Problème 1 - Langages et automates

Dans tout ce problème, les mots sont construits sur un alphabet à deux lettres $\Sigma = \{a, b\}$. Le mot vide est noté ε.

La longueur d'un mot u est notée $|u|$. On dit qu'un langage est pair (resp. impair) lorsque tous les mots du langage sont de longueurs paires (resp. impaires).

Partie 1

On définit une application de Σ^* dans Σ^* de la façon suivante :

— $\Phi(\varepsilon) = \varepsilon$;

— si u est un mot non vide de longueur paire $a_1a_2 \dots a_{2k-1}a_{2k}$, on pose $\Phi(u) = a_2a_1a_4a_3 \dots a_{2k}a_{2k-1}$;

— si u est un mot de longueur impaire $a_1a_2 \dots a_{2k-1}a_{2k}a_{2k+1}$, on pose $\Phi(u) = a_2a_1a_4a_3 \dots a_{2k}a_{2k-1}a_{2k+1}$.

Autrement dit, $\Phi(u)$ est le mot obtenu à partir de u en échangeant chaque lettre d'indice pair avec la lettre qui la précède. Par exemple, $\Phi(a) = a$, $\Phi(ab) = ba$, $\phi(abbaabb) = baabbab$.

Question 1) Soit u un mot de Σ^*. Montrez l'équivalence :

$$|u| \text{ est pair} \quad \iff \quad \forall v \in \Sigma^* \ \Phi(uv) = \Phi(u)\Phi(v)$$

Question 2) Soit L le langage contenant les mots invariants par Φ : $u \in L \iff \Phi(u) = u$.

a) Montrez que L est un langage régulier en donnant une expression régulière qui le définit.

b) Représentez un automate (déterministe ou non) qui reconnaît le langage L.

c) Donnez un automate déterministe complet qui reconnaît L.

Question 3) Soit M le complémentaire de L dans Σ^*.

a) Montrez que M est reconnaissable et donnez un automate qui reconnaît M.

b) Donnez une expression régulière qui définit M.

Question 4) Soit N le langage a^*b^*.

a) Soit $u = a^nb^p$ un mot de N. Précisez $\Phi(u)$ (vous pourrez discuter selon la parité de n).

b) Donnez une expression régulière définissant le langage $\Phi(N)$.

Partie 2

Question 1) Soit L un langage et x une lettre. On note $P_x(L)$ l'ensemble des mots u tels que $ux \in L$.

Montrez que si L est reconnaissable, alors $P_x(L)$ est un langage reconnaissable.

On montre de même (il n'est pas demandé de le faire) que $S_x(L) = \{u \in \Sigma^* \ / \ xu \in L\}$ est aussi un langage reconnaissable.

Question 2) Soit L, L' deux langages non vides. Montrez les équivalences suivantes :

a) L^* est pair si et seulement si L est pair.

b) $L \cup L'$ est pair si et seulement si L et L' sont pairs.

c) LL' est pair si et seulement si L et L' sont des langages pairs ou impairs de même parité.

Question 3) Soit L, L' deux langages impairs. Montrez que $LL' = \bigcup_{x,y \in \Sigma} P_x(L)xyS_y(L')$.

Question 4) On définit par induction les expressions régulières paires :

— $\varnothing$, ε sont des expressions régulières paires ;

— pour tout couple de lettres (x, y), l'expression régulière xy est paire ;

— si e et f sont deux expressions régulières paires, alors e^*, ef et $e|f$ sont des expressions régulières paires.

Montrez par induction que tout langage régulier pair est défini par une expression régulière paire.

Question 5) On définit par induction une application φ sur l'ensemble des expressions régulières paires :
- $\varphi(\varnothing) = \varnothing$, $\varphi(\varepsilon) = \varepsilon$;
- pour tout couple de lettres (x, y), $\varphi(xy) = yx$;
- si e et f sont deux expressions régulières paires, alors $\varphi(e^*) = \varphi(e)^*$, $\varphi(ef) = \varphi(e)\varphi(f)$ et $\varphi(e|f) = \varphi(e)|\varphi(f)$.

Montrez que si L est un langage reconnaissable pair, alors $\Phi(L)$ est aussi reconnaissable.

