Option Informatique en Spé MP et MP*
DS du 1er février 2000 : le corrigé

Comparateurs

Question 1 • Ici, $A = a_0$, $B = b_0$ donc $A = B$ ssi $a_0 = b_0$, si bien que $\boxed{e = a_0b_0 + \overline{a_0}\,\overline{b_0}}$. $A > B$ ssi $a_0 > b_0$, soit $a_0 = 1$ et $b_0 = 0$, donc $\boxed{s = a_0\overline{b_0}}$.

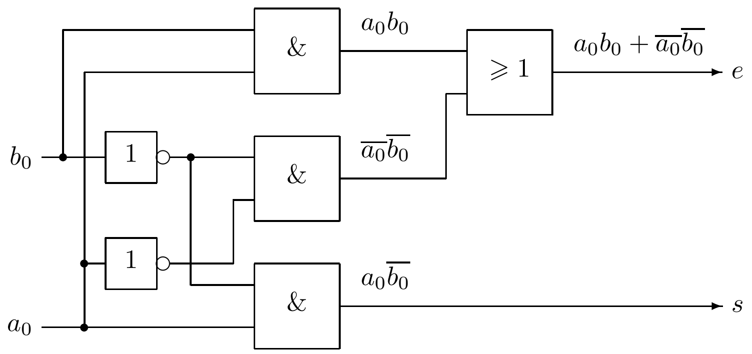

Figure 1: comparateur 1 bit

Le comparateur 1 bit utilise six portes, et son délai est égal à trois. Dans la suite, nous considérerons ce circuit comme une porte logique notée COMP, ayant deux entrées et deux sorties, et représentée comme l'indique la figure 2.

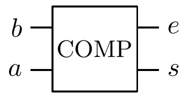

Figure 2: le symbole du comparateur 1 bit

Question 2 • Notons $A_k = \sum_{0 \leqslant i \leqslant k} 2^i a_i$: $A_k = 2^k a_k + A_{k-1}$ et $0 \leqslant A_k < 2^k$, si bien que a_k et A_{k-1} sont respectivement le quotient et le reste dans la division de A_k par 2^k. Définissons de même B_k. Alors $A_k = B_k$ ssi $a_k = b_k$ et $A_{k-1} = B_{k-1}$, si bien que $\boxed{e_k = e_{k-1}(a_kb_k + \overline{a_k}\,\overline{b_k})}$.

• De même, $A_k > B_k$ ssi $a_k > b_k$, ou $a_k = b_k$ et $A_{k-1} > B_{k-1}$; par suite, $s_k = a_k\overline{b_k} + s_{k-1}(a_kb_k + \overline{a_k}\,\overline{b_k})$. Il est clair qu'une partie des calculs peut être assurée par un circuit COMP identique à celui réalisé à la question 1 : ses entrées seront a_k et b_k ; notant ε_k et σ_k ses sorties, nous aurons $\boxed{e_k = e_{k-1}\varepsilon_k \text{ et } s_k = \sigma_k + s_{k-1}\varepsilon_k}$.

Question 3 • Le comparateur n bits que nous nous proposons de construire fonctionne en *cascade* ; il est clair que sa taille comme son délai seront linéaires par rapport à n. Ce comparateur comporte un premier niveau formé de n comparateurs 1 bit qui fonctionnent en parallèle et nécessitent chacun 6 portes. Le deuxième étage est formé de $n-1$ circuits d'arbitrage réalisant la fonction $(e_{k-1}, s_{k-1}, \varepsilon_k, \sigma_k) \mapsto (e_k, s_k)$ et nécessitent chacun trois portes. Bilan : $\boxed{C(n) = 9n - 3}$.

• Les n comparateurs 1 bit fonctionnent en parallèle ; après un délai égal à 3, nous disposons des n couples $(\varepsilon_k, \sigma_k)$. La propagation des résultats à travers les $n-1$ circuits d'arbitrage de fait avec un délai $2(n-1)$. Finalement : $\boxed{D(n) = 2n + 1}$.

