ÉCOLE POLYTECHNIQUE **OPTION MP**

CONCOURS D'ADMISSION 1998

COMPOSITION D'INFORMATIQUE

(Durée : 3 heures)

L'utilisation des calculatrices n'est pas autorisée pour cette épreuve.

***

AVERTISSEMENT. On attachera une grande importance à la clarté, à la précision et à la concision de la rédaction. En particulier, le candidat doit systématiquement justifier ses résultats.

Partie I. Motifs et automates.

MOTS ET MOTIFS.

Dans tout le problème, étant donnés un ensemble $\mathcal{E}$ et un entier naturel n, on définit un mot de longueur n sur $\mathcal{E}$ comme une séquence $m = a_0 a_1 \dots a_{n-1}$ d'éléments de $\mathcal{E}$. (Dans le cas $n = 0$, m est le mot vide.) Étant donnés deux entiers naturels i et k avec $0 \leq i < n$ et $0 \leq i + k < n$, le sous-mot de m de longueur k positionné en i est le mot $a_i a_{i+1} \dots a_{i+k-1}$. La concaténation de deux mots $m = a_0 a_1 \dots a_{n-1}$ et $m' = a'_0 a'_1 \dots a'_{n'-1}$ est le mot $m'' = mm' = a''_0 a''_1 \dots a''_{n+n'-1}$, avec $a''_i = a_i$, pour i compris entre 0 et $n-1$, et $a''_i = a'_{i-n}$, pour i compris entre n et $n+n'-1$. Réciproquement, m est alors un préfixe de m'' et m' un suffixe de m''.

Une lettre α est une lettre de l'alphabet, `a`, `b`, ..., `z`. Un mot (alphabétique) m est un mot de lettres. Un motif simple s est soit une lettre α, soit l'étoile `*`, soit un motif simple optionnel s'`?` où s' est un motif simple. Un motif p est un mot de motifs simples. Par exemple, `coucou` est un mot et un motif, tandis que `c?ou*cou` est un motif, ce dernier étant composé des sept motifs simples `c?`, `o`, `u`, `*`, `c`, `o` et `u`.

Les motifs filtrent les mots au sens suivant :

- Le motif vide filtre le mot vide,
- le motif lettre α filtre le mot d'une lettre α,
- le motif étoile `*` filtre tous les mots,
- un motif simple optionnel s`?` filtre les mêmes mots que s, plus le mot vide,
- un motif $p_1 p_2$, tel que ni p_1 ni p_2 ne sont le motif vide, filtre les mots $m_1 m_2$, où p_1 filtre m_1 et p_2 filtre m_2.

QUESTION 1. On étudie quelques propriétés du filtrage:

a) Donner les mots filtrés par les motifs `c1?ou` et `c?ou*cou` .

b) Trouver un motif p et un mot m, tels que p filtre m de deux façons distinctes, c'est à dire qu'il existe une décomposition $p_1 p_2$ de p, ainsi que deux décompositions différentes $m_1 m_2$ et $m_3 m_4$ de m, telles que p_1 filtre m_1 et m_3, et p_2 filtre m_2 et m_4.

c) Soient un motif $p = p_1 p_2$ et un mot m, tels que p filtre m. Montrer qu'il existe deux mots m_1 et m_2, tels que p_1 filtre m_1, p_2 filtre m_2 et $m = m_1 m_2$.

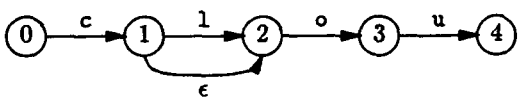
d) Deux motifs sont dits équivalents quand ils filtrent les mêmes mots. Le poids d'un motif est le nombre de caractères qui le composent. Par exemple, le poids du motif `c?ou*cou` est huit. Pour chacun des motifs `*?`, `**` et `a???`, donner un motif équivalent de poids minimal.

