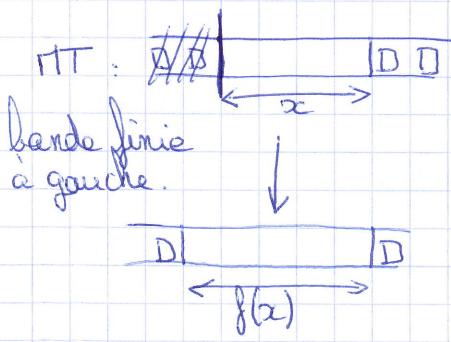
pour tout  $z \in \Sigma$ .

le temps de calcul est par plus multiplié par le nombre de lettres de l'alphabet.

### exercice 3.2:

Une machine RAM a des registres qui contiennent des entiers non limités  $\Leftrightarrow$  infinité de bandes infinies.

mais on est toujours dans une mémoire dénombrable.



$$D = 0$$

$$a = 1$$

$$b = 2$$

Stockez le mot d'entrée dans R1, R2, ... et la position de la tête de lecture dans RO. À la fin, écrive R1, ..., RM jusqu'à ce qu'on trouve un registre à zéro.

Dans le main : - une étiquette pour chaque état

- une étiquette pour chaque couple (état, lettre).

s'il n'y a pas de règle de la forme  $q^a \rightarrow q'$

$(q_0, a)$ : jump fin

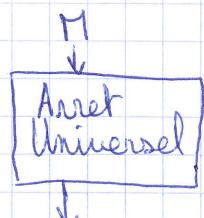
### exercice 2.8:



on admet que ce problème est indécidable.  
(cf. cours).

1 si  $M$  s'arrête sur  $x$

0 sinon

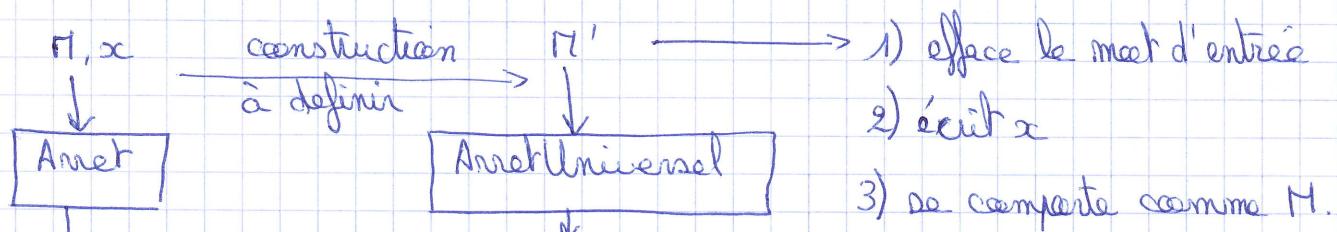


1 si  $M$  s'arrête sur tous les mots d'entrée

0 sinon

Réduction: "si je sauvais résoudre Arrêt Universel, je saurais résoudre Arrêt"

Réduction de Arrêt à Arrêt Universel



$M$  s'arrête sur  $x \Leftrightarrow M'$  s'arrête sur toute entrée

on suppose qu'on a une fonction int arrêtUniversel (int  $\neq f$ ) (int)

d'une fonction int g (int m)

```

int f (int x) {
    return g (m);
}
    
```

// fonction constante se comportant comme g(m)

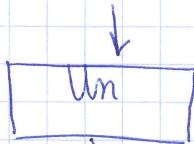
int main () {

```

}
    return arrêtUniversel (f); // renvoie 1 si g s'arrête sur m, 0 sinon
    
```

arrêt Existential :  $M'$  s'arrête sur au moins une entrée.

$M$  à 2 états OK et KO, xc



1, si  $M$  s'arrête sur xc dans l'état OK  
0, sinon

Un est-il décidable ou non ?

- conjecture

- si la réponse est OUI: donner un algo

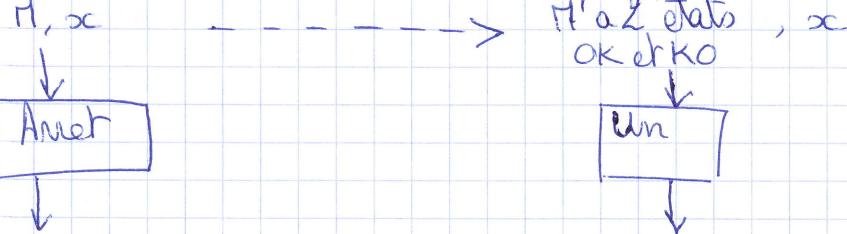
- si la réponse est NON: essayer de réduire ARRET à UN.

```

int f (int m) {
    if (m == 42)
        return OK;
    if (m == 0)
        return KO;
    while (1);
}

```

// décidable pour tout m  
si  $m = 42$  alors 1  
sinon 0.

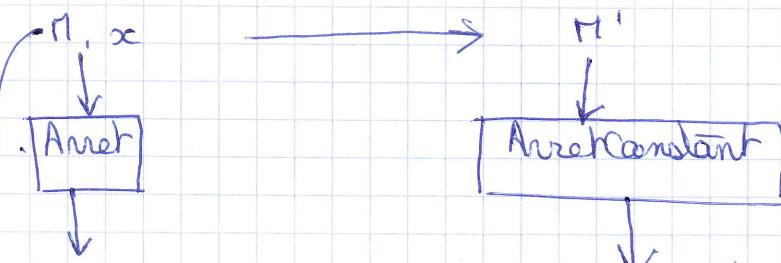


$M$  s'arrête sur xc  $\Leftrightarrow M'$  s'arrête sur xc dans l'état OK

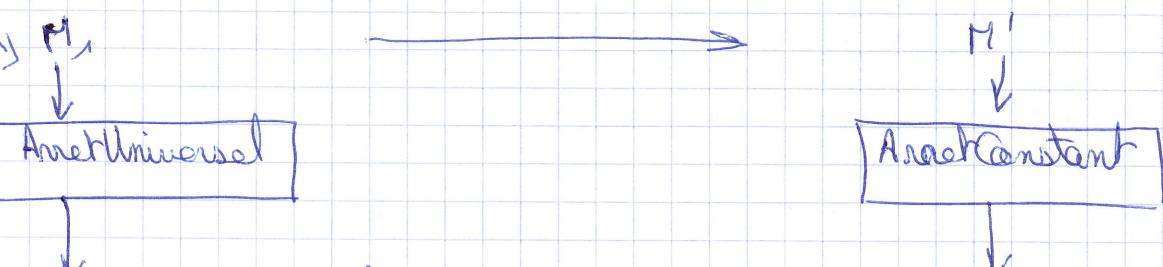
$M'$ :  $M$  augmenté de deux états OK et KO (s'ils existaient déjà, on les fusionne en un état mé-OK) et des transitions  $q \xrightarrow{a,a,-} \text{OK}$  si il n'existe pas dans  $M$   $q \xrightarrow{a,a,-}$  et  $\text{mé-OK} \xrightarrow{a,a,-}$  pour toute lettre  $a$ .  
Le problème d'Arrêt est indécidable donc Un est indécidable.



1 si  $M$  a le même comportement sur toutes les entrées (arrêt ou boucle), 0 sinon

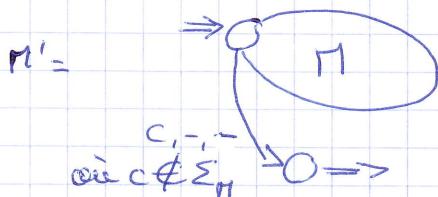


$M$  s'arrête sur  $x \Leftrightarrow M'$  s'arrête sur  $x$  et boucle sur toutes les entrées  
 $M$  boucle sur  $x \Leftrightarrow$  il existe un mot  $x_1$  sur lequel  $M'$  s'arrête et un mot  $x_2$  sur lequel  $M'$  boucle.



$M$  s'arrête sur tous les mots d'entrée  $\Leftrightarrow M'$  a le même comportement pour tous les mots

$M$  boucle sur un mot  $x \Leftrightarrow M'$  s'arrête sur  $x$  et boucle sur  $x_2$ .



$M' = M$  pour tous les mots différents de  $x_1 = c$ .

On peut résumer les réductions donc le problème de l'arrêt constant est indécidable.

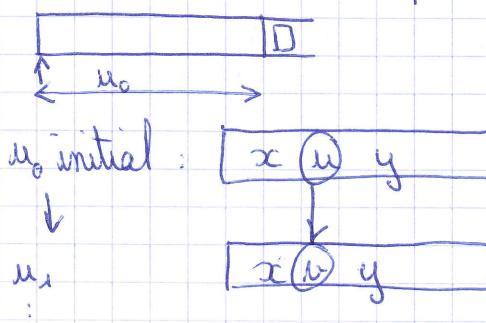
### exercice 3.3:

1. Toute transition d'une machine de Turing est de la forme :  $p \xrightarrow{a,b,d} q$ .

Dans un système de réécriture, on aurait :  $a \xrightarrow{\alpha} b + a$ .

grammaire algébrique = hors contexte

simulation d'un SRR par une MT non déterministe :



### exercice 3.4:

\* simulation d'une machine à piles par une MT:

idée 1: machine à  $k$  bandes représentant les piles

le sommet correspondant à la tête de lecture

chaque règle de la MP correspond à:

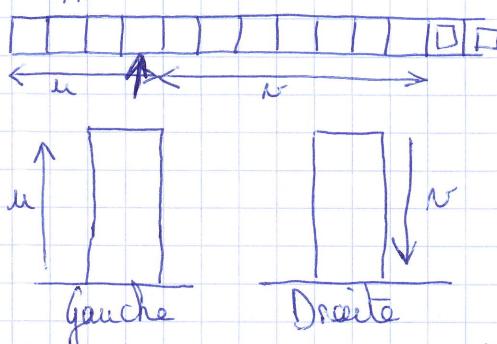
dépilement: écrire  $\square$  et déplacer la tête à gauche

empilement: écrire et déplacer la tête à droite

changer d'état

\* simulation d'une MT par une MP (deux piles):

on suppose une MT à bande finie à gauche



Règles de la machine à pile:

pour toute règle  $q \xrightarrow{a,b,l} q'$  de la MT:  $(q, a, x) \rightarrow (q', \epsilon, bx)$

$q \xrightarrow{a,b,r} q': (q, a, x) \rightarrow (q', bx, \epsilon)$  si  $x \neq \perp$ ;  $(q, a, \perp) \rightarrow (q', b\perp, \perp)$

$q \xrightarrow{a,b,l} q': (q, a, x) \rightarrow (q', l, x)$

pour toute lettre  $x$  ( $y$  compris  $\perp$ )

### exercice:

Ecrire une machine à 2 piles avec un état initial  $q_0$  et deux états d'arrêt

OK et KO telle que si au départ  $\boxed{\text{gauche}} \quad \boxed{\text{pile vide}} \quad \boxed{\text{droite}}$  alphabet:  $\{a, b\}$

alors la machine s'arrête dans OK si  $n \in \{a^m b^m \mid m \geq 0\}$ , dans KO sinon.

