Option Informatique en Spé MP et MP*
DS du 3 décembre 2002 : le corrigé

Logique

Question 1 On dresse un tableau donnant les valeurs des cinq clauses pour chacune des huit assignations possibles de x, y et z :

x	y	z	$x \vee \neg y \vee z$	$\neg x \vee \neg y \vee z$	$\neg x \vee y \vee \neg z$	$x \vee y \vee \neg z$	$\neg x \vee \neg y \vee \neg z$
faux	faux	faux	vrai	vrai	vrai	vrai	vrai
faux	faux	vrai	vrai	vrai	vrai	**faux**	vrai
faux	vrai	faux	**faux**	vrai	vrai	vrai	vrai
faux	vrai	vrai	vrai	vrai	vrai	vrai	vrai
vrai	faux	faux	vrai	vrai	vrai	vrai	vrai
vrai	faux	vrai	vrai	vrai	**faux**	vrai	vrai
vrai	vrai	faux	vrai	**faux**	vrai	vrai	vrai
vrai	vrai	vrai	vrai	vrai	vrai	vrai	**faux**

On constate que les assignations qui satisfont simultanément les cinq clauses sont :

- $x =$ faux, $y =$ faux et $z =$ faux ;
- $x =$ faux, $y =$ vrai et $z =$ vrai ;
- $x =$ vrai, $y =$ faux et $z =$ faux.

On peut aller plus vite en notant que chacune des cinq clauses, en faisant intervenir une et une seule fois chacune des trois variables, interdit une assignation parmi les huit possibles ; les trois qui restent sont les bonnes.

Question 2 Par exhaustion, il faudrait examiner seize cas. x, y et z jouent des rôles symétriques ; ceci permet d'accélérer l'étude, en discutant selon le nombre p de variables instanciées à vrai parmi x, y et z. Résumons par un tableau ; q est le nombre de clauses vraies, parmi $(\neg x) \vee (\neg y)$ et consorts ; r (resp. s) est le nombre de clauses vraies parmi $x \vee (\neg w)$ et consorts lorsque w est instanciée à vrai (resp. faux). t_{vrai} (resp. t_{faux}) est le nombre total de clauses vraies parmi les dix de l'ensemble F lorsque w est instanciée à vrai (resp. faux) : on a $t_{\text{vrai}} = p + q + r + 1$ et $t_{\text{faux}} = p + q + s$. Enfin, t est le nombre maximal de clauses vraies parmi les dix.

p	q	r	s	t_{vrai}	t_{faux}	t
0	3	0	3	4	6	6
1	3	1	3	6	7	7
2	2	2	3	7	7	7
3	0	3	3	7	6	7

Résumons : on peut satisfaire au plus sept clauses parmi les dix.

Circuits logiques

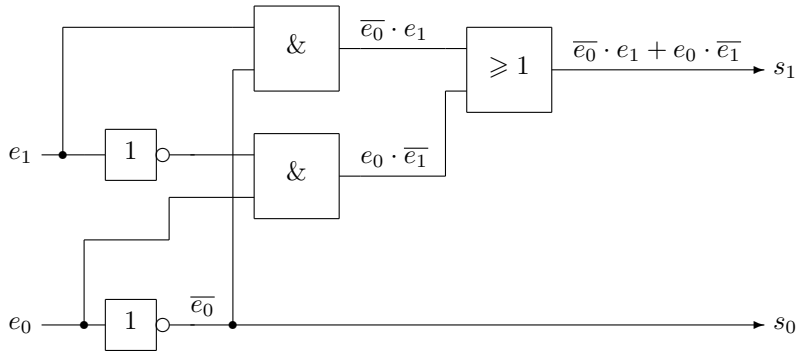
Question 1 Le circuit en question est l'inverseur ; en effet, $x + 1 = 1 - x = \overline{x}$, modulo 2.

Question 2 Il suffit de remarquer que s_1 est le quotient, et s_0 le reste, dans la division de $e_0 + 2e_1 + 1$ par 2.

e_0	e_1	$e_0 + 2e_1 + 1$	$e_0 + 2e_1 + 1 \pmod 4$	s_0	s_1
0	0	1	1	1	0
1	0	2	2	0	1
0	1	3	3	1	1
1	1	4	0	0	0

Question 3 $s_0 = \overline{e_0}$ et $s_1 = \overline{e_0} \cdot e_1 + e_0 \cdot \overline{e_1}$.

Question 4 Circuit dessiné sans aucune finesse.