AUTOMATES EN LIGNE

Un automate en ligne à n états est un couple (E,T), où E est une suite d'états $e_0, e_1, \ldots, e_{n-1}$ et T est un ensemble de transitions. Une transition est un triplet formé de deux états et d'une étiquette. On se limite à trois formes de transitions :

- les transitions alphabétiques de la forme (e_i, e_{i+1}, α), où α est une lettre,
- les transitions étoile de la forme $(e_i, e_i, *)$,
- les transitions instantanées de la forme (e_i, e_{i+1}, ϵ).

Le problème ne traitera que des automates en ligne que l'on appellera tout simplement "automates". Un automate est donc un graphe dirigé dont les sommets sont les états $e_0, e_1, \ldots, e_{n-1}$ et les arcs sont les transitions. Voici par exemple un automate à cinq états :



À cause des transitions étoile et des transitions instantanées les automates en ligne sont des automates non-déterministes. Une façon de traiter ce non-déterminisme est de considérer des ensembles d'états.

Pour tout ensemble d'états A, on définit $\epsilon(A)$, la fermeture par transitions instantanées de A, comme le plus petit sur-ensemble de A qui obéit à la propriété suivante :

$$e_i \in \epsilon(A) \text{ et } (e_i, e_{i+1}, \epsilon) \in T \Rightarrow e_{i+1} \in \epsilon(A)$$

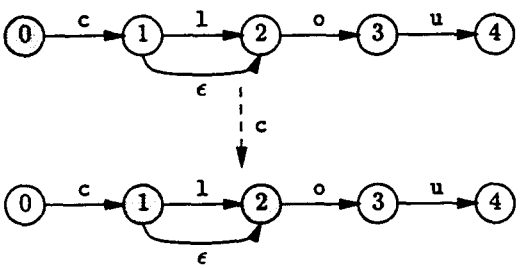
Un ensemble d'états A est dit clos par transitions instantanées, lorsque l'on a $\epsilon(A) = A$.

Étant donnés un ensemble d'états A et une lettre α, on note $\alpha(A)$ l'ensemble :

$$\alpha(A) = \{e_i \mid (e_{i-1} \in A \text{ et } (e_{i-1}, e_i, \alpha) \in T) \text{ ou } (e_i \in A \text{ et } (e_i, e_i, *) \in T)\}$$

Enfin, étant donnés un ensemble d'états A clos par transitions instantanées et une lettre α, on définit l'étape $A \xrightarrow{\alpha} A'$ en posant $A' = \epsilon(\alpha(A))$.

Voici par exemple, l'étape $\{e_0\} \xrightarrow{c} \{e_1, e_2\}$ dans le cas de l'automate déjà dessiné :



La lecture d'un mot $m = \alpha_0\alpha_1 \ldots \alpha_{k-1}$ de longueur $k \geq 0$ par un automate $\mathcal{A}$ à n états est la production de $A_0, A_1, \ldots, A_k$, suite d'ensembles d'états de $\mathcal{A}$, avec :

$$A_0 = \epsilon(\{e_0\}) \xrightarrow{\alpha_0} A_1 \xrightarrow{\alpha_1} \cdots \xrightarrow{\alpha_{k-1}} A_k$$

On note $\mathcal{A}[m]$ le dernier ensemble d'états A_k. Par définition, l'automate $\mathcal{A}$ reconnaît le mot m quand son état final e_{n-1} appartient à $\mathcal{A}[m]$. Par exemple, $\{e_0\} \xrightarrow{c} \{e_1, e_2\} \xrightarrow{o} \{e_3\} \xrightarrow{u} \{e_4\}$ est la reconnaissance de cou par l'automate dessiné.

QUESTION 2. On demande de construire un automate $\mathcal{A}(p)$ qui reconnaît les mots filtrés par p et seulement eux dans les cas suivants :

a) Le motif p est un motif simple.

b) Le motif p est vide.

c) Le motif p est composite : $p = p_1 p_2$.

d) Application : donner $\mathcal{A}$(c?ou*cou).

IMPLANTATION MACHINE DES AUTOMATES

La grande simplicité des automates en ligne autorise une implémentation efficace à l'aide d'entiers binaires, pourvu que leur nombre d'états soit suffisamment petit, condition que l'on supposera toujours vérifiée.

