ASR1 – TD2 : Calcul booléen

{ Jeremie.Detrey, Patrick.Loiseau, Nicolas.Veyrat-Charvillon } @ens-lyon.fr

http://perso.ens-lyon.fr/jeremie.detrey/06_asr1/

2, 3 et 6 octobre 2006

1 Algèbres booléennes

1. Donnez une définition d'une algèbre de Boole.

Réponse :

Une algèbre de Boole est un ensemble B contenant au moins deux éléments $\top$ et $\bot$, muni de deux lois de composition internes $\vee$ et $\wedge$ ainsi que d'une loi unaire $\neg$:

– $\top, \bot \in B$;
– $\vee, \wedge : B \times B \to B$;
– $\neg : B \to B$.

B doit de plus vérifier les axiomes suivants :

– *Associativité :* $\forall a, b, c \in B$

$$(a \vee b) \vee c = a \vee (b \vee c), \qquad (a \wedge b) \wedge c = a \wedge (b \wedge c).$$

– *Commutativité :* $\forall a, b \in B$

$$a \vee b = b \vee a, \qquad a \wedge b = b \wedge a.$$

– *Absorption :* $\forall a, b \in B$

$$a \wedge (a \vee b) = a, \qquad a \vee (a \wedge b) = a.$$

– *Distributivité :* $\forall a, b, c \in B$

$$(a \vee b) \wedge c = (a \wedge c) \vee (b \wedge c), \qquad (a \wedge b) \vee c = (a \vee c) \wedge (b \vee c).$$

– *Complémentarité :* $\forall a \in B$

$$a \vee \neg a = \top, \qquad a \wedge \neg a = \bot.$$

Pour votre culture, Huntington a démontré en 1933 que seuls suffisaient les trois axiomes suivants :

– *Associativité :* $\forall a, b, c \in B, (a \vee b) \vee c = a \vee (b \vee c)$.
– *Commutativité :* $\forall a, b \in B, a \vee b = b \vee a$.
– *Équation de Huntington :* $\forall a, b \in B, \neg(\neg x \vee y) \vee \neg(\neg x \vee \neg y) = x$.

Dans la foulée, Robbins s'est demandé s'il était possible de remplacer l'équation de Huntington par sa duale, l'équation de Robbins, à savoir : $\forall a, b \in B, \neg(\neg(x \vee \neg y) \vee \neg(x \vee y)) = x$.

Les algèbres de Robbins, définies par ces trois axiomes (associativité, commutativité et équation de Robbins) sont elles équivalentes aux algèbres de Boole ? Cette équivalence n'a pu être démontrée qu'en 1996 par McCune.

2. Montrez que $(\mathcal{P}(A), A, \emptyset, \cup, \cap, C_A)$ est une algèbre de Boole[1].
Réponse :
Ça se fait bien en revérifiant un par un tous les axiomes.
On peut aussi voir ça de manière plus intuitive, en considérant chaque partie de A comme les points vérifiant la condition d'appartenance à cette partie, et en appliquant les connecteurs booléens classique à ces conditions.
Ainsi, en prenant une partie $X \subset A$, on peut écrire :

$$X = \{x \in A | x \in X\}.$$

L'union de deux parties $X, Y \subset A$ devient alors :

$$X \cup Y = \{x \in A | x \in X\} \cup \{x \in A | x \in Y\} = \{x \in A | (x \in X) \vee (x \in Y)\}.$$

On redéfinit ainsi les trois opérations $\cup$, $\cap$ et C_A grâce aux connecteurs $\vee$, $\wedge$ et $\neg$, ce qui nous permet de réutiliser directement les axiomes précédents.

3. Énoncez et démontrez toutes les identités booléennes que vous connaissez.
Réponse :
– *Neutres :* $\forall a \in B, a \vee \bot = a, a \wedge \top = a$.
– *Idempotence :* $\forall a \in B, a \vee a = a, a \wedge a = a$.
– *Saturation :* $\forall a \in B, a \vee \top = \top, a \wedge \bot = \bot$.
– *Compléments :* $\neg\top = \bot, \neg\bot = \top$.
– *Lois de de Morgan :* $\forall a, b \in B, \neg(a \vee b) = \neg a \wedge \neg b, \neg(a \wedge b) = \neg a \vee \neg b$.
– *Involution :* $\forall a \in B, \neg\neg a = a$.

4. On peut noter le "et" par . ou $\wedge$, le "ou" par $+$ ou $\vee$. Discutez ces notations et la dualité de toute chose. Faites l'autocritique de vos conventions de parenthésage.