Question 6) On montre de même (il n'est pas demandé de le faire) que si L est un langage reconnaissable impair, alors $\Phi(L)$ est aussi reconnaissable.

Soit L un langage reconnaissable. Pour $i \in \{0, 1\}$, on pose $L_i = \{u \in L \ / \ |u| \equiv i \quad [2]\}$.

a) Montrez que L_0 et L_1 sont reconnaissables.

b) Montrez que $\Phi(L)$ est reconnaissable.

Problème 2

Un graphe orienté est un couple (S, A) où S est un ensemble fini de sommets et A un ensemble de couples (a, b), appelés arcs, où a et b sont deux sommets de S, distincts : a est l'origine de l'arc et b le but et on note encore $\overrightarrow{ab}$ l'arc (a, b), on dit aussi que b est un fils de a ou que a est un parent de b.

Un chemin dans G est une suite de sommets $(s_0, \ldots, s_p)$ telle que pour tout $i \in [\![1, p-1]\!]$, $\overrightarrow{s_i s_{i+1}}$ est un arc (un chemin peut être réduit à un seul sommet) : s_0 et s_p sont les extrémités du chemin, s_0 en est l'origine, s_p le but, p est la longueur du chemin (*i.e.* le nombre d'arcs). Un cycle est un chemin dont les extrémités sont égales.

Dans ce problème, les sommets du graphe sont représentés par des entiers de 0 à $n-1$ et le graphe est représenté par un tableau de listes d'adjacence g : si i est le numéro d'un sommet, $g.(i)$ est la liste des sommets b tels que $\overrightarrow{ib}$ soit un arc du graphe.

On définit pour cela en CAML les types suivants :

```
type sommet == int;
type graphe == sommet list vect;;
```

Vous pouvez bien sûr écrire des fonctions auxiliaires pour répondre aux différentes questions, mais **votre code ne doit utiliser ni boucle for ou while, ni référence**.

Question 1) Écrivez une fonction `pred gr s` de type `graphe -> sommet -> sommet list`, qui calcule la liste des parents du sommet `s`.

Question 2) Écrivez une fonction `inverse gr` de type `graphe -> graphe`, qui calcule le graphe inverse de `gr`, c'est-à-dire celui où tous les arcs ont été retournés, les origines devenant les buts et vice-versa.

Question 3) On appelle descendants d'un sommet s les sommets t qui sont les extrémités des chemins d'origine s.

Pour calculer l'ensemble des descendants d'un sommet, on utilise une méthode dite de marquage :
- on le place dans une liste l ;
- tant que cette liste n'est pas vide, pour tout sommet de la liste l, on le retire de la liste et s'il n'est pas déjà marqué, on le marque et on place ses fils dans la liste l.

Quand la liste est vide, les sommets marqués sont les descendants du sommet initial.

Un marquage est représenté par un tableau de booléens m initialement tous égaux à `false` : quand i est marqué, on bascule la valeur de $m.(i)$ à `true`.

Écrivez une fonction `marquage gr s` de type `graphe -> sommet -> bool vect`, qui calcule le tableau de marquage permettant de retrouver les descendants du sommet `s` dans le graphe `g`.

Question 4) Estimez la complexité de votre algorithme en fonction de n, nombre de sommets du graphe et m, nombre d'arcs du graphe.

Question 5) En modifiant légèrement votre fonction précédente, donnez une fonction `existe_cycle gr s` de type `graphe -> sommet -> bool`, qui teste s'il existe un cycle passant par le sommet `s`.

Question 6) Une source du graphe est un sommet dont tous les autres sommets sont des descendants. Écrivez une fonction `est_source gr s` de type `graphe -> sommet -> bool * bool vect`, qui calcule le couple `(b,m)` où `b` est un booléen qui indique si le sommet est une source et `m` est le tableau de marquage calculé à partir du sommet `s`.

Question 7) On recherche l'existence ou non d'un sommet source dans le graphe.

a) Montrez que si un sommet n'est pas une source, alors aucun de ses descendants n'en est une.

b) Déduisez-en une fonction `source gr` de type `graphe -> sommet list`, qui calcule la liste vide si le graphe n'a pas de source, une liste à un sommet source si le graphe en possède une. Votre fonction ne devra pas être simpliste et tester tous les sommets un à un...