Un bit est 0 ou 1. La représentation binaire sur p bits d'un entier n est la suite de p bits, notée $b_{p-1} \ldots b_1 b_0$ ou (b_i), telle que l'on a $n = \sum_{i=0}^{p-1} b_i 2^i$. Du point de vue de la machine, un entier n et sa représentation binaire sont identiques. Outre l'arithmétique traditionnelle, un ordinateur fournit certaines opérations sur les entiers binaires. Le décalage de k bits à gauche $(b_i) \ll k$ est la suite (c_i), avec $c_i = 0$ pour $i < k$ et $c_i = b_{i-k}$ autrement. Le "ou logique" $(b_i) \vee (c_i)$, est la suite $(b_i \vee c_i)$, où $\vee$ est le "ou binaire", défini par $0 \vee 0 = 0$, $1 \vee 0 = 1$, $0 \vee 1 = 1$ et $1 \vee 1 = 1$. Le "et logique" $(b_i) \wedge (c_i)$ est la suite $(b_i \wedge c_i)$, où $\wedge$ est le "et binaire", défini par $0 \wedge 0 = 0$, $0 \wedge 1 = 0$, $1 \wedge 0 = 0$ et $1 \wedge 1 = 1$.

Un ensemble d'états A d'un automate à n états, avec $n \leq p$, sera implanté en machine par un entier (b_i), avec $b_i = 1$ si $e_i \in A$, les autres bits étant nuls. Les transitions de l'automate seront représentées de façon similaire, par un certain nombre d'entiers, à raison d'un entier N_α par lettre α, d'un entier N_* et d'un entier N_ϵ. On pose, $N_\alpha = (b_i)$, avec $b_i = 1$, si $(e_i, e_{i+1}, \alpha) \in T$; $N_* = (b_i)$, avec $b_i = 1$ si $(e_i, e_i, *) \in T$; et $N_\epsilon = (b_i)$ avec $b_i = 1$ si $(e_i, e_{i+1}, \epsilon) \in T$. (Tous les bits non spécifiés sont nuls.)

Par exemple, dans le cas de l'automate déjà dessiné et représentable sur cinq bits, il y a quatre entiers N_α non-nuls, à savoir $N_c = 00001$, $N_l = 00010$, $N_o = 00100$, $N_u = 01000$; en outre, on a $N_* = 00000$ (il n'y a aucune transition étoile) et $N_\epsilon = 00010$ (il existe une transition instantanée de e_1 vers e_2). L'étape $\{e_0\} \xrightarrow{c} \{e_1, e_2\}$ déjà dessinée est la transformation de 00001 en 00110.

QUESTION 3. Étant donnés un automate à n états implanté en machine par les entiers N_α, N_* et N_ϵ, un ensemble d'états A, clos par transitions instantanées, codé par un entier a, et une lettre α, donner une méthode de calcul de a', l'entier qui encode A' tel que $A \xrightarrow{\alpha} A'$. On n'oubliera pas que A' est, par définition, clos par transitions instantanées.

IMPORTANT. Le candidat doit à partir de maintenant choisir un langage de programmation. S'il choisit Caml, il poursuivra le problème par la partie 2-Caml ci-dessous, s'il choisit Pascal, il poursuivra le problème par la partie 2-Pascal.

Partie 2-Caml. Reconnaissance de motifs.

Les mots et les motifs sont représentés en machine par des chaînes de caractères. Formellement les chaines (de type `string`) sont des mots sur l'ensemble des caractères de la machine (de type `char`). On rappelle quelques primitives sur les caractères et les chaînes. La fonction `int_of_char` de type `char -> int` renvoie le code du caractère passé en argument. La fonction `char_of_int` réalise l'opération inverse. Il est à noter que les codes des lettres de l'alphabet sont consécutifs. La fonction `string_length` de type `string -> int` renvoie la longueur de la chaîne passée en argument. L'opération primitive `s1^s2` produit une nouvelle chaîne qui est la concaténation de `s1` et `s2`. Étant donnés une chaîne `s` de longueur n et un entier i, avec $0 \leq i < n$, la notation `s.[i]` désigne le caractère d'indice `i` de la chaîne `s`. (En Caml, l'indice du premier caractère dans une chaîne est zéro).