### exercice 3.3:

simulation d'un SRR par une MT non déterministe:

idée 1: MT parcourt le mot jusqu'à ce qu'elle reconnaisse un  $u$ :

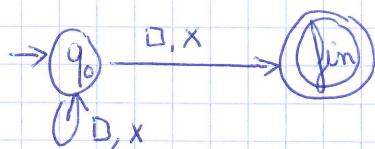
et le réécrit en  $v$ : (quitte à décaler ce qu'il y a à droite)

problème: la MT ne simule qu'un comportement possible du SRR.

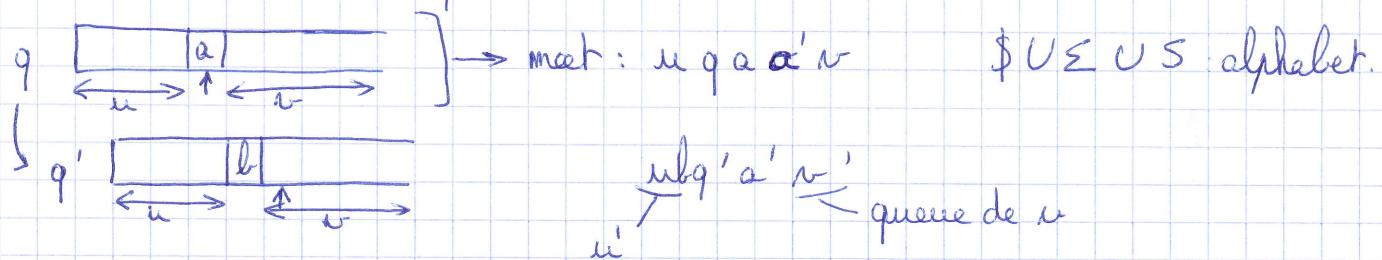
$a \rightarrow a$  la machine va reconnaître  $a b$  et le réécrit donc elle ne  
 $b \rightarrow ab$  fait jamais la seconde règle.

idéal: Ce qu'en veut: MT choisit de façon non déterministe un facteur  $u$ ; qu'elle réécrit en  $v$  et s'arrête s'il n'y a en pas.

écrire une MT qui sur la bande vierge écrit de façon non déterministe un certain nombre de  $x$  et s'arrête ou ne s'arrête pas.



simulation d'une MT par un SRR:



pour chaque règle de la MT: on crée dans le SRR pour toute lettre  $a' \in \Sigma$ :

$$\begin{array}{ll}
 q \xrightarrow{a, b, \square} q' & q a a' \rightarrow b q' a' \text{ et } q a \$ \rightarrow b q' \square \$ \\
 q \xrightarrow{a, b, \square} q' & q a \rightarrow q' b \\
 q \xrightarrow{a, b, \square} q' & a' q a \rightarrow q' a' b
 \end{array}$$

### exercice 3.5:

donnée: une grammaire algébrique:  $S$ : symbole initial

$\Sigma_N$ : non terminaux ( $S \in \Sigma_N$ )

$\Sigma_T$ : terminaux

P: ensemble de règles de production

de la forme:  $N \xrightarrow{\epsilon} u \in (\Sigma_N \cup \Sigma_T)^*$

grammaire ambiguë: on peut construire plusieurs arbres pour un même mot en résultat (ex:  $E = \text{nombre} \mid E + E \mid E \times E$ )

$\rightarrow$  Oui s'il existe un mot de  $\Sigma_T^*$  avec 2 arbres différents, NON sinon.

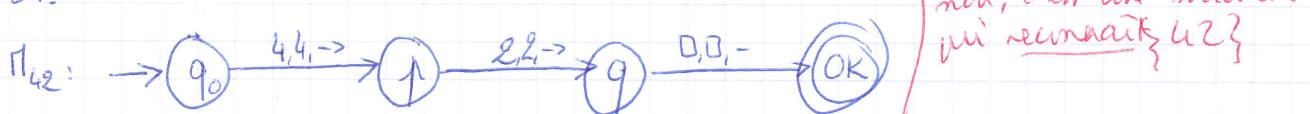
GARNAUD Eve groupe 3:

le 03. 11.08

## Dévoir 2 MC

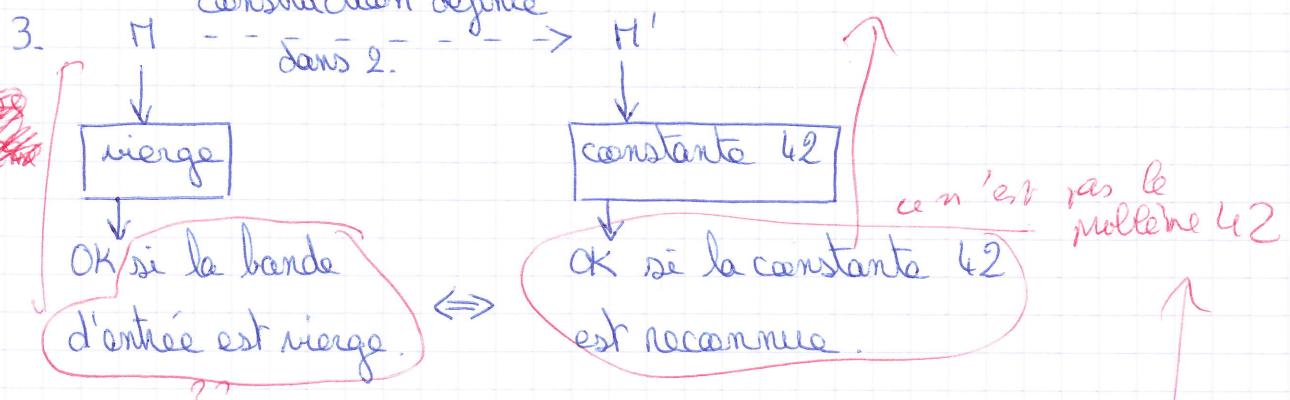
### Exercice 1 "Problème 42" non compris

1.

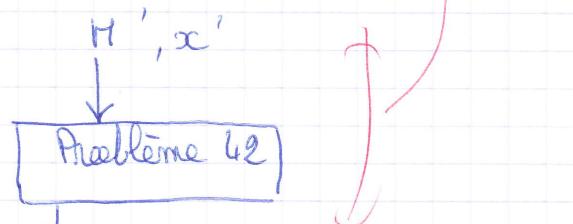
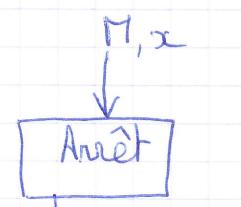


2.  $M'$  commence par calculer la constante 42. Nous arrivons dans un état pré-OK. Il faut ensuite effacer la bande et faire tourner  $M$  sur cette nouvelle entrée. Si 42 n'est pas reconnue initialement, on n'efface pas la bande donc  $M$  ne s'arrête pas.  $M'$  s'arrête si  $M$  s'arrête. Cela ne répond pas à la question posée.

3. construction définie



4. On peut réduire le problème 42 au problème de l'Arrêt sur toute entrée  $x$



On sait que le problème de l'Arrêt est indécidable donc le problème 42 est indécidable.

Exercice 2:

1.  $L(G_0) = (b_i u_i)^+$

cela correspondrait plutôt à  
 $(b_1 u_1)^+ \cup (b_2 u_2)^+ \cup \dots \cup (b_p u_p)^+$

avec  $1 \leq i \leq p$  pas clair

2.  $G_0$  n'est pas ambiguë car, pour un mot  $w \in L(G_0)$  donné, on ne peut construire qu'un seul arbre de dérivation. Par contre?

3.  $L(G_1) = (b_i u_i)^+ + (b_i v_i)^+$  avec  $1 \leq i \leq p$

4. S'il existe un mot sur  $A$  admettant 2 décompositions distinctes alors  $G$  est ambiguë car on peut choisir de dériver  $S$  en  $U$  ou  $V$  sans que cela ne change le mot.

5.

$v_u$

## Test 2

### Exercice 1

On dit qu'une machine de Turing travaillant sur l'alphabet ASCII calcule la constante 42 si pour tout mot d'entrée, elle s'arrête en ayant sur sa bande le mot □42□.

1. Décrire une machine de Turing  $M_{42}$  qui calcule la constante 42.
2. Décrire un algorithme qui, à partir d'une machine de Turing  $M$ , construit une machine de Turing  $M'$  telle que  $M$  s'arrête sur la bande vierge si et seulement si  $M'$  calcule la constante 42.

Le problème 42 est le suivant :

**Donnée** Une machine de Turing.

**Question** Cette machine de Turing calcule-t-elle la constante 42 ?

3. Exprimer précisément le résultat de la question 2 en terme de réduction.
4. Montrer que le problème 42 est indécidable.

### Exercice 2

Soient  $u_1, u_2, \dots, u_p$  des mots sur un alphabet  $A$ .

Soient  $U, b_1, b_2, \dots, b_p$  des symboles n'appartenant pas à  $A$ . Soit  $G_0$  la grammaire à un seul symbole non-terminal ~~et~~  $U$  (initial), dont l'alphabet terminal est  $A \cup \{b_1, b_2, \dots, b_p\}$ , et dont les productions sont les  $U \rightarrow b_i u_i U$  et les  $U \rightarrow b_i u_i$ , pour  $1 \leq i \leq p$ .

1. Quel est le langage engendré par  $G_0$  ?
2. La grammaire  $G_0$  est-elle ambiguë ?

Soient maintenant  $v_1, v_2, \dots, v_p$  des mots sur  $A$ . Soit  $G$  la grammaire obtenue en ajoutant à  $G_0$  deux nouveaux symboles non terminaux  $S$  (initial) et  $V$ , ainsi que les productions  $S \rightarrow U$ ,  $S \rightarrow V$ , et les  $V \rightarrow b_i v_i V$  et  $V \rightarrow b_i v_i$ , pour  $1 \leq i \leq p$ .