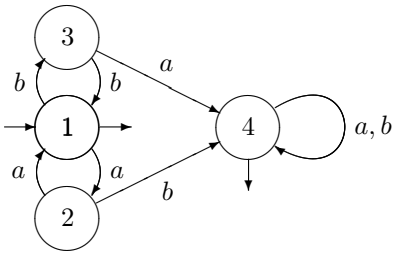


Langages rationnels et automates finis

Question 1 • L_1 est l'ensemble des mots u de longueur $n \geqslant 2$, tels qu'il existe k vérifiant $2 \leqslant 2k \leqslant n$ et $u_{2k-1} \neq u_{2k}$.

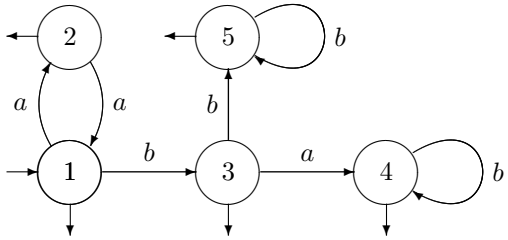
Question 2 • Montrons que L_1 est décrit par l'expression rationnelle $e = (aa + bb)^*(ab + ba)(a + b)^*$. Soit $u \in L_1$, de longueur $n \geqslant 2$. Notons k le plus petit indice tel que $2 \leqslant 2k \leqslant n$ et $u_{2k-1} \neq u_{2k}$. Alors $u_{2i-1} = u_{2i}$ pour $1 \leqslant i < k$. En décomposant u comme produit des trois mots $u[1..2k-2]$, $u_{2k-1}u_{2k}$ et $u[2k+1..n]$, nous constatons que u appartient au langage décrit par e. Réciproquement, soit $u \in \mathcal{L}_{ER}(e)$: il existe un indice k tel que $2 \leqslant 2k \leqslant n$ et $u_{2k-1}u_{2k} \in \{ab,ba\}$; donc $u \in L_1$.

Question 3 • La traduction de l'expression rationnelle précédente en un a.f.d.c. est immédiate.
Nous constatons qu'un mot u est l'étiquette d'un calcul réussi de cet automate ssi u se compose de zéro, un ou plusieurs facteurs de la forme aa ou bb, suivis d'un facteur ab ou ba (menant à l'état 4), ce dernier suivi d'un facteur quelconque : ceci montre que cet automate reconnaît exactement L_1.



Question 4 • Soit $u \in L_2$: $u = a^p b^q$. Il y a deux cas de figure : si p est pair ou si $q = 0$, alors $\varphi(u) = u$. Sinon, $u = a^{p-1}abb^{q-1}$; $p-1$ est pair, donc $\varphi(u) = a^{p-1}bab^{q-1}$. Réciproquement, tout mot de cette forme est dans $\varphi(L_2)$. Finalement, ce langage est décrit par l'expression rationnelle $a^* + (aa)^*b^* + (aa)^*bab^*$, laquelle peut se simplifier en $(aa)^*(a + b^* + (ba)b^*)$.

Question 5 • L'automate non déterministe ci-contre reconnaît L_2. Pour le voir, notons $\overline{k}$ l'état k, considéré comme non final. Alors :



- le sous-automate constitué des états 1 et 2 reconnaît le langage décrit par a^* ;
- le sous-automate formé des états $\overline{1}$, $\overline{2}$, 3 et 5 reconnaît le langage décrit par $(aa)^*b^*$: le calcul se termine dans l'état 3 pour les mots décrits par $(aa)^*b$, dans l'état 5 pour les autres ;
- le sous-automate formé des états $\overline{1}$, $\overline{2}$, $\overline{3}$ et 4 reconnaît le langage décrit par $(aa)^*bab^*$.

Question 6 • L'ensemble P des mots de longueur paire est rationnel : il est décrit par l'expression rationnelle $(aa + ab + ba + bb)^*$. Donc $L_p = L \cap P$ est rationnel. Du coup, $L_i = L \setminus L_p$ est lui aussi rationnel.

Question 7 • Soit $\mathcal{A} = (Q, \delta, i, F)$ un a.f.d.c. reconnaissant L_p. Définissons l'automate non déterministe $\mathcal{B} = (Q', \delta', i', F')$ où :

- $Q' = Q \times \{1, 2\}$;
- $i' = (i, 1)$;
- $F' = F \times \{1\}$;

- pour chaque paire de transitions $q \xrightarrow{x} q'$ et $q' \xrightarrow{y} q''$ appartenant à δ, δ' contient les transitions $(q,1) \xrightarrow{y} (q',2)$ et $(q',2) \xrightarrow{x} (q'',1)$.