Le décalage à gauche $\ll$ est réalisé par la primitive `lsl`, le "ou logique" par la primitive `lor`, et le "et logique" par la primitive `land`. Ces trois primitives sont de type `int -> int -> int`. On rappelle que les entiers machine sont toujours assez grands pour pouvoir représenter les ensembles d'états des automates. On outre, on ne se préoccupera pas d'éventuelles erreurs dues aux débordements de capacité.

Comme on l'a déjà vu, un automate est la donnée d'une taille n, de N_α pour α variant de a à z, de N_* et de N_ϵ. Ces données seront rangées respectivement dans une variable globale `n_etats`, un tableau d'entiers `alpha`, ainsi que dans deux variables globales `etoile` et `epsilon`, dont les déclarations suivent :

```
let alpha = make_vect 26 (-1)
and etoile = ref (-1)
and epsilon = ref (-1)
and n_etats = ref (-1)
;;
```

QUESTION 4. Dans cette question, on suppose que les variables `n_etats`, `alpha`, etc. représentent un automate $\mathcal{A}$ à n états. Soit un entier `etats` qui représente un ensemble d'états A de $\mathcal{A}$, tel que A est clos par transitions instantanées. Soit également une lettre α représentée en machine par le caractère `c`.

a) Écrire les fonctions `mange` de type `int -> char -> int` et `ferme` de type `int -> int`, telles que les appels `mange etats c` et `ferme etats` renvoient un entier qui encode respectivement les ensembles $\alpha(A)$ et $\epsilon(A)$.

b) Écrire la fonction `etape` de type `int -> char -> int`, telle que l'appel `etape etats c` renvoie un entier qui encode l'ensemble d'états A' tel que $A \xrightarrow{\alpha} A'$.

c) Exprimer la complexité d'une exécution de `etape`, d'abord dans le cas où le motif p ne contient pas de motif optionnel, puis dans le cas général. Dans ce cas général, on pourra borner le coût d'exécution de `etape` en utilisant toute caractéristique jugée pertinente du motif p.

d) Écrire la fonction `etat_final` de type `int -> bool`, telle que l'appel `etat_final etats` teste la présence de l'état final e_{n-1} dans l'ensemble d'états A.

QUESTION 5. Le but de cette question est la construction de l'automate $\mathcal{A}(p)$.

a) Écrire la fonction `construire_auto` de type `string -> unit` qui prend en argument une chaîne de caractères `p` représentant un motif p et initialise les variables `n_etats`, `alpha`, `etoile` et `epsilon`, afin qu'elles encodent l'automate $\mathcal{A}(p)$. Dans le cas où la chaîne `p` n'est pas un motif, la construction de l'automate devra échouer. Pour ce faire, on dispose de la primitive `echoue` qui prend un message d'erreur en argument.

b) Évaluer la complexité de l'appel `construire_auto p` en fonction de la longueur de la chaîne `p`.

QUESTION 6. On programme la reconnaissance du filtrage d'un mot par un motif.

a) Écrire la fonction `filtre` de type `string -> string -> bool`, telle que l'appel `filtre p m` teste le filtrage du mot m par le motif p.

b) Donner la complexité de l'appel `filtre p m`, en fonction de la longueur de la chaîne `m` et de toute caractéristique pertinente de p, d'abord dans le cas où p ne contient pas de motif optionnel, puis dans le cas général.