3. Quel est le langage engendré par  $G$  ?
4. Montrer que s'il existe un mot sur  $A$  admettant deux décompositions distinctes

$$u_{i_1} u_{i_2} \cdots u_{i_n} = v_{i_1} v_{i_2} \cdots v_{i_n}$$

alors la grammaire  $G$  est ambiguë.

5. Montrer que, si la grammaire  $G$  est ambiguë, il existe un mot sur  $A$  admettant deux décompositions distinctes  $u_{i_1} u_{i_2} \cdots u_{i_n} = v_{i_1} v_{i_2} \cdots v_{i_n}$ .
6. En déduire que l'on ne peut pas décider si une grammaire algébrique est ambiguë.

## *Correction* Test 2

### Exercice 1

- Une machine de Turing  $M_{42}$  calculant la constante 42 : il suffit d'une machine à 5 états  $s_0$  (initial),  $s_1, s_2, s_3$  et  $s_4$  (terminal), et, pour tout caractère  $x$ , les règles  $s_0 \xrightarrow{x:\square,\rightarrow} s_1$ ,  $s_1 \xrightarrow{x:4,\rightarrow} s_2$ ,  $s_2 \xrightarrow{x:2,\rightarrow} s_3$  et  $s_3 \xrightarrow{x:\square,-} s_4$ .
- Soit  $M' = M; M_{42}$ . Si  $M$  s'arrête sur l'entrée  $u$ , alors  $M'$  calcule la constante 42. Si  $M$  ne s'arrête pas sur l'entrée  $u$ ,  $M'$  ne s'arrête pas non plus, donc ne répond pas à la définition des machines calculant la constante 42.
- La construction de  $M'$  à partir de  $M$  est une réduction du problème d'arrêt sur la bande vierge (équivalant au problème d'acceptation du mot vide) au problème 42.
- Comme le problème d'arrêt sur la bande vierge est indécidable, le problème 42 est indécidable.

### Exercice 2

- Le langage engendré par  $G_0$  est  $\{b_1u_1, b_2u_2, \dots, b_pu_p\}^+$  (ensemble des mots non vides qui se décomposent en facteurs  $b_iu_i$ ).
- L'arbre syntaxique associé à un mot  $b_{i_1}u_{i_1}\cdots b_{i_{n-1}}u_{i_{n-1}}b_{i_n}u_{i_n}$  est unique : en effet, chaque  $b_{i_k}$  provient nécessairement d'une application de la règle  $U \rightarrow b_{i_k}u_{i_k}U$  si  $k < n$ , et de la règle  $U \rightarrow b_{i_n}u_{i_n}$  si  $k = n$ . La grammaire  $G_0$  n'est donc pas ambiguë.
- Soit  $G_1$  la grammaire définie comme  $G_0$ , mais à partir des mots  $v_1, v_2, \dots, v_p$ . Les dérivations de  $S$  suivant  $G$  sont les dérivations commençant par  $S \rightarrow U$  et se prolongeant par une dérivation de  $U$  suivant  $G_0$ , et les dérivations commençant par  $S \rightarrow V$  et se prolongeant par une dérivation de  $V$  suivant  $G_1$ . Le langage engendré par  $G$  est donc la réunion des langages engendrés par  $G_0$  et  $G_1$ , c'est-à-dire  $\{b_1u_1, b_2u_2, \dots, b_pu_p\}^+ \cup \{b_1v_1, b_2v_2, \dots, b_pv_p\}^+$ .
- Supposons qu'il existe un mot sur  $A$  admettant deux décompositions distinctes

$$u_{i_1}u_{i_2}\cdots u_{i_n} = v_{i_1}v_{i_2}\cdots v_{i_n}$$

Alors le mot  $b_{i_1}u_{i_1}b_{i_2}u_{i_2}\cdots b_{i_n}u_{i_n} = b_{i_1}v_{i_1}b_{i_2}v_{i_2}\cdots b_{i_n}v_{i_n}$  est dans chacun des langages engendrés par  $G_0$  et  $G_1$ , il peut donc être produit par une dérivation commençant par  $S \rightarrow U$  et par une dérivation commençant par  $S \rightarrow V$ . Cela conduit à deux arbres syntaxiques distincts : la grammaire  $G$  est donc ambiguë.

5. Supposons la grammaire  $G$  ambiguë. Il existe donc un mot  $w$  engendré par  $G$  correspondant à deux arbres syntaxiques distincts. Comme ni  $G_0$  ni  $G_1$  ne sont ambiguës,  $w$  est nécessairement à la fois dans  $L(G_0)$  et dans  $L(G_1)$ .

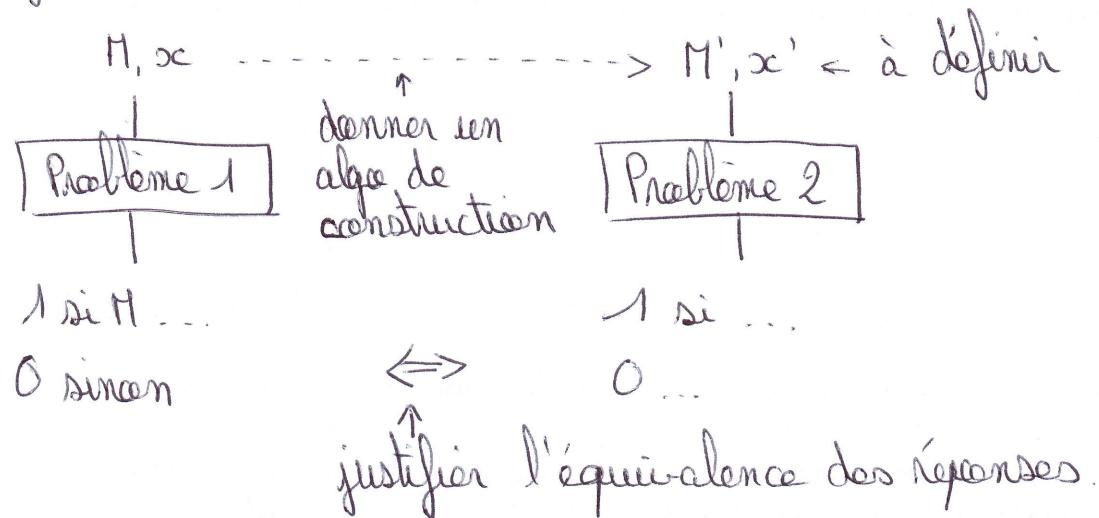
$w$  peut donc se décomposer en produit de facteurs  $b_i u_i$  et en produit de facteurs  $b_j v_j$ ; la suite des lettres  $b_i$  doit être la même dans les deux décompositions, on a donc une identité de la forme

$$w = b_{i_1} u_{i_1} b_{i_2} u_{i_2} \cdots b_{i_n} u_{i_n} = b_{i_1} v_{i_1} b_{i_2} v_{i_2} \cdots b_{i_n} v_{i_n}$$

d'où l'on déduit  $u_{i_1} u_{i_2} \cdots u_{i_n} = v_{i_1} v_{i_2} \cdots v_{i_n}$ .

6. À toute instance  $(u_1, v_1)(u_2, v_2) \cdots (u_n, v_n)$  du problème de correspondance de Post, on peut associer une grammaire  $G$  qui est ambiguë si et seulement si PCP a une solution. PCP étant indécidable, on ne peut donc pas décider si une grammaire algébrique est ambiguë.

Faire une réduction du problème 1 au problème 2:



## Modèles de Calcul : feuille 3

modèles alternatifs aux machines de Turing et indécidabilité.

### Exercice 3.1

Montrer que toute machine de Turing peut être simulée par une autre machine de Turing dont la tête se déplace à chaque transition.

### Exercice 3.2

Montrer que toute machine de Turing peut être simulée par une machine RAM.

### Exercice 3.3 Systèmes de règles de réécriture

Étant donné un alphabet fini  $\Sigma$ , on appelle *système de réécriture sur  $\Sigma$*  un sous-ensemble fini de  $\Sigma^+ \times \Sigma^*$ . Si  $R$  est un système de réécriture sur  $\Sigma$  on dit que  $R$  réécrit la chaîne  $s_1$  en  $s_2$  (ce que l'on note  $s_1 \rightarrow_R s_2$ ) si il existe  $(w_1, w_2)$  dans  $R$  tel que  $s_1 = v_1 w_1 v_2$  et  $s_2 = v_1 w_2 v_2$ . On dit que la chaîne  $s_r$  est le résultat du calcul de  $R$  à partir de  $s_i$  si il existe une suite finie  $(s_k)_{0 \leq k \leq n}$  telle que  $s_0 = s_i$ ,  $s_r = s_n$ , pour tout  $k < n$ ,  $s_k \rightarrow_R s_{k+1}$  et il n'existe pas de  $t$  tel que  $s_r \rightarrow_R t$ .

1. Simuler une machine de Turing à l'aide d'un système de réécriture.
2. Simuler un système de réécriture à l'aide d'une machine de Turing non-déterministe.

*GR: ensemble fini de règles u (pas non vide)  $\rightarrow$  on choisit une règle de façon non-déterministe*

*(pas nécessaire)*

### Exercice 3.4 machines à k-piles

Une machine à  $k$  piles est définie par un triplet  $(Q, \Sigma, \delta)$  où  $Q$  est un ensemble fini d'états,  $\Sigma$  est un alphabet fini et  $\delta$  est un sous-ensemble fini de  $(Q \times \Sigma \cup \{\perp\}) \times (Q \times (\Sigma^*)^k)$  avec  $\perp \notin \Sigma$ . La configuration d'une telle machine est donnée par  $(q, w_1, \dots, w_k)$  avec  $q \in Q$ , et  $w_i \in \Sigma^* \perp$  pour tout  $i$  dans  $[1; k]$ . Elle peut passer d'une configuration  $(q_1, w_1^1, \dots, w_k^1)$  à une configuration  $(q_2, w_1^2, \dots, w_k^2)$  si  $w_j^1 = X_j v_j^1$ ,  $w_j^2 = u_j v_j^2$  et  $((q_1, (X_1, \dots, X_k)), (q_2, (u_1, \dots, u_k))) \in \delta^1$ .