5. Mettez sous formes disjonctive et conjonctive les expressions suivantes. À quoi sert-ce ?
– $(x.y) + (x.(z + t))$;
– $(x + y).(x + (z.t))$;
– $(x.y.\overline{t}) + \overline{(y.t)}$;
– $(x.y.\overline{t}) + \overline{(y.t)}$.
Réponse :
– $(x.y) + (x.z) + (x.t)$ et $x.(y + z + t)$;
– $x + (y.z.t)$ et $(x + y).(x + z).(x + t)$;
– $(x.y) + (y.t)$ et $y.(x + t)$;
– $\overline{y} + \overline{t}$ et $(\overline{y} + \overline{t})$.

6. En technologie CMOS (*complementary metal-oxide-semiconductor*) on a uniquement les briques de bases suivantes :
– l'inverseur : $\text{NOT}(x) = \overline{x}$;
– le non-et : $\text{NAND}(x_1, \ldots, x_n) = \overline{x_1.\cdots.x_n}$;
– le non-ou : $\text{NOR}(x_1, \ldots, x_n) = \overline{x_1 + \cdots + x_n}$.
Écrivez les expressions précédentes pour les implanter en CMOS.
Réponse :
– $NOR(NOT(x), NOR(y, z, t))$;

[1] Mathworld définit une algèbre de Boole comme "*the partial order on subsets defined by inclusion*".

- $NAND(NOT(x), NAND(y, z, t))$;
- $NOR(NOT(y), NOR(x, t))$;
- $NAND(y, t)$.

7. Rappelez ce qu'est un littéral, un monôme (parfois appelé aussi un minterme), un monal (ou maxterme), un polynôme.
Réponse :
- *Littéral : une variable ou sa négation. Par exemple x, y ou $\overline{t}$.*
- *Minterme : une conjonction de littéraux. Par exemple $x.y.\overline{z}.t$. On appelle cela un minterme car la faire la conjonction de plusieurs littéraux revient à prendre la valeur minimale de ces littéraux.*
- *Maxterme : le dual du minterme, c'est-à-dire une disjonction de littéraux. Par exemple $x + y + \overline{z} + t$.*
- *Polynôme : une disjonction de mintermes. Par exemple, $(x.y + x.\overline{z}.t) + (\overline{y}.t)$.*

8. Rappelez ce qu'est un monôme canonique (parfois appelé minterme complet).
Réponse : Il s'agit d'un minterme qui fait intervenir toutes les variables de l'expression, complémentées ou non. Ainsi, si les variables sont x, y, z et t, alors $x.\overline{y}.\overline{z}.t$ est un monôme canonique, mais pas $x.z.\overline{t}$.

9. Quelle est la différence entre une fonction booléenne et une expression booléenne ?
Réponse : On parle de fonction quand on désigne un calcul, c'est-à-dire une transformation des entrées pour obtenir un résultat. L'expression est comment ce calcul est effectué. Une fonction peut correspondre ainsi à plusieurs expressions, différentes même si elles calculent toutes la même chose.

Ainsi, la fonction "majorité", qui peut être donnée par sa table de vérité, par exemple, a plusieurs expressions :

$$\begin{aligned}\mathrm{maj}(x, y, z) &= (x.y) + (y.z) + (z.x), \text{ ou bien} \\ &= (x + y).(y + z).(z + x).\end{aligned}$$

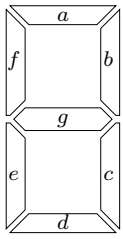
10. Rappelez les théorèmes de Shannon.
Réponse :
Pour $f : B^n \to B$:
$f(x_1, \ldots, x_i, \ldots, x_n) = \overline{x_i}.f(x_1, \ldots, 0, \ldots, x_n) + x_i.f(x_1, \ldots, 1, \ldots, x_n)$,
$f(x_1, \ldots, x_i, \ldots, x_n) = (x_i + f(x_1, \ldots, 0, \ldots, x_n)).(\overline{x_i} + f(x_1, \ldots, 1, \ldots, x_n))$.

11. Comment construit-on les première et deuxième formes normales ? Quel est leur intérêt ? Quelle est leur taille ? Quelle simplification triviale peut-on apporter à chacune de ces formes normales ? Pour une fonction booléenne donnée, laquelle choisir a priori ?
Réponse : Vu en cours.
À noter pour la forme normale à choisir, lorsque la table de vérité contient n_0 zéros et n_1 uns :
- *sous forme normale disjonctive, l'expression va contenir n_1 mintermes, et*
- *sous forme normale conjonctive, l'expression va contenir n_0 maxtermes.*

Il faut donc choisir la forme qui minimise le nombre de termes : disjonctive si $n_1 \leq n_0$ et conjonctive si $n_1 \geq n_0$.