QUESTION 7. Dans cette question, on étudie la reconnaissance des sous-mots d'un mot.

a) Soit p un motif et m un mot. Trouver un motif q tel que q filtre m, si et seulement si p filtre un sous-mot de m. En déduire une fonction `filtre_sous_mot` de type `string -> string -> bool`, telle que l'appel `filtre_sous_mot p m` teste le filtrage d'au moins un sous-mot de m par p.

b) On considère un motif p et un mot $m = \alpha_0\alpha_1 \ldots \alpha_{k-1}$. Soit de nouveau la suite d'ensemble d'états $A_0, A_1, \ldots, A_k$ engendrée par la lecture de m par $\mathcal{A}(p)$:

$$A_0 = \epsilon(\{e_0\}), \quad A_{i+1} = \epsilon(\alpha_i(A_i)) \text{ pour } 1 \leq i < k$$

Justifiez brièvement que l'état final de $\mathcal{A}(p)$ appartient à A_i, si et seulement si p filtre un préfixe $m_i = \alpha_0\alpha_1 \ldots \alpha_{i-1}$ de m.

Trouver (en justifiant son choix) une suite $B_0, B_1, \ldots, B_k$, telle que l'état final de $\mathcal{A}(p)$ appartient à B_i, si et seulement si p filtre un sous-mot $\alpha_j\alpha_{j+1} \ldots \alpha_{i-1}$ de m.

c) De la suite B_i, déduire une fonction `compte_sous_mots` de type `string -> string -> int`, telle que l'appel `compte_sous_mots s m` compte le nombre de sous-mots de m qui sont égaux à un mot s. La complexité de l'appel `compte_sous_mots s m` devra être du même ordre de grandeur que celle de `filtre s m`.

d) Montrer par un contre-exemple que la suite B_i ne permet pas, à elle seule, de déterminer le nombre de sous-mots d'un mot m que filtre un motif p quelconque.

QUESTION 8. Soient deux mots m et μ, on dit que m diffère de μ d'au plus une erreur dans les trois cas suivants :

- OUBLI : Il existe une décomposition $\mu = m'\alpha m''$ avec $m = m'm''$.
- INSERTION : Il existe une décomposition $\mu = m'm''$ avec $m = m'\alpha m''$.
- SUBSTITUTION : Il existe une décomposition $\mu = m'\alpha m''$ avec $m = m'\beta m''$.

(α et β ci-dessus sont des lettres quelconques.)

a) On se place d'abord dans le cas d'au plus un oubli. Montrer comment reconnaître si un mot m diffère de μ d'au plus un oubli, à l'aide de l'automate $\mathcal{A}(\mu)$ et de deux suites d'ensembles d'états A_i et $A_i^!$. On justifiera le choix de ces suites.

b) Généraliser au cas d'une erreur quelconque.

c) Soient deux chaînes `mu` et `m` qui représentent respectivement les mots μ et m. Écrire une fonction `filtre_erreur` de type `string -> string -> bool`, telle que l'appel `filtre_erreur mu m` répond vrai lorsque m diffère de μ d'au plus une erreur, et faux sinon.

d) Écrire une fonction `sous_mot_erreur` de type `string -> string -> bool`, telle que l'appel `sous_mot_erreur mu m`, répond vrai lorsqu'un sous-mot de m diffère de μ d'au plus une erreur, et faux sinon. La complexité de l'appel `sous_mots_erreur mu m` devra être du même ordre de grandeur que celle de l'appel `filtre_erreur mu m`.

Partie 2-Pascal. Reconnaissance de motifs.

Les mots et les motifs sont représentés en machine par des chaînes de caractères. Formellement les chaînes (de type `string`) sont des mots sur l'ensemble des caractères de la machine (de type `char`). Le traitement des chaînes est peu spécifié en Pascal standard. On est donc conduit à considérer une extension de Pascal. Dans cette extension, les variables de type `string` peuvent contenir n'importe quelle chaîne. En outre, on dispose des primitives dont les déclarations suivent :

```
function length(s:string):integer;
function concat(s1,s2:string):string;
```

La fonction `length` renvoie la longueur de la chaîne passée en argument. La procédure `concat` renvoie la concaténation de ses deux arguments. Une chaîne `s` peut être vue comme un tableau de caractères dont la taille est variable et dont les indices commencent à un. De sorte que, étant

donnés une chaîne `s` de longueur n et un entier `i`, avec $1 \leq i \leq n$, l'expression `s[i]` dénote le i-ème caractère de `s`. Enfin, il est rappelé que les constantes chaînes et caractères ont la même syntaxe. Dans un programme, `'a'` est soit une chaîne de longueur un, soit un caractère, selon le contexte. En outre, les caractères sont ordonnés et les lettres de l'alphabet sont consécutives selon cet ordre.