1. Montrer que toute machine de Turing peut être simulée par une machine à deux piles.
2. Montrer que toute machine à  $k$  piles peut être simulée par une machine de Turing non-déterministe.

<sup>1</sup>on suppose que si  $X_k = \perp$  alors  $u_k$  appartient à  $\Sigma^* \perp$ .

$x_1 \quad x_2 \quad x_3 \quad x = \perp \text{ si les piles sont vides}$

machine à 3 piles : alphabet  $\Sigma$  / ensemble d'états  $S$

Règles de la forme :  $q, x_1, x_2, x_3 \rightarrow q', u_1, u_2, u_3$

on décale  $x_1$  et on empile  $u_1$  à la place

### Exercice 3.5

Ambiguïté d'une grammaire. Montrer que l'on ne peut pas décider si une grammaire algébrique est ambiguë, en réduisant le problème de Post à ce problème.

### Exercice 3.6

Intersection de deux grammaires. Montrer que l'on ne peut pas décider si les langages engendrés par deux grammaires algébriques données ont une intersection non-vide.

### Exercice 3.7

Problème de correspondance de Post modifié. On veut montrer que le problème suivant (dit Problème de correspondance de Post modifié) est indécidable : la donnée est une suite de  $n$  couples de mots  $((u_1, v_1), \dots, (u_n, v_n))$  sur un alphabet  $A$ . La question est : existe-t-il un mot  $w$  sur  $A$  qui admet deux compositions distinctes avec pour premier élément le couple  $(u_1, v_1)$  :  $w = u_1 u_{i_2} \dots u_{i_k} = v_1 v_{i_2} \dots v_{i_k}$  ?

Pour montrer que ce problème est indécidable, soit  $\mathcal{P} = ((u_1, v_1), \dots, (u_n, v_n))$  une instance du problème de Post modifié sur un alphabet  $A$  et  $\$$  une lettre n'appartenant pas à  $A$ . Définir une instance  $\mathcal{Q}$  du problème de Post telle que  $\mathcal{P}$  admet une solution  $w = x_1 x_2 \dots x_m$  si et seulement si  $\mathcal{Q}$  admet comme solution  $\$x_1 \$x_2 \$ \dots \$x_m \$$ .

**Indication :** Soit un mot  $u = a_1 \dots a_n$ , on notera  $u\$$  le mot  $a_1 \$ a_2 \$ \dots \$ a_n$ . Le couple de l'instance de  $\mathcal{Q}$  à construire qui permettra d'obliger la solution de  $\mathcal{P}$  à commencer par  $(u_1, v_1)$  sera  $(\$u_1\$, \$v_1\$\$)$ . En plus de ce couple, il faudra ajouter à l'instance de  $\mathcal{Q}$  deux couples pour chaque couple  $(u_i, v_i)$ , choisis pour forcer une solution de  $\mathcal{Q}$  à commencer par  $(\$u_i\$, \$v_i\$\$)$ .

### Exercice 3.8 ~~Machines à compteur~~

Une machine à  $k$  compteurs est une machine qui utilise  $k$  registres  $r_1, \dots, r_k$  qui contiennent chacun un entier. Un programme pour une telle machine est une série d'instructions numérotées de la forme :

1.  $n : \text{inc}(r_i); \text{goto } m$
2.  $n : \text{dec}(r_i); \text{goto } m$
3.  $n : \text{if } r_i = 0 \text{ then goto } m_1 \text{ else goto } m_2$
4.  $n : \text{stop}$

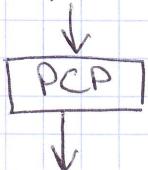
où  $n, m, m_1$  et  $m_2$  dénotent des numéros d'instruction. On suppose également que si  $\text{dec}(r_i)$  est exécutée alors que  $r_i$  contient zéro, la valeur de  $r_i$  reste zéro. La configuration d'une machine à  $k$  compteurs est donnée par un tuple  $(n, p_1, \dots, p_k)$  où  $n$  est le numéro de l'instruction courante et  $p_i$  est le nombre contenu dans le registre  $r_i$ . La transition d'une configuration à une autre s'effectue en exécutant l'instruction courante est en prenant comme nouvelle instruction courante celle qui est donnée par le **goto** associé.

1. Montrer qu'une machine à deux compteurs peut simuler une machine à une pile.
2. En déduire qu'une machine à quatre compteurs peut simuler une machine à deux piles.
3. Montrer qu'une machine à  $n$  compteurs peut être simulée par une machine à deux compteurs.

On peut démontrer que si une machine à deux compteurs contient  $n$  dans l'un de ces registres et 0 dans l'autre alors il n'est pas possible qu'elle s'arrête avec un registre contenant  $2^n$  et l'autre 0. Expliquer pourquoi ce théorème n'est pas en contradiction avec les résultats obtenus plus haut.

Réduction du problème de Post au problème d'ambiguïté : PCP  $\leq$  Ambiguïté

$m, m$  paires de mots

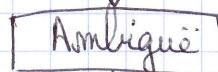


OK si on a  $i_1, \dots, i_k$

telle que  $u_{i_1} \dots u_{i_k} = v_{i_1} \dots v_{i_k}$

KO sinon

$m', g$

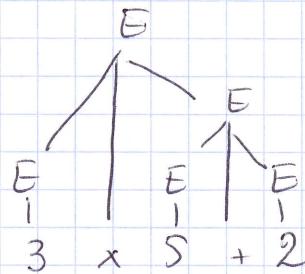
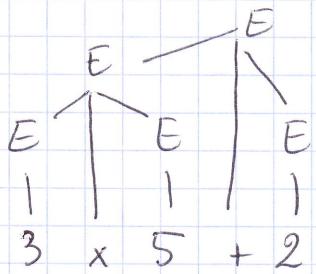


OK s'il existe un mot  $w \in \Sigma^*$

avec deux arbres différents

KO sinon

$w = u_{i_1} \dots u_{i_k}$ , il est obtenu de deux façons différentes.



$E \rightarrow$  membre

$E \rightarrow E + E$

$E \rightarrow E \times E$

Soyons les paires :  $\boxed{3 \times 5} \quad \boxed{+ 2}$  on décrit le même mot  $3 \times 5 + 2$   
 $\boxed{3 \times} \quad \boxed{5 + 2}$  de deux manières.

PCP indécidable  $\Rightarrow$  Ambiguïté indécidable.

exercice 3.6 :

Gm fait une réduction : PCP  $\leq$  Intersection.

Gm construit  $G_1$  et  $G_2$  telles que la réponse au PCP sur  $(u_1, v_1) \dots (u_m, v_m)$  est oui  $\Leftrightarrow L(G_1) \cap L(G_2) \neq \emptyset$ .

$G_1$  et  $G_2$  travaillent sur l'alphabet  $A \cup \{\#, 1, \dots, m\}$  où  $A$  est l'alphabet sur lequel  $u_i$  et  $v_i$  sont écrits.

$$G_1: S \rightarrow 1S u_1 | 2S u_2 | \dots | mS u_m | 1\# u_1 | 2\# u_2 | \dots | m\# u_m$$

$$G_2: S \rightarrow 1S v_1 | 2S v_2 | \dots | mS v_m | 1\# v_1 | 2\# v_2 | \dots | m\# v_m.$$

$$L(G_1) \cap L(G_2) = i_1 i_2 \dots i_m \# u_m \dots u_2 u_1 = i_1 i_2 \dots i_m \# v_m \dots v_2 v_1$$

Suites de Syracuse :

$$u_{m+1} = \begin{cases} u_m + 2 & \text{si } u_m \text{ est pair} \\ 3u_m + 1 & \text{si } u_m \text{ est impair, } u_m \neq 1 \end{cases}$$

ça s'arrête si  $u_m = 1$ .

## exercice 4.2

1. langage des mots codant les machines :  $L = (R \#)^*$

avec  $R = S \$ \{0, 1, \square\} \$ \{0, 1, \square\} \$ \{0, 1, \square\} \$ S$  et  $S = \{0\} \cup \{1, 0\}^*$

2.  $0 < 1 < \square < \$ < \#$

$o_0 \Rightarrow 0$  code : E

$o_1 \Rightarrow 0 \underset{0,0,-}{\overrightarrow{\#}} 0$  code : 0\$0\$0\$0\$0\$0#

$o_2 \Rightarrow 0 \underset{0,0,-}{\overrightarrow{\#}} 1$  code : 0\$0\$0\$0\$0\$1#

on a  $2 \times 3 \times 3 \times 3 \times 2 = 108$  codes de cette longueur.

$o_4 \Rightarrow 0 \underset{0,0,-}{\overrightarrow{\#}} 0$  code : 0\$0\$0\$0\$0\$0\$0#

3.  $\underset{0,0,-}{\overbrace{0 \rightarrow 0}} \xrightarrow{1,1,-} 0$  0\$0\$0\$0\$0\$0#0\$1\$1\$1\$0\$1#  
longueur 20

on a un algorithme pour trouver la machine connaissant son numéro  
trouver le numéro d'une machine donnée.

→ énumération effective des machines de Turing.

par convention : OK = état 1

4.  $P_m(x) = \begin{cases} 1 & \text{si } o_m \text{ en travaillant sur la représentation binaire de } x, \text{ s'arrête en OK} \\ 0 & \text{si } o_m \text{ non défini sinon } (\uparrow) \end{cases}$

$P_m : \mathbb{N} \rightarrow \{0, 1\}$  prédictor semi-calculable.

→ énumération effective des prédictors semi-calculables (avec répétitions)

$Q(m, x) = \begin{cases} 1 & \text{si } P_m(x) = 1 \\ 0 & \text{si } P_m(x) \text{ n'est pas défini} \end{cases}$

x	0	1	2	3	4	...
m	↑	↑	↑	↑	↑	
$P_0$	↑	↑	↑	↑	↑	
$P_1$	↑	↑	↑	↑	↑	
$P_2$	1	↑	↑	↑	↑	

$f(m) = \overline{Q(m, m)}$  on remplace les éléments de la diagonale par leur complémentaire.