Question 8) Un puits est un sommet qui est le descendant de tous les autre sommets. Donnez une fonction qui trouve éventuellement un puits dans un graphe.

Problème 3 - Logique

Lors d'une de ses nombreuses expéditions, le célèbre archéologue Maurice Jaunes (un aïeul berrichon d'Indiana Jones) se retrouve coincé dans une salle secrète d'un temple maya. Pour s'en sortir, il découvre un astucieux mécanisme constitué de trois leviers en position verticale, qu'il doit pencher à droite ou à gauche pour ouvrir un sas d'évacuation. Chaque levier est suivi d'une inscription en caractères hiéroglyphes qu'il déchiffre sans peine, le tout surmonté d'une autre inscription attribué au dieu maya de la vérité, réputé pour ne jamais mentir.

On associe aux trois leviers trois variables propositionnelles a, b, c qui représentent l'état du levier correspondant : on associe la valeur "vrai" pour signifier qu'on penche le levier à droite, "faux" sinon.

À côté des leviers L_a, L_b, L_c, Maurice lit les inscriptions suivantes :

A : tu ne dois pas pencher les trois leviers vers la droite ;

B : si l'inscription **A** est vraie, alors penche le levier L_b vers la droite et le levier L_a vers la gauche ;

C : l'inscription **B** est fausse et penche le levier L_a vers la droite.

Enfin, Maurice lit l'inscription attribué au dieu de la vérité, qui donne l'injonction suivante notée **R** :

« tu dois pencher à droite les leviers portant une inscription exacte et uniquement ceux-là »

Question 1)

a) Donnez en fonction de a, b, c une expression des trois formules logiques A, B, C qui traduisent les trois inscriptions.

b) Justifiez par le calcul des propositions que $B \equiv b \wedge (\neg a \vee c)$ (on rappelle les règles de distributivité et d'absorption : $x \wedge (x \vee y) \equiv x \equiv x \vee (x \wedge y)$).

c) Donnez de même une expression équivalente à C qui ne contient que trois termes.

Question 2) Exprimez la formule logique R qui traduit l'injonction du dieu maya en fonction de a, b, c et de A, B, C. Quel type de problème logique Maurice doit-il résoudre ?

Question 3) Sauvez Maurice d'une mort affreuse grâce à une table de vérité.

Devoir surveillé 4 - Corrigé

Problème 1

Partie 1

Question 1) Si $|u|$ est pair et $u \neq \varepsilon$ (pour $u = \varepsilon$, la proposition est évidente), alors $u = a_1 a_2 \dots a_{2k-1} a_{2k}$, donc pour tout mot $v = b_1 b_2 \dots b_p$, on a $uv = a_1 a_2 \dots a_{2k-1} a_{2k} b_1 b_2 \dots b_p$, donc

$\Phi(uv) = a_2 a_1 a_4 a_3 \dots a_{2k} a_{2k-1} b_2 b_1 \dots$ donc $\Phi(uv) = \Phi(u)\Phi(v)$.

Réciproquement, si $|u|$ est impair, alors en notant x la dernière lettre de u, on peut noter $u = wx$ et on choisit y une lettre différente de x ; alors $\Phi(uy) = \Phi(wxy)$, or $|w|$ est pair, donc $\Phi(uy) = \Phi(w)\Phi(xy) = \Phi(w)yx$, tandis que $\Phi(u)\Phi(y) = \Phi(w)xy$, donc $\Phi(uy) \neq \Phi(u)\Phi(y)$: on a donc trouvé un mot v tel que $\Phi(uv) \neq \Phi(u)\Phi(v)$.

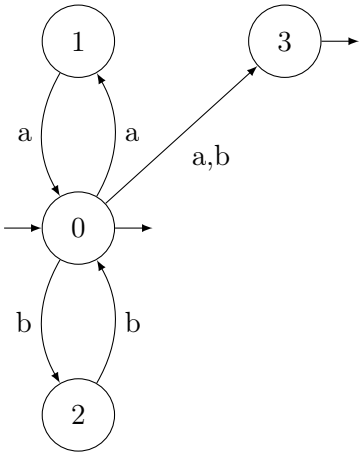
Question 2)

a) Un mot de longueur paire appartient à L si et seulement si il est de la forme $a_1 a_1 a_2 a_2 \dots a_k a_k$, donc appartient au langage défini par l'expression régulière $(a^2|b^2)*$.