Il est clair que le mot $u = u_1 \dots u_{2n}$ est l'étiquette d'un calcul de $\mathcal{A}$ menant d'un état q à un état q' ssi le mot $\varphi(u) = u_2u_1u_4u_3 \dots u_{2N}u_{2n-1}$ est l'étiquette d'un calcul de $\mathcal{B}$ menant de l'état $(q,1)$ à l'état $(q',1)$. De plus, ε est reconnu par $\mathcal{A}$ ssi $i \in F$, soit $i' \in F'$, ce qui revient à dire que ε est reconnu par $\mathcal{B}$. Ceci montre que $\mathcal{B}$ reconnaît $\varphi(L_p)$.

Question 8 • Nous allons donner deux preuves : la première consiste à construire un automate fini reconnaissant M_x à partir d'un automate reconnaissant L_i ; la deuxième consiste à construire une expression rationnelle décrivant M_x, à partir d'une expression décrivant L_i.

• Première méthode. Soit $\mathcal{C} = (Q, \delta, i, F)$ un a.f.d.c. reconnaissant L_i. Construisons un automate $\mathcal{D} = (Q, \delta, i, F')$ qui reconnaît M_a : nous placerons dans F' tout état $q \in Q$ tel qu'il existe une transition $q \xrightarrow{a} q'$ de $\mathcal{C}$ avec $q' \in F$. Soit $u \in M_a$: alors $ua \in L_i$, donc ua est l'étiquette d'un calcul réussi de $\mathcal{C}$, qui se termine dans un état $q' \in F$. La dernière transition de ce calcul est de la forme $q \xrightarrow{a} q'$, donc $q \in F'$. Par suppression de cette dernière transition, nous obtenons un calcul réussi de $\mathcal{D}$, d'étiquette u. Réciproquement, soit u l'étiquette d'un calcul réussi de $\mathcal{D}$: ce calcul se termine dans un état $q \in F'$; notant $q' = \delta(q,a)$ nous avons $q' = \delta^*(i, ua) \in F$, donc ua est l'étiquette d'un calcul réussi de $\mathcal{C}$, donc $ua \in L_i$ d'où $u \in M_a$. Finalement, nous avons montré que $\mathcal{D}$ reconnaît exactement M_a, donc ce langage est rationnel.

• Deuxième méthode. Nous noterons $\mathcal{L}_{ER}(e)$ le langage décrit par l'expression rationnelle e. Définissons sur l'ensemble des expressions rationnelles une première fonction λ :

- $\lambda(\emptyset) = \emptyset$; $\lambda(\varepsilon) = \varepsilon$; $\lambda(y) = \emptyset$ pour toute lettre $y \in \Sigma$;
- $\lambda(e + e') = \lambda(e) + \lambda(e')$; $\lambda(e \cdot e') = \lambda(e) \cdot \lambda(e')$; $\lambda(e^*) = \varepsilon$.

Il est clair que $\mathcal{L}_{ER}\big(\lambda(e)\big) = \mathcal{L}_{ER}(e) \cap \{\varepsilon\}$. Définissons maintenant une deuxième fonction ψ_x :

- $\psi_x(\emptyset) = \emptyset$; $\psi_x(\varepsilon) = \emptyset$;
- $\psi_x(x) = x$, et $\psi_x(y) = \emptyset$ pour toute lettre $y \in \Sigma \setminus \{x\}$;
- $\psi_x(e + e') = \psi_x(e) + \psi_x(e')$;
- $\psi_x(e \cdot e') = e \cdot \psi_x(e') + \psi_x(e) \cdot \lambda(e')$;
- $\psi_x(e^*) = e^* \cdot \psi_x(e)$.

On se convaincra aisément que si e décrit un langage S, alors $\psi_x(e)$ décrit le langage $\Psi_x(S) = \{u \in \Sigma^* \mid ux \in S\}$. En particulier, si e décrit L_i, alors $\mathcal{L}_{ER}\big(\psi_x(e)\big) = \Psi_x\big(\mathcal{L}_{ER}(e)\big) = \Psi_x(L_i) = M_x$.