12. Dessinez un OBDD. Comment construit-on une expression booléenne à partir de ce graphe ?
Réponse : Vu en cours.

2 Mise en œuvre : l'afficheur 7 segments



1. Combien de segments a l'afficheur 7 segments ?
   Réponse : 42 *(à un facteur* 6 *près)*[2].
2. Combien de bits faut-il pour coder un chiffre de 0 à 9 en binaire ? On complètera avec un afficheur éteint.
   Réponse : 4 bits.
3. Dessinez les 10 chiffres dans cet afficheur et définissez la fonction booléenne à implémenter pour décoder et afficher un chiffre.
   Réponse :

x_3 x_2 x_1 x_0	*Affichage*	a b c d e f g
0 0 0 0	0	1 1 1 1 1 1 0
0 0 0 1	1	0 1 1 0 0 0 0
0 0 1 0	2	1 1 0 1 1 0 1
0 0 1 1	3	1 1 1 1 0 0 1
0 1 0 0	4	0 1 1 0 0 1 1
0 1 0 1	5	1 0 1 1 0 1 1
0 1 1 0	6	1 0 1 1 1 1 1
0 1 1 1	7	1 1 1 0 0 1 0
1 0 0 0	8	1 1 1 1 1 1 1
1 0 0 1	9	1 1 1 1 0 1 1
1 0 1 0		0 0 0 0 0 0 0
1 0 1 1		0 0 0 0 0 0 0
1 1 0 0		0 0 0 0 0 0 0
1 1 0 1		0 0 0 0 0 0 0
1 1 1 0		0 0 0 0 0 0 0
1 1 1 1		0 0 0 0 0 0 0

4. Traduisez-la sous forme d'expression booléenne et d'arbre de décision.
   Réponse : On présente ici uniquement la solution pour le segment a.
   La table de vérité de a *comporte autant de zéros que de uns, on peut donc utiliser les deux solutions :*
   – *sous forme disjonctive :*

$$a(x_3,x_2,x_1,x_0) = (\overline{x_3}.\overline{x_2}.\overline{x_1}.\overline{x_0}) + (\overline{x_3}.\overline{x_2}.x_1.\overline{x_0}) + (\overline{x_3}.\overline{x_2}.x_1.x_0) + (\overline{x_3}.x_2.\overline{x_1}.x_0) + (\overline{x_3}.x_2.x_1.\overline{x_0}) + (\overline{x_3}.x_2.x_1.x_0) + (x_3.\overline{x_2}.\overline{x_1}.\overline{x_0}) + (x_3.\overline{x_2}.\overline{x_1}.x_0).$$

[2]Pour celles et ceux qui se demandent pourquoi 42, aller emprunter *The Hitchhiker's Guide to the Galaxy* de Douglas Adams à votre bibliothèque favorite.

– *sous forme conjonctive :*

$$\begin{aligned} a(x_3,x_2,x_1,x_0) = {} & (x_3+x_2+x_1+\overline{x_0}).(x_3+\overline{x_2}+x_1+x_0).(\overline{x_3}+x_2+\overline{x_1}+x_0). \\ & (\overline{x_3}+x_2+\overline{x_1}+\overline{x_0}).(\overline{x_3}+\overline{x_2}+x_1+x_0).(\overline{x_3}+\overline{x_2}+x_1+\overline{x_0}). \\ & (\overline{x_3}+\overline{x_2}+\overline{x_1}+x_0).(\overline{x_3}+\overline{x_2}+\overline{x_1}+\overline{x_0}). \end{aligned}$$

5. Minimisez l'expression booléenne ou l'OBDD, au choix.
Réponse :
Pour minimiser, on utilise l'algorithme de Quine-McCluskey dans le cas d'une forme disjonctive. Cet algorithme fonctionne en gros en trois étapes.

1. *Identifier et trier les mintermes correspondant à des* 1 *dans la table de vérité.*
   On note $[0]$ *le minterme correspondant à* $(x_3\ x_2\ x_1\ x_0) = (0\ 0\ 0\ 0)$*, puis* $[1]$ *le minterme correspondant à* $(0\ 0\ 0\ 1)$*, et ainsi de suite. Pour la fonction* a*, ça correspond donc aux huit mintermes* $[0]$*,* $[2]$*,* $[3]$*,* $[5]$*,* $[6]$*,* $[7]$*,* $[8]$ *et* $[9]$*.*
   On trie ensuite nos mintermes par le nombre de bits à 1 *(c'est-à-dire le nombre de littéraux non complémentés dans son expression). Ça nous donne :*

Nb. de bits à 1	*Minterme*	*Valeur*
0	$[0]$	$(0\ 0\ 0\ 0)$
1	$[2]$	$(0\ 0\ 1\ 0)$
	$[8]$	$(1\ 0\ 0\ 0)$
2	$[3]$	$(0\ 0\ 1\ 1)$
	$[5]$	$(0\ 1\ 0\ 1)$
	$[6]$	$(0\ 1\ 1\ 0)$
	$[9]$	$(1\ 0\ 0\ 1)$
3	$[7]$	$(0\ 1\ 1\ 1)$