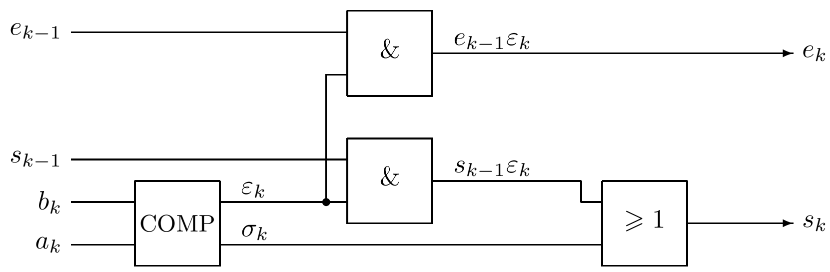

Figure 3: étage k du comparateur n bits

Question 4 • Notons $A_L = \sum_{0 \leqslant k < n} 2^{k-n} a_k$ et $A_H = \sum_{n \leqslant k < 2n} 2^k a_k$. Nous avons $A = 2^n A_H + A_L$ et $0 \leqslant A_L < 2^n$, si bien que A_H et A_L sont respectivement le quotient et le reste dans la division de A par 2^n. Définissons de même B_L et B_H. Alors $A = B$ ssi $A_H = B_H$ et $A_L = B_L$, si bien que $\boxed{e = e_H e_L}$. Ensuite, $A > B$ ssi $A_H > B_H$, ou $A_H = B_H$ et $A_L > B_L$; ainsi $\boxed{s = s_H + e_H s_L}$. Cette couche logique utilise donc trois portes, et introduit un délai égal à deux.

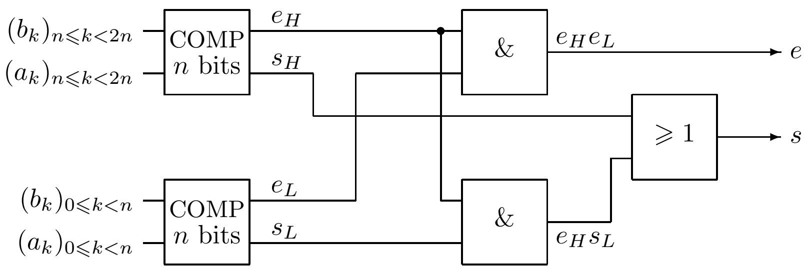

Figure 4: étage du comparateur «diviser pour régner»

Question 5 • $\Gamma(2^{p+1}) = 2\Gamma(2^p)+3$, donc $\Gamma(2^{p+1})+3 = 2(\Gamma(2^p)+3)$ puis $\Gamma(2^p)+3 = 2^p(\Gamma(1)+3) = 9{\cdot}2^p$ puisque $\Gamma(1) = C(1) = 6$; bref : $\boxed{\Gamma(2^p) = 9 \cdot 2^p - 3 = C(2^p)}$. Notons que le comparateur «diviser pour régner» requiert exactement autant de portes que le comparateur banal.

• De même, $\Delta(2^{p+1}) = \Delta(2^p) + 2$ puisque e_H et s_L doivent, chacun, traverser deux poretes logiques. $\Delta(1) = D(1) = 3$, donc $\boxed{\Delta(2^p) = 2p + 3}$: le comparateur «diviser pour régner» a un délai logarithmique par rapport à n.

Question 6 • Remplissage immédiat du tableau ci-dessous.

p	3	4	5	6
2^p	8	16	32	64
$C(2^p)$	69	141	285	573
$D(2^p)$	25	49	97	193
$\Gamma(2^p)$	69	141	285	573
$\Delta(2^p)$	9	11	13	15

Automates finis (Centrale 1998)

Question 1 • L'automate présenté a toutes ses flèches dirigées vers le bas, et ne reconnaît donc qu'un nombre fini de mots : on peut les énumérer «à la main». On trouve $\{aacaa, abcab, bacba, bbcbb\}$ soit exactement l'ensemble L_2.

Question 2 • L'automate $\mathcal{A}_n$ a pour ensemble d'états $Q = \{e_u | u \in E_n\} \cup \{f_u | u \in E_n\}$, où E_n est l'ensemble des mots de longueur n au plus sur l'alphabet $\{a, b\}^*$. L'état initial est e_ε, l'état final f_ε. Les transitions sont de trois types : $\delta(e_v, x) = e_{vx}$ si $x \in \{a, b\}$ et $|vx| \leqslant n$; $\delta(e_v, c) = f_v$ si $|v| = n$; et $\delta(f_{vx}, x) = f_v$ si $x \in \{a, b\}$ et $|vx| \leqslant n$. On vérifie sans peine que $\mathcal{A}_n$ reconnaît exactement L_n.
• Le nombre d'états de $\mathcal{A}_n$ est le double du nombre de mots de longueur n au plus sur l'alphabet $\{a, b\}^*$, soit $2(1 + 2 + 4 + \cdots + 2^n) = 2^{n+2} - 2$.