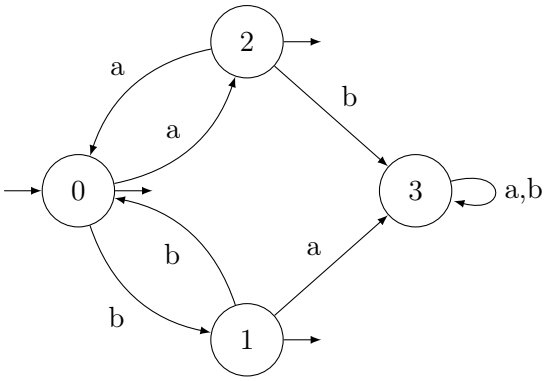
Un mot de longueur impaire appartient à L si et seulement si il est de la forme $a_1 a_1 a_2 a_2 \dots a_k a_k a_{k+1}$, donc appartient au langage défini par l'expression régulière $(a^2|b^2)*(a|b)$.

Comme un mot est soit de longueur paire, soit de longueur impaire, on en déduit que L est défini par l'expression régulière $(a^2|b^2)*(a|b|\varepsilon)$.

b)

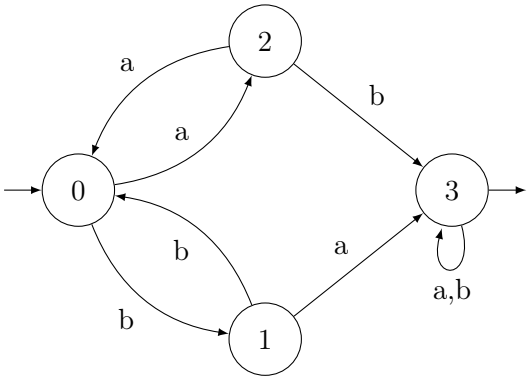


c)



Question 3)

a) On sait que le complémentaire d'un langage reconnaissable est reconnaissable (voir cours) et en plus on sait comment construire un automate qui reconnaît M à partir d'un automate complet qui reconnaît L.



b) On peut obtenir une expression régulière définissant le langage M grâce au lemme d'Arden appliqué plusieurs fois sur l'automate précédent.

On note M_i le langage des mots étiquette d'un calcul partant de l'état i vers l'état final 3. On cherche donc une expression de $M = M_0$. On a alors les équations :

$$\begin{array}{rcl} M_3 & = & (a+b)M_3 + \varepsilon \\ M_1 & = & aM_3 + bM_0 \\ M_2 & = & bM_3 + aM_0 \\ M_0 & = & aM_2 + bM_1 \end{array}$$

L'équation (1) donne $M_3 = (a+b)^*$. On reporte dans les équations (2) et (3), on obtient $M_1 = bM_0 + a(a+b)^*$ et $M_2 = aM_0 + b(a+b)^*$, puis on reporte ces égalités dans l'équation (4). On obtient donc $M_0 = a^2M_0 + ab(a+b)^* + b^2M_0 + ba(a+b)^*$.

Donc $M_0 = (a^2+b^2)M_0 + (ab+ba)(a+b)^*$, donc on en déduit que $M_0 = (a^2+b^2)^*(ab+ba)(a+b)^*$.

Une expression régulière définissant M est donc par exemple $(a^2|b^2)*(ab|ba)(a|b)*$.

Question 4)

a) Si n est pair, alors $\Phi(u) = u$.

Si n est impair et $p \geqslant 1$, on note $n = 2k+1$ donc $u = a^{2k+1}b^p = a^{2k}abb^{p-1}$, donc $\Phi(u) = a^{2k}bab^{p-1}$.