Question 9 • D'après la définition de M_a et M_b, nous avons $L_i = M_a\{a\} \cup M_b\{b\}$; nous en déduisons $\varphi(L_i) = \varphi(M_a\{a\}) \cup \varphi(M_b\{b\})$. Nous avons $\varphi(ux) = \varphi(u)x$ pour tout mot u de longueur paire et toute lettre x ; donc $\varphi(M_x\{x\}) = \varphi(M_x)\{x\}$. Conclusion : $\varphi(L_i) = \varphi(M_a)\{a\} \cup \varphi(M_b)\{b\}$.

Question 10 • L_p est rationnel ; le résultat de la question 7 montre que $\varphi(L_p)$ est lui aussi rationnel. Le résultat de la question 8 montre que $\varphi(M_a)$ et $\varphi(M_b)$ sont rationnels ; l'égalité démontrée à la question 9 montre que $\varphi(L_i)$ est rationnel. Comme L est la réunion (disjointe) de L_p et L_i, nous pouvons conclure : $\varphi(L)$ est rationnel, en tant que réunion des deux langages rationnels $\varphi(L_p)$ et $\varphi(L_i)$.

Question 11 • La réponse est négative, puisque φ est involutive !

Question 12 • L'écriture de gauche ci-dessous est immédiate mais ne répond pas exactement à la question, car la fonction obtenue a pour type `'a list -> 'a list`. Pour remédier à ceci, nous forçons le typage dans l'un des deux motifs. Une autre technique consisterait à faire apparaître les trois motifs qui assurent la terminaison, soit `[A]`, `[B]` et `[]`.

```
let rec phi = function
 | x::y::q -> y::x::(phi q)
 | u -> u ;;
```

```
let rec phi = function
 | x::y::q -> y::x::(phi q)
 | (_:lettre list) as u -> u ;;
```

Algorithmique : le problème de partition

Question 1 • Notons $\mathcal{P}(X,k)$ le problème consistant à exhiber une partie Y de X vérifiant $\|Y\| = k$. Avec cette notation, le problème de partition est $\mathcal{P}\big(X, \|X\|/2\big)$. Fixons donc $k = \|X\|/2$ et présentons à l'oracle le couple (X,k) : si sa réponse est négative, inutile de continuer. Sinon, choisissons $x \in X$, et présentons à l'oracle le couple (Y,k) où $Y = X \setminus \{x\}$; deux cas peuvent se produire :

- si sa réponse est positive, nous éliminons x, et il nous reste à résoudre le problème $\mathcal{P}(Y,k)$;
- sinon, nous gardons x et il nous reste à résoudre le problème $\mathcal{P}(Y,k-x)$.

Notons k_i la valeur de k et G_i l'ensemble des éléments gardés après i étapes ; on vérifie sans peine que $k_i + \|G_i\|$ est constante. Au bout de n étapes, chacun des éléments de X aura été gardé ou éliminé ; et $k_n = 0$. Donc $\|G_n\| = k$. Nous aurons consulté $n+1$ fois l'oracle. Il est clair que l'on peut arrêter les frais aussitôt que $k_i = 0$.

Question 2 • `toto` est une fonction à deux arguments. `s = 0` indique que le premier est de type `int` ; le deuxième est de type `int list` au vu du filtrage et de `s = t` ; enfin, le résultat est de type `bool` au vu de `s = 0`. On vérifie sans peine que les différents motifs sont cohérents. Concluons : `toto` est de type `int -> int list -> bool`.

Question 3 • Nous dirons que s est réalisable avec la liste x lorsque l'on peut extraire de x une sous-liste dont la somme est s. Nous allons prouver, par induction structurelle sur la liste x, que `toto s x` rend la valeur `true` ssi s est réalisable avec la liste x.

- si x est la liste vide, le résultat est clair : la seule somme réalisable est 0.
- sinon, soit t la tête de la liste x et q sa queue. Distinguons deux cas de figure :
  - si $s < t$, alors t ne peut servir à réaliser s ; donc s est réalisable avec la liste x ssi il peut l'être avec la liste q ;
  - si $s \geqslant t$, alors s est réalisable en faisant éventuellement intervenir t (c'est le cas ssi $s-t$ est réalisable avec q) ou sans faire intervenir t (c'est le cas ssi s est réalisable avec q).

Question 4 • L'essentiel du travail est fait par la fonction `réaliser_total` : appliquée au couple `(c,x)`, elle extrait de la liste x une sous-liste dont la somme est égale à c, en appliquant les règles suivantes :

- si $c = 0$, la liste vide répond à la question ;
- sinon : on ne peut réaliser $c \neq 0$ avec la liste vide ;
- sinon : $c \neq 0$ et $x = t :: q$; si c est réalisable sans avoir recours à t, faisons-le !
- sinon, on réalise $c - t$ avec la liste q, et on adjoint t au résultat.