Question 3 • Il suffit d'ajouter des boucles étiquetées a et b sur les états e_v et f_v pour tout mot v de longueur n sur l'alphabet $\{a, b\}$, ainsi que sur les états e_ε et f_ε.

Question 4 • On n'a ajouté aucun état, donc cet automate a autant d'états que $\mathcal{A}_n$, soit $2^{n+2} - 2$.

Question 5 • Il suffit d'appliquer ce qui a été dit à la question 2 au cas $n = 2$.

Question 6 • Soit v de longueur $2n + 1$, reconnu par $\mathcal{A}$. Comme v appartient à L'_n, il est clair qu'il s'écrit ucu avec $|u| = n$. Supposons que $\mathcal{A}$ passe deux fois par un même état au cours de la reconnaissance de ce mot. Notons x le préfixe qui amène une première fois dans cet état, y le facteur (non vide) qui y amène une deuxième fois, et z le mot tel que $v = xyz$. Alors xz est aussi reconnu par $\mathcal{A}$; mais ceci est contradictoire, car $|xz| < |v|$, alors que L'_n ne contient que des mots de longueur supérieure ou égale à $2n + 1$.

► L'énoncé devait être compris comme suit : «soit $\mathcal{A}$ un automate fini non *nécessairement* déterministe reconnaissant L'_n». Certain(e)s ont cru qu'il s'agissait de l'automate construit à la question 3.

Question 7 • Nous allons montrer que $\mathcal{A}$ possède au moins autant d'états accessibles qu'il y a de mots dans E_n ; comme $|E_n| = 2^{n+1} - 1$, on aura répondu à la question. Soit donc $v \in E_n$; considérons un mot w de longueur $n - |v|$, par exemple $a^{n-|v|}$. Le mot $vwcvw$ appartient à $\mathcal{L}'_n$, donc est reconnu par $\mathcal{A}$: par suite, il est l'étiquette d'un calcul réussi de cet automate. Soit v' un autre élément de E_n : de la même façon, il existe un mot w' de longueur $n - |v'|$ et un calcul réussi de $\mathcal{A}$ d'étiquette $v'w'cv'w'$. Notons q (resp. q') l'état dans lequel se trouve $\mathcal{A}$ après avoir lu v (resp. v') au cours du calcul que nous avons considéré pour $vwcvw$ (resp. $v'w'cv'w'$). Si l'on avait $q = q'$, alors le mot $vw'cv'w'$ serait lui aussi l'étiquette d'un calcul réussi de $\mathcal{A}$; or il est clair que ce mot n'appartient pas à $\mathcal{L}'_n$ puisque vw' et $v'w'$ sont deux mots distincts, de même longueur n, sur l'alphabet $\{a, b\}$.

Question 8 • Reprenons l'idée exploitée à la question précédente. On peut assurer l'existence de $2^{n+1} - 1$ états coaccessibles, associés aux mots de E_n ; plus précisément, si x est un tel mot, on considère un mot y de longueur $n - |v|$, puis un calcul réussi de $\mathcal{A}$ reconnaissant le mot $yxcyx$: nous nous intéressons à l'état dans lequel se trouve $\mathcal{A}$ après avoir lu $yxcy$, au cours de ce calcul. Le raisonnement tenu plus haut montre que ces états sont tous distincts. Montrons qu'ils sont distincts des états accessibles construits à la question précédente. Si ce n'était pas le cas, considérons des mots v, w, x, y sur l'alphabet $\{a, b\}$ tels que $|vw| = |yx| = n$; notons q (resp. q') l'état dans lequel se trouve $\mathcal{A}$ au cours d'un calcul réussi d'étiquette $vwcvw$ (resp. $yxcyx$) après avoir lu le mot v (resp. $yxcy$) : si l'on avait $q = q'$, le mot vx serait reconnu par $\mathcal{A}$; mais ceci est contradictoire, puisque la lettre c n'a aucune occurrence dans ce mot.