## Modèles de Calcul : feuille 4

Diagonalisations - Théorème de Rice

### Exercice 4.1

- Pour tout entier  $n \geq 0$ , soit  $f(n)$  la valeur décimale approchée par défaut à  $10^{-n}$  près de  $\pi$  : ainsi,  $f(0) = 3$ ;  $f(1) = 3,1$ ;  $f(2) = 3,14\dots$ . La fonction  $f$  est-elle calculable (au sens intuitif du terme) ?
- Que calcule la fonction suivante ?

```

int syracusain(unsigned int n) {
    while (n > 1)
        n = (n%2 == 0)? n/2 : 3*n + 1;
    return 1;
}

```

### Exercice 4.2

#### Énumération effective des prédictats semi-calculables de $\mathbb{N}$ .

On considère l'ensemble  $\mathcal{M}$  des machines de Turing à états numérotés qui travaillent sur l'alphabet  $\Sigma = \{0, 1\}$ . Par convention, l'état 0 est l'état initial. Chaque machine de  $\mathcal{M}$  est définie par la liste  $[r_1, r_2, \dots, r_n]$  de ses règles.

On code chaque règle  $s \xrightarrow{a:b,\leftarrow} s'$  par le mot  $n_s \$ a \$ b \$ d \$ n_{s'}$  où  $n_s$  et  $n_{s'}$  sont les écritures binaires des numéros des états  $s$  et  $s'$ , et  $d$  est le codage du déplacement : 0 pour  $\rightarrow$ , 1 pour  $\leftarrow$ ,  $\square$  pour un déplacement nul.

On code une machine  $[r_1, r_2, \dots, r_n]$  de  $\mathcal{M}$  par le mot  $u_1 \# u_2 \# \dots \# u_n \#$  où chaque  $u_i$  est le mot codant la règle  $r_i$ .

- Donner une expression rationnelle du langage  $L$  des mots codant les machines de  $\mathcal{M}$ .
- On ordonne l'alphabet  $\{0, 1, \square, \$, \#\}$  par  $0 < 1 < \square < \$ < \#$ , et on numérote les mots de  $L$  dans l'ordre hiérarchique. Déterminer les machines de  $\mathcal{M}$  codées par les mots de numéros 0; 1; 2; 42. Plus généralement, comment trouver la machine  $M_n$  codée par le mot numéro  $n$  de  $L$  ?
- Soit  $M$  la machine de Turing définie par les règles  $[0 \xrightarrow{0:0,\rightarrow} 1, 0 \xrightarrow{\square:0,\rightarrow} 0]$ . Quel est le numéro dans  $L$  du mot codant  $M$ ? Plus généralement, comment trouver le numéro du mot codant une machine donnée de  $\mathcal{M}$  ?
- On définit la fonction  $P_n : \mathbb{N} \longrightarrow \{1\}$  par  $P_n(x) = 1$  si la machine  $M_n$ , en travaillant sur l'écriture binaire de  $x$ , s'arrête dans l'état OK ; sinon,  $P_n(x)$  n'est pas défini. Démontrer, en utilisant l'idée de diagonalisation, que le prédictat à deux arguments

$$Q(n, x) = \begin{cases} 1 & \text{si } P_n(x) = 1 \\ 0 & \text{si } P_n(x) \text{ n'est pas défini} \end{cases}$$

n'est pas calculable.

### Exercice 4.3

**Fonction universelle.** On se donne une numérotation effective des machines de Turing déterministes travaillant sur l'alphabet  $\{0, 1\}$  :  $M_0, M_1, \dots, M_n, \dots$

On considère le tableau  $T$  infini, indexé par  $\mathbb{N} \times \mathbb{N}$ , tel que  $T[n, p]$  soit la valeur calculée par la machine  $M_n$  travaillant sur la représentation binaire de l'entier  $p$  (ou  $\uparrow$  si la machine ne s'arrête pas).

1. Est-il vrai que, pour toute fonction calculable  $g : \mathbb{N} \rightarrow \mathbb{N}$ , il existe une ligne de ce tableau dont les valeurs sont  $g(0), g(1), \dots, g(n), \dots$  ?
2. Est-il vrai que, pour toute fonction calculable  $g : \mathbb{N} \rightarrow \mathbb{N}$ , il existe une colonne de ce tableau dont les valeurs sont  $g(0), g(1), \dots, g(n), \dots$  ?
3. Montrer qu'il n'existe pas de fonction calculable totale (partout définie)  $F : \mathbb{N} \times \mathbb{N} \rightarrow \mathbb{N}$  telle que si  $T[n, p] \neq \uparrow$ , alors  $F(n, p) = T[n, p]$ .

### Exercice 4.4

1. Montrer que toute fonction totale, croissante et bornée (de  $\mathbb{N}$  dans  $\mathbb{N}$ ) est calculable.
2. Donner un exemple de fonction totale bornée qui n'est pas calculable.
3. Donner un exemple de fonction croissante bornée qui n'est pas calculable.
4. Construire une fonction totale croissante qui n'est pas calculable.
5. Montrer que, si  $f : \mathbb{N} \rightarrow \mathbb{N}$  est une bijection calculable, sa réciproque  $f^{-1}$  est calculable.
6. Pour toute fonction calculable  $f : \mathbb{N} \rightarrow \mathbb{N}$ , on définit

$$\bar{f}(n) = \max\{x \in \mathbb{N} \mid f(x) \leq n\}$$

- (a) Supposons que  $f$  est totale, croissante, non bornée et telle que  $f(0) = 0$ . La fonction  $\bar{f}$  est-elle totale ? calculable ?
- (b) Supposons que  $f$  est totale, croissante et non bornée. La fonction  $\bar{f}$  est-elle totale ? calculable ?
- (c) Supposons que  $f$  est totale et croissante. La fonction  $\bar{f}$  est-elle totale ? calculable ?

### Exercice 4.5

**Théorème de Rice.** Voici une liste d'applications directes du théorème de Rice :

1. on ne peut pas décider si une machine de Turing calcule une fonction totale ;
2. on ne peut pas décider si une machine de Turing calcule une fonction injective ;
3. on ne peut pas décider si une machine de Turing calcule une fonction croissante ;
4. on ne peut pas décider si une machine de Turing calcule une fonction à valeurs bornées.

Dans chaque cas, formuler précisément le problème, et construire (sur le modèle de la preuve du théorème de Rice) une machine de Turing contradictoire, qui prouve par l'absurde le résultat d'indécidabilité.

$f$  ne peut être aucun des  $P_m$

si on avait  $f = P_m$ , on aurait  $f(m) = P_m(m)$  et  $f(m) = \begin{cases} 1 & \text{si } P_m(m) = 1 \\ \uparrow & \text{si } P_m(m) = 1 \end{cases}$

On a construit un prédictat contradictoire.

Si  $Q$  était calculable,  $f$  le serait aussi.

### exercice 4.3

généralisation  $\rightarrow$  énumération des fonctions calculables de  $\mathbb{N}$  dans  $\mathbb{N}$

machine de Turing calculant une fonction  
de  $\{0,1\}^+$  dans  $\{0,1\}^+$ .

$m$	0	1	2	...	$p$
0					$g(0)$
1					$g(1)$
2				:	
$\vdots$					
$m$					$T(m,p)$ = valeur calculée par $c/m$ sur l'entrée $p$ .

1.  $g$ : fonction calculable de  $\mathbb{N}$  dans  $\mathbb{N}$

$\hookrightarrow$  il existe une machine de Turing  $c/m$  qui calcule  $g$ ,  $g(p) = T(m,p)$   
pour tout  $p$  (les valeurs de  $g$  seront la  $m$ -ième ligne).

2.  $g$  correspond à la  $p$ -ième colonne  $\Leftrightarrow \forall m, g(m) = T(m,p)$ .

contre-exemples: -  $g(x) = x$ .

-  $m$  = le numéros d'une machine qui branche tout le temps  
( $f_m$  n'est définie nulle part)  $\rightarrow T(m,p) = \uparrow \neq g(m) = m$ .

$T(m,p)$  calculable:

- construire la machine  $c/m$

- la faire tourner sur  $p$

$\hookrightarrow$  machine universelle.

#### Exercice 4.4.

1. Toute fonction totale, croissante et bornée est calculable car elle ne prend qu'un nombre fini de valeurs et qu'elle est stationnaire à partir d'un certain rang.

2.  $f(m) = \begin{cases} 1 & \text{si } M_m \text{ s'arrête sur la bande vierge} \\ 0 & \text{sinon} \end{cases}$

(exemple type de fonction non calculable).

machine qui reconnaît un langage

$\Leftrightarrow$  fonction retournant 0 ou 1.

machine qui calcule un résultat

$\Leftrightarrow$  fonction affichant le résultat à l'écran

3.  $g(m) = \begin{cases} 0 & \text{si } M_m \text{ ne s'arrête pas bande vierge} \\ 1 & \text{sinon} \end{cases}$

4.  $h(m) = \begin{cases} m & \text{si } M_m \text{ s'arrête sur la bande vierge} \\ m-1 & \text{sinon} \end{cases}$

#### Exercice : Théorème de Rice

P fixé / - propriété de langages

/ ou - ensemble de langages récursivement énumérables

M : machine de Turing

question :  $L(M) \in P ?$

Théorème de Rice : si P n'est pas triviale alors la question est indécidable.

exemples de propriété :

- L est vide

- L est fini

- L est rationnel

- L est algébrique

P triviale  $\Leftrightarrow$  nulle pour tous les langages ou fausse pour tous les langages.

$E \in L(M) \Leftrightarrow M \text{ s'arrête sur la bande vierge dans l'état } OK$

$L(M) = \emptyset \Leftrightarrow \text{pour aucun mot d'entrée } M \text{ ne s'arrête sur } OK.$

$$M \rightarrow 1. L(M) = \Sigma^*$$

2. M accepte le mot "42"  $\Leftrightarrow 42 \in L(M)$

3. M s'arrête sur tout mot d'entrée ... se réduit à 1 et réciproquement

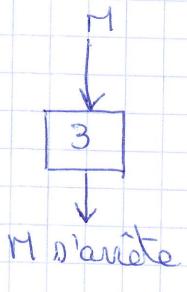
4. M n'accepte pas le mot "729"  $\Leftrightarrow 729 \notin L(M)$

5. M s'arrête sur la bande vierge ... se réduit à  $\epsilon \in L(M)$

6. M est un automate déterministe fini  $\Rightarrow L(M)$  est rationnel

7.  $L(M)$  est rationnel.

Quelles sont les applications directes du théorème de Rice ? Toute sauf 6.  
réduction du problème 3 au problème 1.



$M = M'$  et on complète la machine + remplacer les transitions vers  $K_0$  par des transitions vers  $OK$ .