Le décalage à gauche $\ll$ est réalisé par la primitive `lsl`, le "ou logique" par la primitive `lor`, et le "et logique" par la primitive `land`. Ces trois primitives prennent deux entiers en arguments et renvoient un entier. On rappelle que les entiers machine sont toujours assez grands pour pouvoir représenter les ensembles d'états des automates. On outre, on ne se préoccupera pas d'éventuelles erreurs dues aux débordements de capacité.

Comme on l'a déjà vu, un automate est la donnée d'une taille n, de N_α pour α variant de `a` à `z`, de N_* et de N_ϵ. Ces données seront rangées respectivement dans une variable globale `n_etats`, un tableau d'entiers `alpha`, ainsi que dans deux variables globales `etoile` et `epsilon`, dont les déclarations suivent :

```
alpha : array['a'..'z'] of integer;
etoile,epsilon,n_etats : integer;
```

(Remarquer que, conformément à une pratique courante en Pascal, le tableau `alpha` est indicé par des caractères alphabétiques.)

QUESTION 4. Dans cette question, on suppose que les variables `n_etats`, `alpha`, etc. représentent un automate $\mathcal{A}$ à n états. Soit un entier `etats` qui représente un ensemble d'états A de $\mathcal{A}$, tel que A est clos par transitions instantanées. Soit également une lettre α représentée en machine par le caractère `c`.

a) Écrire les fonctions `mange` et `ferme`, telles que les appels `mange(etats,c)` et `ferme(etats)` renvoient un entier qui encode respectivement les ensembles $\alpha(A)$ et $\epsilon(A)$. On se conformera à :

```
function mange(etats:integer ; c:char):integer;
function ferme(etats:integer):integer;
```

b) Écrire la fonction `etape`, telle que l'appel `etape(etats,c)` renvoie un entier qui encode l'ensemble d'états A' tel que $A \xrightarrow{\alpha} A'$. On se conformera à la déclaration suivante :

```
function etape(etats:integer ; c:char):integer;
```

c) Exprimer la complexité d'une exécution de `etape`, d'abord dans le cas où le motif p ne contient pas de motif optionnel, puis dans le cas général. Dans ce cas général, on pourra borner le coût d'exécution de `etape` en utilisant toute caractéristique jugée pertinente du motif p.

d) Écrire la fonction `etat_final`, telle que l'appel `etat_final(etats)` teste la présence de l'état final e_{n-1} dans l'ensemble d'états A. On se conformera à la déclaration :

```
function etat_final(etats:integer):boolean;
```

QUESTION 5. Le but de cette question est la construction de l'automate $\mathcal{A}(p)$.

a) Écrire la procédure `construire_auto` qui prend en argument une chaîne de caractères `p` représentant un motif p et initialise les variables `n_etats`, `alpha`, `etoile` et `epsilon`, afin qu'elles encodent l'automate $\mathcal{A}(p)$. Dans le cas où la chaîne `p` n'est pas un motif, la construction de l'automate devra échouer. Pour ce faire, on dispose de la primitive `echoue` qui prend un message d'erreur en argument.

On se conformera à la déclaration suivante :

```
procedure construire_auto(p:string);
```

b) Évaluer la complexité de l'appel `construire_auto(p)` en fonction de la longueur de la chaîne `p`.

QUESTION 6. On programme la reconnaissance du filtrage d'un mot par un motif.

a) Écrire la fonction `filtre`, telle que l'appel `filtre(p,m)` teste le filtrage du mot m par le motif p. La fonction `filtre` devra être conforme à la déclaration suivante :

```
function filtre(p,m:string):boolean;
```

b) Donner la complexité de l'appel `filtre(p,m)`, en fonction de la longueur de la chaîne `m` et de toute caractéristique pertinente de p, d'abord dans le cas où p ne contient pas de motif optionnel, puis dans le cas général.