Question 9 • Notons $E = (a + b)^*c(a + b)^*$; alors L'_2 est représenté par l'expression rationnelle suivante :

$$(a + b)^*(aaEaa + abEab + baEba + bbEbb)(a + b)^*$$

Question 10 • Les mots n'appartenant pas à L'_1 sont : ceux dans lesquels la lettre c n'apparaît pas ; ceux dans lesquels elle apparaît deux fois ; et enfin, ceux dans lesquels elle apparaît exactement une fois, et qui s'écrivent donc vcw, avec v et w dans $\{a, b\}^*$, aucune lettre de v n'apparaissant dans w ; ceci revient à dire que $v \in \{a\}^*$ et $w \in \{b\}^*$, ou l'inverse. Conclusion : $X^* \setminus L'_1$ est représenté par l'expression rationnelle suivante :

$$(a + b)^* + (a + b)^*c(a + b)^*c(a + b + c)^* + a^*cb^* + b^*ca^*$$

► Remarque : pour ces deux dernières questions, parler de *l'expression rationnelle* associée à un langage est incorrect, puisqu'il n'y a pas unicité de cette expression.

Logique (Centrale 1998)

► Il est vivement conseillé de lire l'énoncé avec soin, et de respecter ses notations ; ainsi, certains ont cru voir un connecteur V là où il n'y avait qu'un connecteur T...

► Les définitions que donne l'énoncé de G et $-$ sont imprécises : à première lecture, on a l'impression que, dans ces définitions, les lettres a, b et c désignent des «variables booléennes». Il est clair qu'elles désignent des formules booléennes.

Question 1 • Il suffit de montrer que les autres connecteurs peuvent être reconstruits à partir de $\neg$ et $\wedge$. Or il est bien connu que $a \vee b = \neg\big((\neg a) \wedge (\neg b)\big)$; et on peut écrire $T = a \wedge (\neg a)$, puis $F = \neg T$.

▶ Remarque : un ensemble de connecteurs logiques est *complet* s'il permet de réaliser par assemblage toute fonction logique. On vient donc de montrer que $\{\neg, \wedge\}$ est un ensemble complet de connecteurs. On connaît plusieurs exemples de connecteur réalisant un ensemble complet : l'opérateur NAND (barre de Sheffer) et l'opérateur NOR. La question 2 construira un autre exemple. L'opérateur XOR ne constitue pas un ensemble complet.

Question 2 • On note que $G(a, a, a) = \neg a$, puis que $G(\neg a, \neg a, \neg b) = a \wedge b$. Ainsi, toute formule ne contenant que les connecteurs $\neg$ et $\wedge$ est équivalente à une formule n'utilisant que le connecteur G ; il résulte alors de la question 1 que $\{G\}$ est un système complet de connecteurs.

Question 3 • On note que $\neg a = (\neg a) \wedge T = T - a$, et que $a \wedge b = (\neg(\neg a)) \wedge b = b - (\neg a)$. Ainsi, toute formule ne contenant que les connecteurs $\neg$ et $\wedge$ est équivalente à une formule n'utilisant que les connecteurs T et $-$. Le même raisonnement que celui tenu à la question 2 montre que $\{T, -\}$ est un système complet de connecteurs.

Question 4 • Aucune forumle construite en n'utilisant que le connecteur $-$ n'est équivalente à la formule T. Pour le prouver, nous raisonnons par l'absurde : considérons une formule f n'utilisant que le connecteur $-$ et équivalente à T. On ne restreint pas la généralité en supposant f de longueur minimale. Cette longueur ne peut être égale à 1 : il faudrait $f = T$, or ce connecteur est interdit ! Donc $f = f_1 - f_2$, où f_1 et f_2 ne s'écrivent qu'avec le connecteur $-$. Aucune de ces formules n'est équivalente à T, puisqu'elles sont strictement plus courtes que f. En particulier, il existe une assignation de valeurs T et F aux variables utilisées dans f_1 et f_2 qui donne à f_1 la valeur F : alors la valeur de $f = (\neg f_2) \wedge f_1$ pour cette assignation des variables est F.

▶ Note personnelle : dans le mot *connecteur*, il y a l'idée d'un lien entre deux choses. Je vois donc mal comment il peut exister des connecteurs unaires ou, pire, 0-aires… Le terme d'*opérateur* aurait été plus judicieux. Compte tenu de l'origine de cet exercice, il ne faut pas trop en attendre.

FIN