### Feuille 5:

On considère des MT "qui décident" travaillant sur des entiers (codage non précisé)

$E \subseteq \mathbb{N}$  est récursivement énumérable

$\Leftrightarrow$  il existe une MT telle que  $L(M)$  est l'ensemble des écritures des entiers de  $E$

$\Leftrightarrow$  il existe un algorithme de décision pour  $d(m)$  - si  $m \in E$

$E \subseteq \mathbb{N}$  est décidable = récursif ↑ sinon

$\Leftrightarrow$  il existe une MT s'arrêtant sur toute entrée

$\Leftrightarrow$  il existe un algorithme implementant  $d(m) = \begin{cases} 1 & \text{si } m \in E \\ 0 & \text{sinon} \end{cases}$

### Exercice S.1

Si : supposons  $f$  calculable, montrons  $E$  décidable

si  $f(m+1) = f(m)$  alors  $m \notin E$

sinon  $m \in E$ , on a alors  $f(m+1) = f(m) + 1$

Seulement si : supposons  $E$  décidable.

int  $f(\text{int } m)$

{

resultat = 0;

pour  $i$  de 0 à  $m-1$ ;  $i++$

resultat += appartientAE ( $m$ )

} return resultat;

Pour tracer  $f$ , on examine toutes les  $i < m$  et on compte ceux qui appartiennent à  $E$ .

### Exercice S.2

Donnée :  $M$ : une MT s'arrêtant sur toute entrée dans OK ou KO

Question : 1 si  $L(M)$  est fini  $\Leftrightarrow M$  ne s'arrête en OK que sur un nombre fini d'entrées

O sinon

On peut appliquer le théorème de Rice.

la propriété  $L(M)$  est fini n'est pas triviale

$E \subseteq \mathbb{N}$  est récursivement énumérable

$\Leftrightarrow$  il existe une fonction calculable  $f: \mathbb{N} \rightarrow \mathbb{N}$  telle que  $f(\mathbb{N}) = E$

(def)  $\Leftrightarrow E$  est le langage reconnu par une machine de Turing.

$f$  est une énumération des éléments de  $E$ : au temps 0, on lance  $M$  sur 0,

au temps 1, on lance  $M$  sur 1, ...  
en parallèle

$M'$  fait tourner  $M$  sur 0, 1, 2, ... et compare  $f(k)$

à  $m$  et s'arrête sur OK

$f(k)$

OK si  $m \in E$



1. Montrer que si  $E$  est RE, on peut énumérer  $E$  sans répétition.

→ on stocke les entiers sortis par l'algorithme

2. Montrer que si  $E$  est décidable, on peut l'énumérer en ordre croissant.

→ on teste les éléments dans l'ordre croissant.

3. Montrer que si  $E$  peut être énumérée en ordre croissant, il est décidable.

→ dès qu'en le dépasse, on s'arrête (si  $E$  est fini, ça marche pas,  $E$  décidable)

$E$  est décidable  $\Leftrightarrow$  il existe une fonction calculable croissante telle que  $f(N) \in E$ .

$$E \text{ est co-RE} \Leftrightarrow \bar{E} \in \text{R.E}$$

$m_1, \dots, m_m$  : machines de Turing et  $f_m$  : fonction calculée par  $m_m$

### Exercice 5.3

décidable ?

RE ?

co-RE ?

$\{m | M_m \text{ s'arrête sur la bande vierge}\}$

NON

OUI

NON

$\{m | M_m \text{ boucle}\}$

NON

NON

OUI

$\{m | f_m(0) = 0\}$

NON

OUI

NON

$\{m | f_m \text{ croissante}\}$

NON

NON

OUI ... on calcule  $f_m(j)$   
... on vérifie  
 $f_m(j) > f_m(j-1)$

int estcroissante (int (\*f)) (int)

int self (int m)

if (estcroissante (self)) // propriété sur la fonction

return m % 2 == 0; // code ne vérifiant pas la propriété

else return m; // code vérifiant la propriété

$\{m | \text{Dom}(f_m) \neq \emptyset\}$   
 $= \{m_m \text{ s'arrête pour au moins 1 entrée}\}$

NON

OUI ... on lance  
en parallèle  $M_i(j)$

NON

Toute propriété non triviale des fonctions  $f_m$  est indécidable.

Montrer que 3-CC se réduit polynomiallement à SAT

donnée :

un graphe non orienté  $G$

$G$  est-il 3-colorable ?

donnée :

une fonction booléenne  $f$  en FNC

$f$  est-elle satisfaisable ?

réduction linéaire

variables :  $n_0, n_1, \dots, n_6, v_0, v_1, \dots, v_6, b_0, b_1, \dots, b_6$

$$n_i = \begin{cases} 1 & \text{si le sommet } i \text{ est rouge} \\ 0 & \text{sinon} \end{cases}$$

$$v_i = \begin{cases} 1 & \text{si le sommet } i \text{ est vert} \\ 0 & \text{sinon} \end{cases}$$

~~Si un sommet n'est ni rouge ni vert, il est blanc.~~

$$b_i = \begin{cases} 1 & \text{si le sommet } i \text{ est blanc} \\ 0 & \text{sinon} \end{cases}$$

$0$  et  $2$  sont de couleurs différentes :  $(\bar{n}_0 + \bar{n}_2)(\bar{v}_0 + \bar{v}_2)(\bar{b}_0 + \bar{b}_2)$

idem pour  $0$  et  $3 \dots$  etc.

à quelques m sommets et p arêtes

on a 3 m variables.

$$F = \bigwedge_{\substack{i,j \text{ est une} \\ \text{arête de } G}} (\bar{n}_i \vee \bar{n}_j) \wedge (\bar{v}_i \vee \bar{v}_j) \wedge (\bar{b}_i \vee \bar{b}_j)$$

$\wedge \bigwedge_{\substack{i \text{ est un} \\ \text{sommel}}} (n_i \vee v_i \vee b_i)$

donnée : un entier  $m$  et une

matrice  $(m,m)$  à coefficients de  $\{0,1\}$  / graphes non  
orientés / graphes bipartis

Marriage Perfect

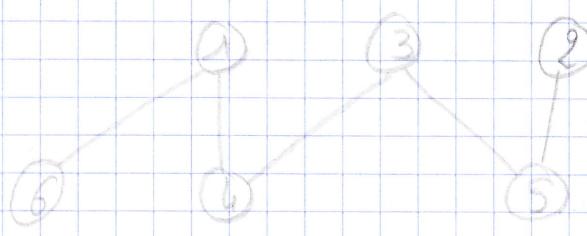
Réduction linéaire  $\rightarrow$  SAT

peut-on remplacer certains 1 /  
par des 0 de façon à avoir / supprimer  
exactly 1 membre 1 / des arcs

par ligne et par colonne ? / chaque sommet  
a une place  
entrant et une  
sortante

Jean Aziz Paul

	1	2	3	
4	Marie	1	0	1
5	Anne	0	1	1
6	Nadia	1	0	0



## Modèles de Calcul : feuille 5

Ensembles récursifs, récursivement énumérables, co-récursivement énumérables

### Exercice 5.1

Soit  $E$  un ensemble d'entiers naturels, et  $\varphi$  la fonction définie par :

$$\varphi(n) = \#\{x \in E \mid x < n\}$$

*le nombre d'éléments de l'ensemble*

Montrer que  $E$  est décidable si et seulement si  $\varphi$  est calculable.

### Exercice 5.2

Peut-on décider si un ensemble décidable est fini ? On commencera par formuler rigoureusement le problème.

### Exercice 5.3

On considère une énumération effective  $M_1, \dots, M_n, \dots$  des machines de Turing déterministes à une bande finie à gauche, travaillant sur l'alphabet  $\{\square\}$  (entiers en écriture unaire). On note  $f_n$  la fonction de  $\mathbb{N}$  dans  $\mathbb{N}$  calculée par la machine  $M_n$  :  $f_n(x) = y$  si la machine  $M_n$ , en travaillant sur le mot  $|^x$ , s'arrête à gauche du mot  $|^y \square$  (si  $M_n$  ne s'arrête pas,  $f_n(x)$  n'est pas définie).

Les ensembles suivants sont-ils décidables ? récursivement énumérables ? co-récursivement énumérables ?

1. l'ensemble des entiers  $n$  tels que  $f_n(0)$  est définie
2. l'ensemble des entiers  $n$  tels que  $f_n(0) = 0$
3. l'ensemble des entiers  $n$  tels que  $f_n$  est croissante
4. l'ensemble des entiers  $n$  tels que  $\text{Dom}(f_n) \neq \emptyset$
5. l'ensemble des entiers  $n$  tels que, pour tout entier  $x$ ,  $f_n(x) \leq x^2$
6. l'ensemble des entiers  $n$  tels que la machine  $M_n$  travaille sur  $|^x$  en temps  $O(x^2)$

### Exercice 5.4

(Examen 2007-2008.) On dit qu'un langage infini  $L \subseteq \{0, 1\}^*$  peut être énuméré en ordre croissant s'il existe une bijection calculable  $f : \mathbb{N} \setminus \{0\} \rightarrow L$  qui à chaque  $n > 0$  associe le  $n$ -ième mot de  $L$  dans l'ordre hiérarchique.

1. Montrer que tout langage décidable infini peut être énuméré en ordre croissant.
2. Montrer que tout langage qui peut être énuméré en ordre croissant est décidable.
3. Montrer que tout ensemble récursivement énumérable infini contient un ensemble décidable infini.
4. Montrer que l'ensemble des machines de Turing qui acceptent au moins un mot de la forme  $0^n$  est récursivement énumérable, mais n'est pas récursif.

1 si on garde l'arête  
0 si on la supprime arête  
 $\hookrightarrow x_{ii}$  pour chaque couple  $(i, i')$  pour tous les sommets.