Pas de remarques particulières concernant la programmation :

```
exception Somme_impaire ;;
exception Partition_impossible ;;

let rec réaliser_total = function
 | (0,_) -> []
 | (_,[]) -> raise Partition_impossible
 | (c,t::q) when toto c q -> réaliser_total (c,q)
 | (c,t::q) -> t::(réaliser_total (c-t,q)) ;;

let partition l =
 let s = sum l in
 if (s mod 2) = 1
  then raise Somme_impaire
  else réaliser_total (s/2,l) ;;
```

Question 5 • Il n'est pas nécessaire de tester les parties qui contiennent x_n : en effet, dans chaque partition de X en deux parts, l'une des deux ne contient pas x_n. Ceci économise très précisément 50 % des calculs !

Question 6 • $m(0,j) =$ `vrai` s'il existe une partie Y de $\{x_0\}$ vérifiant $\|Y\| = j$; or $\{x_0\}$ n'a que deux parties, l'ensemble vide (de somme nulle) et lui-même (de somme x_0). Donc tous les coefficients de cette ligne ont la valeur `faux`, sauf ceux des colonnes 0 et x_0.

Question 7 • Si $m(i,j) =$ `vrai`, alors $m(i+1,j) =$ `vrai` à plus forte raison. Sinon, $m(i+1,j) =$ `vrai` ssi $j \geqslant x_{i+1}$ et $j - x_{i+1}$ est réalisable avec une partie de $\{x_0, \ldots, x_i\}$: donc $m(i+1,j) = m(i, j - x_{i+1})$.

Question 8 • Le problème de décision admet une réponse positive ssi $m(n-1, \|X\|/2) =$ `vrai` ; ceci sous réserve que $\|X\|$ soit pair, bien entendu !

Question 9 • Observons la colonne d'indice $\|X\|/2$: le coefficient du bas vaut `vrai`. Déterminons le premier indice i tel que $m(i, \|X\|/2) =$ `vrai` : x_i appartient à la solution. Nous recommençons, en observant cette fois la colonne d'indice $\|X\|/2 - x_i$, et en remontant à partir du coefficient de la ligne i ; de proche en proche, nous reconstituons un sous-ensemble Y de X vérifiant $\|Y\| = \|X\|/2$.

Question 10 • $n = 2$ et $\|X\| = 10$. Pour faciliter la lecture, seuls sont marqués (par un **t**) les coefficients de la matrice égaux à `vrai`.

$i\downarrow \quad j\rightarrow$	0	1	2	3	4	5	6	7	8	9	10
0	**t**					**t**					
1	**t**		**t**			**t**		**t**			
2	**t**		**t**	**t**		**t**		**t**	**t**		**t**

Question 11 • La fonction suivante applique le schéma que nous venons d'exposer aux questions 6 et 7.

```
let partition_dynamique x =
 let (n,v,s) = (list_length x , vect_of_list x , sum x) in
 let m = make_matrix n (s+1) false in
  m.(0).(0) <- true ; m.(0).(v.(0)) <- true ;
  for i = 1 to n-1 do
   for j = 0 to s do
    m.(i).(j) <- m.(i-1).(j) or (j >= a.(i) & m.(i-1).(j-v.(i)))
   done;
  done ;
 m ;;
```

Question 12 • L'argument de la question 5 s'applique tel quel : nous pouvons éviter de calculer la dernière ligne ; mais cette économie est négligeable. Plus intéressant : nous pouvons nous arrêter dès que nous trouvons un indice i tel que $m(i, \|X\|/2) =$ `vrai`. En fait, la véritable économie repose sur la remarque suivante : comme le résultat se trouve dans la case $m(n-1, \|X\|/2)$ et ne dépend que des cases situées au-dessus et/ou à gauche de celle-ci, il est inutile de remplir les colonnes d'indice supérieur à $\|X\|/2$. Enfin, notons que la première colonne ne sert à rien ...

Références bibliographiques

▶ L'exercice de logique a été posé à l'oral du concours d'entrée à Cachan.

▶ L'exercice sur les circuits combinatoires provient du sujet du concours d'entrée à l'école de l'Air, session 2002.

▶ Les deux derniers exercices sont basés sur des suggestions de Philippe Espéret.

▶ On trouvera des compléments sur le problème de partition dans le livre *Computers and intractability*, de Garey et Johnson.

FIN