Si n est impair et $p = 0$, alors $\Phi(u) = u$.

b) $\Phi(N)$ contient donc exactement les mots de la forme :
— $a^{2k}b^p$
— a^{2k+1}
— $a^{2k}bab^p$

Une expression régulière de $\Phi(N)$ est donc par exemple $(a^2)*b* \,|\, (a^2)*a \,|\, (a^2)*bab*$, qu'on peut encore réécrire sous la forme $(a^2)*(b*|a|ba\,b*)$

Partie 2

Question 1) Si L est reconnaissable, alors on considère un automate déterministe qui reconnaît L : (E, i, T, δ) où E l'ensemble des états et T l'ensemble des états finaux, i l'état initial et δ la fonction de transition.

Si x est une lettre qui ne termine aucun mot de L, alors $P_x(L) = \varnothing$ donc est reconnaissable.

Si x est une lettre qui termine au moins un mot de L, alors pour chaque état final q de l'automate, on détermine les états p tels qu'il existe une transition de p vers q étiqueté par x : on pose

$$T' = \{p \in E \;/\; \exists q \in T \;\delta(p,x) = q\}$$

L'automate (E, i, T', δ) reconnaît le langage $P_x(L)$.

Question 2)

a) Si L est pair, alors tous les mots de L sont de longueurs paires, donc en les concaténant, on obtient toujours des mots de longueurs paires, donc L^* est pair.

De plus, $L \subset L^*$, donc si L^* est pair, alors L est pair.

b) Si L et L' sont pairs, alors tous leurs mots sont de longueurs paires, donc tous les mots de $L \cup L'$ le sont aussi : $L \cup L'$ est pair.

De plus, $L \subset L \cup L'$, donc si $L \cup L'$ est pair, alors L est pair. Et de même pour L'.

c) Si L et L' ont la même parité, alors en concaténant un mot de L et un mot de L', on obtient un mot dont la longueur est la somme de deux entiers pairs ou de deux entiers impairs, donc dans les deux cas il est de longueur paire. Donc LL' est pair.

Réciproquement, si LL' est pair, alors comme L et L' sont non vides, on choisit un mot u dans L et un mot v dans L', donc le mot $uv \in LL'$ est de longueur paire : $|u| + |v| \equiv 0 \mod 2$, donc les longueurs de u et v ont la même parité : $|u| \equiv |v| \mod 2$.

Pour tout mot v' de L', le même raisonnement s'applique, donc $|u| \equiv |v'| \mod 2$: tous les mots de L' ont donc la même parité, qui est celle de $|u|$.

Et de même, tous les mots de L ont la même parité, qui est celle de $|u|$. Donc finalement, L et L' sont des langages pairs ou impairs de même parité.

Question 3) L et L' étant impairs, il ne contiennent pas le mot vide, donc tout mot de L se terminant par une lettre, on en déduit immédiatement que $L = \bigcup_{x \in \Sigma} P_x(L)x$. De même, $L' = \bigcup_{y \in \Sigma} yS_y(L')$.

Donc en utilisant la distributivité de la concaténation pour l'union, on en déduit que

$$LL' = \bigcup_{x \in \Sigma} P_x(L)x \bigcup_{y \in \Sigma} yS_y(L') = \bigcup_{x,y \in \Sigma} P_x(L)xyS_y(L').$$

Question 4) Si e est un expression régulière, on note $\mathscr{L}(e)$ le langage représenté par e.

Soit L un langage régulier pair. L n'est pas réduit à un singleton-lettre car il est pair.

Cas de base :

— Si $L = \varnothing$, alors L est défini par l'expression régulière $\varnothing$.

— Si $L = \{\varepsilon\}$, alors L est défini par l'expression régulière ε.

Cas récursif : si L est le langage représenté par l'expression régulière e, alors e est de la forme f^*, fg ou $f|g$, où f et g sont des expressions régulières.

— Si $e = f^*$, alors $L = \mathscr{L}(f^*) = \mathscr{L}(f)^*$ est un langage pair, donc d'après la question 2, $\mathscr{L}(f)$ est pair, donc par hypothèse d'induction, $\mathscr{L}(f)$ est représenté par une expression régulière paire $\tilde{f}$, donc $L = \mathscr{L}(\tilde{f}^*)$ est représenté par une expression régulière paire.