$(\overline{x_{00}} \wedge \overline{x_{01}}) \vee (\overline{x_{02}} \wedge \overline{x_{03}})$   $\leftarrow$  0 est marié à quelqu'un

$\wedge (\overline{x_{00}} \vee \overline{x_{01}}) \wedge (\overline{x_{00}} \vee \overline{x_{02}}) \wedge \dots$

pour toutes les paires de couples ayant un élément en commun.  
d'arêtes sommet

variables  $\leftrightarrow$  éléments d'une solution

### Feuille 6:

	Définition	Caractérisation informelle	Technique de preuve
Problème P	peut être résolu par une MT déterministe travaillant en temps polynomial par rapport à la taille du mot d'entrée	on peut le résoudre par un algorithme en temps polynomial par rapport à la taille des données	- donner l'algo - réduire polynomiallement à un autre problème P.
Problème NP polynomial non déterministe	peut être résolu par une MT non déterministe travaillant en temps polynomial par rapport à la taille des données	il existe un système de certificats (= preuves) + vérificateur polynomial dans P tel que $Q(x) \Rightarrow V(x, y)$	- donner un système de certificats + vérificateur V - réduire à un autre P.
Problème NP-complet NP-difficile	tout problème de NP peut s'y réduire $\text{NP-complet} = \text{NP-difficile et NP}$	— —	- réduire polynomiallement à ce problème un problème déjà connu comme NP-complet (ex: SAT, 3-color, clique, ...)

donnée  $x \rightarrow$  Problème Q  $\rightarrow Q(x) \in \{\text{VRAI}, \text{FAUX}\} \Leftrightarrow \exists y \mid V(x, y) = \text{VRAI}$

certificat  $y \rightarrow$  Vérificateur V  $\rightarrow V(x, y) = \text{VRAI}$  seulement si  $Q(x) = \text{VRAI}$

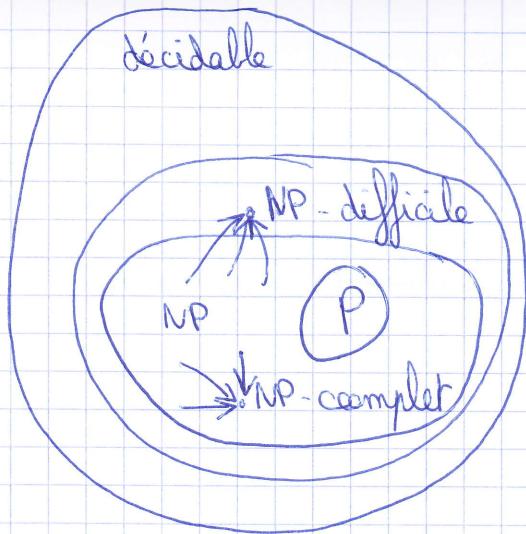
exemple:

$m \rightarrow$  COMPOSITE FAUX si m est premier  
VRAI sinon

certificat:  $(m_1, m_2) \rightarrow$  Vérificateur VRAI si  $m_1 \times m_2 = m$  avec  $m_1, m_2 \neq 1$   
FAUX sinon

Q est dans NP si il existe un problème V dans la classe P tel que

$Q(x) = \text{VRAI} \Leftrightarrow \exists y, V(x, y) = \text{VRAI}$



### Exercice 6.6.



$G$  est 3-colorable  $\Leftrightarrow G'$  est 4-colorable

$G'$ :  $G$  auquel on ajoute un sommet relié à tous ceux de  $G$ .

FND: somme de produits de littéraux ( $x$  ou  $\bar{x}$ ) avec au plus un littéral  $x$  ou  $\bar{x}$  par produit pour chaque variable  $.xyz + \bar{x}\bar{y}\bar{z}$

Automate déterministe

un mot  $w$

D-REC

$w \in L(A) ?$

determiniser  
l'automate  
NON POLYNOMIAL  
(en  $2^m$ )

Automate non déterministe à m états

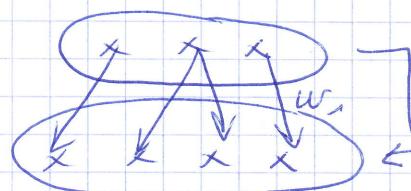
un mot  $w = w_1 w_2 \dots w_p$

N-REC

$w \in L(A) ?$

algéo en temps  $.p \times m^2$

états initiaux



au pire  $m^2$

## Modèles de Calcul : feuille 6

Classes P et NP

### Exercice 6.1

Le problème du *mariage parfait* est le suivant :

**Données** Un entier  $n$ , et une matrice  $n \times n$  à coefficients dans  $\{0, 1\}$ .

**Question** Peut-on remplacer un certain nombre de 1 par 0 dans la matrice donnée pour obtenir une matrice ayant exactement une entrée 1 par ligne et par colonne ?

1. Expliquer comment coder une donnée du mariage parfait.
2. Donner une machine de Turing non déterministe pour résoudre le problème du mariage parfait. Quelles sont les complexités en temps et en espace de cette machine de Turing ? Que peut-on en déduire sur la complexité du problème du mariage parfait ?

### Exercice 6.2

Soit  $\varphi$  une instance de 2-SAT. On définit un graphe orienté  $G(\varphi)$  par son ensemble de sommets

$$S = \{x \mid x \text{ est une variable de } \varphi\} \cup \{\bar{x} \mid x \text{ est une variable de } \varphi\}$$

et son ensemble d'arêtes

$$A = \{(L_1, L_2) \mid (\bar{L}_1 + L_2) \text{ ou } (L_2 + \bar{L}_1) \text{ est une clause de } \varphi\}$$

(où  $L_1$  et  $L_2$  sont des littéraux)

1. Montrer que si  $(L_1, L_2)$  est une arête de  $G(\varphi)$ , alors  $(\bar{L}_2, \bar{L}_1)$  est aussi une arête de  $G(\varphi)$ . Que traduit l'existence d'un chemin de  $L_1$  à  $L_2$  dans  $G$  ?
2. Montrer que  $\varphi$  n'est pas satisfaisable si et seulement si il existe une variable  $x$  telle qu'il y ait un chemin de  $x$  à  $\bar{x}$  et un chemin de  $\bar{x}$  à  $x$  dans  $G$ .
3. En déduire que 2-SAT est dans P.

### Exercice 6.3

Soit  $G$  un graphe non orienté,  $k$  un entier.

Le problème COUVERTURE consiste à déterminer si  $G$  admet une couverture de taille  $k$ , c'est-à-dire s'il existe un ensemble de  $k$  sommets de  $G$  tel que toute arête de  $G$  ait au moins une extrémité dans cet ensemble.

Le problème STABLE consiste à déterminer si  $G$  admet un stable de taille  $k$ , c'est-à-dire s'il existe un ensemble de  $k$  sommets de  $G$  tel que toute arête de  $G$  ait au plus une extrémité dans cet ensemble.

Démontrer que les problèmes COUVERTURE et STABLE sont NP-complets.

### Exercice 6.4

Soit  $G$  un graphe, considéré suivant les cas comme orienté ou non. Démontrer que chacun des problèmes suivants se réduit polynomialement à chacun des autres :

1.  $G$  (non orienté) admet-il un cycle hamiltonien ?
2.  $G$  (non orienté) admet-il une chaîne hamiltonienne ?
3.  $G$  (orienté) admet-il un circuit hamiltonien ?
4.  $G$  (orienté) admet-il un chemin hamiltonien ?
5. Sachant que  $G$  (non orienté) est biparti,  $G$  admet-il un cycle hamiltonien ?

### Exercice 6.5

Montrer que le problème POL2 est dans la classe P :

**Donnée :** Un polynôme  $P(X_1, X_2, \dots, X_n)$  à coefficients dans  $\mathbb{Z}/2\mathbb{Z}$ .

**Question :** Existe-t-il  $x_1, x_2, \dots, x_n \in \mathbb{Z}/2\mathbb{Z}$  tel que  $P(x_1, x_2, \dots, x_n) = 0$  ?

Le professeur Cosinus a griffonné un manuscrit (partiellement lisible) prouvant que P=NP :

1- POL2 est dans la classe P.

2- Les connecteurs  $\wedge, \neg$  s'expriment par des polynômes à deux variables :

$$x \wedge y = x \cdot y, \quad \neg x = 1 + x, \quad x \vee y = \dots$$

(son manuscrit est illisible pour la traduction de  $\vee$ ).

3- Toute formule booléenne  $\varphi$  sur n variables peut donc être traduite en un polynôme  $P(X_1, X_2, \dots, X_n)$  à coefficients dans  $\mathbb{Z}/2\mathbb{Z}$ , tel que l'équation  $P(x_1, x_2, \dots, x_n) = 0$  ait une solution si et seulement si  $\varphi$  est satisfaisable.

4- Comme les formules du point (2) sont constantes, la traduction  $\varphi \rightarrow P$  peut être effectuée en temps linéaire.

5- Par les points (3,4),  $\varphi \rightarrow P$  est une réduction polynomiale de SAT à POL2. Comme SAT est NP-complet et que POL2 est P, tout problème NP est P. Donc P=NP.

Que pensez-vous de ce manuscrit ? Cosinus pourra-t-il encaisser le million de dollars promis par la fondation Clay (voir [http://www.claymath.org/millennium/P\\_vs\\_NP/](http://www.claymath.org/millennium/P_vs_NP/)) ?

### Exercice 6.6

Placer chacun des problèmes ci-dessous dans les classes P,NP, NP-complet :

**Donnée :** Un graphe non-orienté  $G = (S, A)$ .

**PB1 :**  $G$  est-il 2-coloriable ?

**PB2 :**  $G$  est-il 3-coloriable ? *NP-complet*

**PB3 :**  $G$  est-il 4-coloriable ? *un graphe planaire est 4-coloriable* *NP-complet*

**PB4 - Donnée :** Une formule booléenne  $\varphi$  en forme normale disjonctive.

**Question :**  $\varphi$  est-elle satisfaisable ? *temps constant*

**PB5 - Donnée :** Une formule booléenne  $\varphi$  en forme normale conjonctive.

**Question :**  $\varphi$  est-elle satisfaisable ? *SAT* *NP-complet*

**PB6 - Donnée :** Une formule booléenne  $\varphi$ .

**Question :**  $\varphi$  est-elle satisfaisable ?

**PB7 - Donnée :** Une formule booléenne en CNF :  $\varphi = C_1 \wedge C_2 \wedge \dots \wedge C_n$  où chaque  $C_i$  est une clause de Horn, c'est-à-dire  $C_i = L_1 \vee l_2 \dots \vee L_{\ell_i}$  pour des littéraux  $L_1, L_2 \dots L_{\ell_i}$  tels que l'un au plus d'entre eux est positif.