QUESTION 7. Dans cette question, on étudie la reconnaissance des sous-mots d'un mot.

a) Soit p un motif et m un mot. Trouver un motif q tel que q filtre m, si et seulement si p filtre un sous-mot de m. En déduire une fonction `filtre_sous_mot`, telle que l'appel `filtre_sous_mot(p,m)` teste le filtrage d'au moins un sous-mot de m par p. On se conformera à la déclaration suivante :

```
function filtre_sous_mot(p,m:string):boolean;
```

b) On considère un motif p et un mot $m = \alpha_0\alpha_1 \ldots \alpha_{k-1}$. Soit de nouveau la suite d'ensemble d'états $A_0, A_1, \ldots, A_k$ engendrée par la lecture de m par $\mathcal{A}(p)$:

$$A_0 = \epsilon(\{e_0\}), \quad A_{i+1} = \epsilon(\alpha_i(A_i)) \text{ pour } 1 \leq i < k$$

Justifiez brièvement que l'état final de $\mathcal{A}(p)$ appartient à A_i, si et seulement si p filtre un préfixe $m_i = \alpha_0\alpha_1 \ldots \alpha_{i-1}$ de m.

Trouver (en justifiant son choix) une suite $B_0, B_1, \ldots, B_k$, telle que l'état final de $\mathcal{A}(p)$ appartient à B_i, si et seulement si p filtre un sous-mot $\alpha_j\alpha_{j+1} \ldots \alpha_{i-1}$ de m.

c) De la suite B_i, déduire une fonction `compte_sous_mots`, telle que l'appel `compte_sous_mots(s,m)` compte le nombre de sous-mots de m qui sont égaux à un mot s. On se conformera à :

```
function compte_sous_mots(s,m:string):integer;
```

La complexité de l'appel `compte_sous_mots(s,m)` devra être du même ordre de grandeur que celle de `filtre(s,m)`.

d) Montrer par un contre-exemple que la suite B_i ne permet pas, à elle seule, de déterminer le nombre de sous-mots d'un mot m que filtre un motif p quelconque.

QUESTION 8. Soient deux mots m et μ, on dit que m diffère de μ d'au plus une erreur dans les trois cas suivants :

- OUBLI : Il existe une décomposition $\mu = m'\alpha m''$ avec $m = m'm''$.
- INSERTION : Il existe une décomposition $\mu = m'm''$ avec $m = m'\alpha m''$.
- SUBSTITUTION : Il existe une décomposition $\mu = m'\alpha m''$ avec $m = m'\beta m''$.

(α et β ci-dessus sont des lettres quelconques.)

a) On se place d'abord dans le cas d'au plus un oubli. Montrer comment reconnaître si un mot m diffère de μ d'au plus un oubli, à l'aide de l'automate $\mathcal{A}(\mu)$ et de deux suites d'ensembles d'états A_i et A_i^1. On justifiera le choix de ces suites.

b) Généraliser au cas d'une erreur quelconque.

c) Soient deux chaînes `mu` et `m` qui représentent respectivement les mots μ et m. Écrire une fonction `filtre_erreur` telle que l'appel `filtre_erreur(mu,m)` répond vrai lorsque m diffère de μ d'au plus une erreur, et faux sinon. On se conformera à :

```
function filtre_erreur(mu,m:string):boolean;
```

d) Écrire une fonction `sous_mot_erreur`, telle que l'appel `sous_mot_erreur(mu,m)`, répond vrai lorsqu'un sous-mot de m diffère de μ d'au plus une erreur, et faux sinon. On se conformera à :

```
function sous_mot_erreur(mu,m:string):boolean;
```

La complexité de l'appel `sous_mots_erreur(mu,m)` devra être du même ordre de grandeur que celle de l'appel `filtre_erreur(mu,m)`.