— Si $e = f|g$, alors $L = \mathscr{L}(f|g) = \mathscr{L}(f) \cup \mathscr{L}(g)$ est un langage pair, donc d'après la question 2, $\mathscr{L}(f)$ et $\mathscr{L}(g)$ sont pairs, donc par hypothèse d'induction, $\mathscr{L}(f)$ est représenté par une expression régulière paire $\tilde{f}$ et de même pour $\mathscr{L}(g) = \mathscr{L}(\tilde{g})$, donc $L = \mathscr{L}(\tilde{f}|\tilde{g})$ est représenté par une expression régulière paire.

— Si $e = fg$, alors $L = \mathscr{L}(fg) = \mathscr{L}(f)\mathscr{L}(g)$ est un langage pair, donc d'après la question 2, $\mathscr{L}(f)$ et $\mathscr{L}(g)$ sont de même parité.

  — S'ils sont tous les deux pairs, alors par hypothèse d'induction, $\mathscr{L}(f)$ est représenté par une expression régulière paire $\tilde{f}$ et de même pour $\mathscr{L}(g) = \mathscr{L}(\tilde{g})$, donc $L = \mathscr{L}(\tilde{f}\tilde{g})$ est représenté par une expression régulière paire.

  — S'ils sont tous les deux impairs, alors on écrit $L = \bigcup_{x,y \in \Sigma} P_x(\mathscr{L}(f))xyS_y(\mathscr{L}(g))$. Dans cette expression, $P_x(\mathscr{L}(f))$ et $S_y(\mathscr{L}(g))$ sont pairs et réguliers, donc par hypothèse d'induction, ils sont représentés par des expressions régulières paires $\tilde{f}_x$ et $\tilde{g_y}$, donc $L = \bigcup_{x,y \in \Sigma} \tilde{f}_x \, xy \, \tilde{g_y}$ est représenté par une expression régulière paire.

D'après le principe d'induction, tout langage régulier pair est représenté par une expression régulière paire.

Question 5) Un langage reconnaissable est régulier et réciproquement.

Soit donc L un langage régulier pair. D'après la question précédente, il est représenté par une expression régulière paire e. Or par définition de φ, il est presque évident que si e est une expression régulière paire, alors $\Phi(\mathscr{L}(e)) = \mathscr{L}(\varphi(e))$. Donc $\Phi(L) = \mathscr{L}(\varphi(e))$ est un langage régulier donc reconnaissable.

Question 6)

a) $L_0 = L \cap \left(\Sigma^2\right)^*$ est l'intersection de deux langages reconnaissables donc est reconnaissable.

$L_1 = L \cap \left(\Sigma^2\right)^* \Sigma$ est de même un langage reconnaissable.

b) $L = L_0 \cup L_1$ donc $\Phi(L) = \Phi(L_0) \cup \Phi(L_1)$.

L_0 est reconnaissable et pair donc $\Phi(L_0)$ l'est aussi d'après la question 4. L_1 est reconnaissable et impair donc $\Phi(L_1)$ l'est aussi d'après la remarque de l'énoncé.

$\Phi(L)$ est la réunion de deux langages reconnaissables, donc il est reconnaissable.

Problème 2

Question 1)

```
let pred gr s =
  let n = vect_length gr in
  let rec predaux k l =
    if k = n then l
    else predaux (k+1) (if mem s gr.(k) then k :: l else l)
  in
  predaux 0 [];;
```

Question 2)

```
let inverse gr =
  let n = vect_length gr in
  let rg = make_vect n [] in
  let rec invaux k =
    if k = n then ()
    else begin rg.(k) <- pred gr k; invaux (k+1) end
  in
  invaux 0; rg;;
```

Question 3) En fait, le sujet nous propose de programmer un parcours en profondeur d'un graphe à partir d'un sommet. C'est du cours.

```
let marquage gr s =
  let n = vect_length gr in
  let t = make_vect n false in
  let rec marqaux l =
    match l with
    | [] -> t
    | x :: q ->
        if not t.(x) then
          begin t.(x) <- true; marqaux (gr.(x) @ q) end
        else
          marqaux q
  in
  marqaux [s];;
```

Question 4) D'après le cours, un parcours en profondeur avec marquage peut être fait en un coût en $O(n+m)$.