**Question :**  $\varphi$  est-elle satisfaisable ?

**PB8 - Donnée :** Un automate fini déterministe  $A$  et un mot  $w$ .

**Question :**  $w \in L(A)$  ? *P*

**PB9 - Donnée :** Un automate fini non-déterministe  $A$  et un mot  $w$ .

**Question :**  $w \in L(A)$  ? *P*

**PB10 - Donnée :** Un automate fini déterministe  $A$ .

**Question :**  $L(A) \neq \emptyset$  ? *P* *on fait un parcours depuis un état initial pour trouver*

**PB11- Donnée :** Un automate fini non-déterministe  $A$ .

**Question :**  $L(A) \neq \emptyset$  ? *P* *on fait un parcours depuis un état initial pour trouver*

*un état final.*

### exercice 6.2 :

donnée : une formule en FNC dont toute clause comporte au plus 2 littéraux



1 si la formule est satisfaisable

O sinon

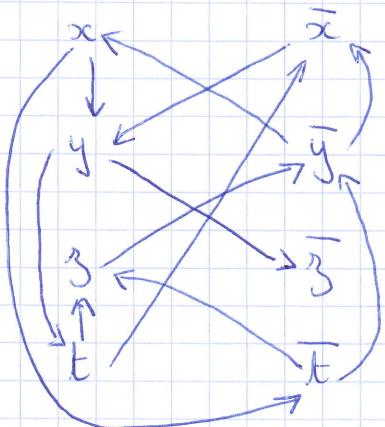
$$(x+y)(\bar{y}+t)(\bar{x}+\bar{t})(\bar{y}+\bar{z})(z+t)(\bar{x}+y)$$

$$\bar{x} \rightarrow y \quad y \rightarrow t \quad \bar{x} \rightarrow \bar{t} \quad \bar{y} \rightarrow \bar{t}$$

$$\bar{y} \rightarrow x \quad \bar{t} \rightarrow \bar{y} \quad \bar{t} \rightarrow \bar{x} \quad t \rightarrow z$$

$$L_1 \rightarrow L_2 \quad L_1 \rightarrow \bar{L}_2 \quad \bar{L}_1 \rightarrow L_2$$

$$\bar{L}_2 \rightarrow \bar{L}_1 \quad \bar{L}_2 \rightarrow \bar{L}_1 \quad L_2 + \bar{L}_1$$



- S'il existe un circuit  $x \rightsquigarrow \bar{x} \rightsquigarrow x$ , la formule n'est pas satisfaisable (évident, si la formule était satisfaisable, on aurait  $x=0$  et  $x=1$ )
- S'il n'y a pas de tel circuit, la formule est satisfaisable.

idée : - s'il existe un chemin de  $x$  à  $\bar{x}$ , on choisit  $x = 0$

- s'il existe un chemin de  $\bar{x}$  à  $x$ , on choisit  $x = 1$ .

→ s'il reste des variables à affecter, on en affecte une à 0.

(puis on affecte à 0 toutes celles qui y conduisent et à 1 toutes celles d'où conduit le complémentaire)

graphes orientés dont les sommets sont appariés (on a  $2 \times m$  sommets)

formule FNC

2-SAT

polynomiale



1 si  $f$  est satisfaisable

0 sinon

existe-t-il un circuit passant  
par 2 sommets appariés ?

### exercice 6.4

• Chemin hamiltonien = chemin passant 1 fois par chaque sommet.

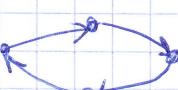
chemin



chaîne



circuit



cycle



D: graphe G

Hamilton

1/ G non orienté a-t-il un cycle H ?

2/ G non orienté a-t-il une chaîne H ?

3/ G orienté a-t-il un circuit H ?

4/ G orienté a-t-il un chemin H ?

5/ G non orienté biparti admet-il un cycle H ?

linéaire

réduction identité

1/ -> on choisit un sommet  $s_0$

on ajoute un graphe 3 sommets d, f, copie de  $s_0$

et les arêtes  $d - s_0$ ,  $s_0' - f$

et pour toute arête  $s - s_0$ , une arête  $s - s_0'$

cycle  $s_0 - s_1 - \dots - s_m - s_0$

chaîne  $d - s_0 - s_1 - \dots - s_m - s_0' - f$  dans  $G'$



**Université Bordeaux 1. Master Sciences & Technologies, Informatique.**

**Examen UE INF 461, Modèles de calcul.**

**Session 1 2008–2009. 17 décembre 2008. Durée : 3h00.**

**Manuscrits et polycopiés autorisés, livres interdits.**

**La notation tiendra compte du soin apporté à la rédaction.**

**Exercice 1** Répondre par vrai ou faux aux questions suivantes, sans donner une justification.  
Barème : +0,5 pour une réponse juste, -0,5 pour une réponse fausse, 0 pour une absence de réponse.

1. On peut écrire un programme qui produit en sortie la suite de tous les programmes C syntaxiquement corrects.
2. On peut écrire un programme qui teste si le fichier qu'on lui transmet en entrée est un programme C syntaxiquement correct qui s'arrête sur toute entrée.
3. S'il existe une réduction polynomiale de  $P_1$  à  $P_2$  ( $P_1 \leq_p P_2$ ), et si  $P_2$  est dans la classe NP, alors  $P_1$  est dans la classe NP.
4. Le complémentaire d'un langage récursivement énumérable est récursivement énumérable.
5. Pour tous  $K, L \subseteq A^*$  avec  $A = \{0, 1\}$ , si  $K$  se réduit à  $L$ , alors  $A^* \setminus K$  se réduit à  $A^* \setminus L$ .
6. Étant données une formule Booléenne  $\varphi$  en forme normale conjonctive, et une valuation  $f : \{x_1, \dots, x_n\} \rightarrow \{0, 1\}$  de chacune de ses variables, on peut décider en temps linéaire si la valuation  $f$  rend  $\varphi$  vraie.

**Exercice 2** 1. Écrire une machine de Turing  $M_{a,b}$  qui accepte les mots sur l'alphabet  $\{a, b\}$  ayant au moins autant de  $a$  que de  $b$ , et rejette les autres. On demande

- une description claire et précise (états et transitions) de  $M_{a,b}$ ,
- une justification que la machine s'arrête toujours et accepte bien le langage demandé.

2. En utilisant des machines du type de celle écrite en question 1, décrire une machine qui accepte les mots ayant autant de  $a$  que de  $b$ , et rejette les autres.

**Exercice 3** 1. On considère le problème suivant :

**Donnée** Une machine de Turing  $M$ .

**Question** Le langage  $\mathcal{L}(M)$  est-il fini ?

Montrer que ce problème est indécidable

(a) en utilisant le théorème de Rice, en justifiant clairement.

(b) sans utiliser le théorème de Rice, par réduction ~~au~~ <sup>à partir du</sup> problème de l'arrêt sur mot vide.

2. Le problème suivant est-il également indécidable ? Justifier.

**Donnée** Une machine de Turing  $M$ .

**Question** Le langage  $[\mathcal{L}(M)]^*$  est-il fini ?

**Exercice 4** On considère le problème MOT-CONCORDANT suivant.

**Donnée** Des mots  $u_1, \dots, u_p$  sur l'alphabet  $\{0, 1, *\}$ , tous de même taille  $n$ .

**Question** Trouver s'il existe un mot  $y$  sur l'alphabet  $\{0, 1\}$ , également de taille  $n$ , qui concorde avec chaque mot  $u_i$  sur au moins une position. C'est-à-dire, si on note  $u_i = a_{i,1} \cdots a_{i,n}$ , qu'on demande l'existence d'un mot  $y = y_1 \cdots y_n$  tel que :

- (1) Pour toute position  $k$  entre 1 et  $n$  :  $y_k = 0$  ou  $y_k = 1$ , et
- (2) Pour tout  $i$  entre 1 et  $p$  : il existe une position  $\ell$  telle que  $y_\ell = a_{i,\ell}$ .

1. Le problème a-t-il une solution sur l'instance  $u_1 = 01*$ ,  $u_2 = **0$ ,  $u_3 = 1*0$ ,  $u_4 = *1*$  ?
2. Le problème a-t-il une solution sur l'instance  $u_1 = 01*$ ,  $u_2 = **0$ ,  $u_3 = 1*1$ ,  $u_4 = *1*$  ?
3. Décrire un algorithme qui vérifie en temps polynomial (par rapport à  $p.n$ ) si un mot  $y$  donné est une solution du problème MOT-CONCORDANT. Peut-on conclure que le problème appartient à la classe P ? à la classe NP ?

On veut construire une réduction polynomiale de 3-SAT à MOT-CONCORDANT. Soit  $\varphi = c_1 \wedge c_2 \wedge \cdots \wedge c_p$  une formule 3-CNF sur  $n$  variables  $x_1, \dots, x_n$ , avec la convention qu'aucune clause ne contient à la fois une variable et sa négation. Pour chaque clause  $c_i$ , on construit un mot  $u_i = a_{i,1} \cdots a_{i,n}$ . La  $k^{\text{ème}}$  lettre  $a_{i,k}$  de  $u_i$  est définie ainsi :

$$a_{i,k} = \begin{cases} 0 & \text{si } \neg x_k \text{ apparaît dans } c_i, \\ 1 & \text{si } x_k \text{ apparaît dans } c_i, \\ * & \text{si ni } x_k, \text{ ni } \neg x_k \text{ n'apparaissent dans } c_i. \end{cases}$$

4. Montrer comment à partir d'une valuation des variables  $x_1, \dots, x_n$  qui rend  $\varphi$  vraie, on peut construire une solution de MOT-CONCORDANT sur l'instance  $u_1, \dots, u_p$  ainsi définie.
5. Inversement, montrer que si MOT-CONCORDANT a une solution  $y$  sur l'instance  $u_1, \dots, u_p$ , alors  $\varphi$  est satisfaisable (en interprétant  $y = y_1 \cdots y_n$  comme une valuation des variables  $x_1, \dots, x_n$ ).
6. Que déduit-on des questions précédentes concernant le problème MOT-CONCORDANT ?

- Exercice 5**
1. Montrer que la fonction  $f(x, y) = y^x$  est primitive récursive (avec par convention :  $0^0 = 1$ ). On pourra utiliser le fait que la multiplication est primitive récursive.
  2. Montrer que l'ensemble des fonctions primitives récursives de  $\mathbb{N}$  dans  $\mathbb{N}$  est dénombrable.
  3. En utilisant un argument de diagonalisation, montrer qu'il existe des fonctions totales de  $\mathbb{N}$  dans  $\mathbb{N}$ , calculables, mais non primitives récursives.