Question 5)

```
let existe_cycle gr s =
  let n = vect_length gr in
  let t = make_vect n false in
  let rec cyclaux l =
    match l with
    | [] -> false
    | x :: q ->
        x = s ||
        if not t.(x) then
```

```
        begin t.(x) <- true; cyclaux (gr.(x) @ q) end
      else
        cyclaux q
  in
  t.(s) <- true; cyclaux gr.(s);;
```

Question 6)

```
let rec est_source gr s =
  let t = marquage gr s in
  let n = vect_length t in
  let rec aux k =
    k = n || (t.(k) && aux (k+1))
  in
  aux 0, t;;
```

Question 7)

a) Si un descendant t d'un sommet s est une source du graphe, alors pour tout autre sommet u, il existe un chemin de t à u et il existe un chemin de s à t, donc en concaténant les chemins, il existe un chemin de s à u : s est donc une source.

Par contraposée, on a montré ce qu'on veut.

b)

```
let rec liste n =
  match n with
  | 0 -> []
  | _ -> (n -1) :: liste (n-1);;

let rec retirer l m =
  match l with
  | [] -> []
  | x :: q -> let l' = retirer q m in
     if m.(x) then l' else x :: l';;

let source gr =
  let candidat = liste (vect_length gr) in
  let rec sourcaux l =
    match l with
    | [] -> []
    | x :: q -> let b, m = est_source gr x in
                if b then [x]
                else let l' = retirer l m in
                  sourcaux l'
  in
  sourcaux candidat;;
```

Question 8)

```
let puits gr =
  let rg = inverse gr in
  source rg;;
```

Problème 3

Question 1)

a) $A = \neg(a \wedge b \wedge c)$,
$B = \ A \Rightarrow (\neg a \wedge b) = (a \wedge b \wedge c) \vee (\neg a \wedge b)$,
$C = \neg B \wedge a = \neg\left[(a \wedge b \wedge c) \vee (\neg a \wedge b)\right] \wedge a$.

b)
$$\begin{aligned}
B &\equiv (a \wedge b \wedge c) \vee (\neg a \wedge b) \\
&\equiv [(a \wedge b \wedge c) \vee \neg a] \wedge [(a \wedge b \wedge c) \vee b] && \text{distributivité de } \vee \text{ pour } \wedge \\
&\equiv [(a \vee \neg a) \wedge (b \vee \neg a) \wedge (c \vee \neg a)] \wedge [(a \wedge b \wedge c) \vee b] && \text{distributivité de } \vee \text{ pour } \wedge \\
&\equiv [(b \vee \neg a) \wedge (c \vee \neg a)] \wedge b && \text{absorption} \\
&\equiv [b \wedge (b \vee \neg a)] \wedge (c \vee \neg a) && \text{commutativité et associativité de } \wedge \\
&\equiv b \wedge (c \vee \neg a) && \text{absorption}
\end{aligned}$$

c) De même, on trouve $C \equiv a \wedge (\neg b \vee \neg c)$.

Question 2) $R = (A \iff a) \wedge (B \iff b) \wedge (C \iff c)$.

Il s'agit d'un problème de satisfiabilité : on veut trouver une assignation des variables a, b, c qui satisfait la formule R, c'est-à-dire que sous cette assignation, la formule R a pour valeur 1.

Question 3) On dresse la table de vérité des formules A, B, C et R :

a	b	c	A	B	C	$A \iff a$	$B \iff b$	$C \iff c$	R
0	0	0	1	0	0	0	1	1	0
0	0	1	1	0	0	0	1	0	0
0	1	0	1	1	0	0	1	1	0
0	1	1	1	1	0	0	1	1	0
1	0	0	1	0	1	1	1	0	0
1	0	1	1	0	1	1	1	1	1
1	1	0	1	0	1	1	0	0	0
1	1	1	0	1	0	0	1	0	0

On constate qu'il n'existe qu'une seule assignation qui satisfait la formule R. Maurice doit donc pencher les leviers L_a et L_c à droite et le levier L_b à gauche.