2. *Réduire les différents mintermes en cherchant des similitudes entre eux.*
   On recherche toutes les paires de mintermes qui ne diffèrent que d'un bit. Cela signifie donc que ces deux mintermes appartiennent forcément à deux catégories successives, puisque leur nombre de bits à 1 *ne peut différer que d'un.*
   Ainsi, on peut voir que les mintermes $[0]$ $(0\ 0\ 0\ 0)$ *et* $[2]$ $(0\ 0\ 1\ 0)$ *ne diffèrent que de la valeur du bit* x_1*. On peut donc combiner les mintermes* 0 *et* 2 *en un minterme réduit noté* $[0,2]$ *et correspondant à la valeur* $(0\ 0\ -\ 0)$*, le symbole* $-$ *indiquant que le bit* x_1 *peut prendre les deux valeurs* 0 *ou* 1 *(on appelle cela un "don't care").*
   En cherchant toutes les paires que l'on peut réduire, on obtient au final un second tableau similaire au premier, dans lequel les mintermes réduits sont toujours triés par nombre de bits à 1*.*
   Tant qu'il reste des mintermes réductibles, on réitère l'opération. Lorsqu'un minterme n'est réductible avec aucun autre, on le marque par une étoile. Cela donne d'abord la table :

Nb. de bits à 1	*Minterme*	*Valeur*	
0	$[0,2]$	$(0\ 0\ -\ 0)$	$*$
	$[0,8]$	$(-\ 0\ 0\ 0)$	$*$
1	$[2,3]$	$(0\ 0\ 1\ -)$	
	$[2,6]$	$(0\ -\ 1\ 0)$	
	$[8,9]$	$(1\ 0\ 0\ -)$	$*$
2	$[3,7]$	$(0\ -\ 1\ 1)$	
	$[5,7]$	$(0\ 1\ -\ 1)$	$*$
	$[6,7]$	$(0\ 1\ 1\ -)$	

De cette table on construit la suivante :

Nb. de bits à 1	*Minterme*	*Valeur*	
1	$[2,3,6,7]$	$(0-1-)$	$*$

La liste des mintermes suffisants pour calculer notre fonction est donc l'ensemble des mintermes irréductibles, soit $[0,2]$*,* $[0,8]$*,* $[8,9]$*,* $[5,7]$ *et* $[2,3,6,7]$*.*

3. *Chercher une couverture minimale des mintermes initiaux avec nos mintermes réduits.*

   Il y peut en effet y avoir de la redondance dans nos mintermes réduits. Il faut donc éliminer les mintermes inutiles. On dresse pour cela une table indiquant les mintermes initiaux couverts par les mintermes réduits :

	$[0]$	$[2]$	$[3]$	$[5]$	$[6]$	$[7]$	$[8]$	$[9]$
$[0,2]$	$*$	$*$						
$[0,8]$	$*$						$*$	
$[8,9]$							$*$	$*$
$[5,7]$				$*$		$*$		
$[2,3,6,7]$		$*$	$*$		$*$	$*$		

   Le minterme $[2,3,6,7]$ *est nécessaire car c'est le seul couvrant* $[3]$*. De même,* $[5,7]$ *est le seul couvrant* $[5]$ *et* $[8,9]$ *le seul couvrant* $[9]$*. On a donc le choix entre* $[0,2]$ *et* $[0,8]$ *pour couvrir* $[0]$*.*

   Si l'on choisit $[0,2]$*, on obient l'expression finale :*

$$a(x_3,x_2,x_1,x_0) = (\overline{x_3}.\overline{x_2}.\overline{x_0}) + (x_3.\overline{x_2}.\overline{x_1}) + (\overline{x_3}.x_2.x_0) + (\overline{x_3}.x_1).$$

6. Comment optimiser encore ?

Réponse :

Essentiellement deux idées ici.

La première est de réutiliser des sous-expressions communes entre les fonctions des sept segments. On peut ainsi tout réduire vers une forme normale disjonctive pour ensuite identifer les mintermes partagés entre les diverses expressions. Ainsi, une seule porte logique est nécessaire pour chaque minterme.

La seconde idée est d'assouplir les spécifications de notre afficheur : demander d'éteindre tous les segments de l'afficheur lorsque les quatre bits d'entrée codent un nombre supérieur à 10 *est en fait une méchante contrainte. On peut donc choisir des valeurs plus sympathiques pour les fonctions sur ces entrées, permettant ainsi une meilleure minimisation des expressions.*

7. *Bonus* : Faites de même pour un afficheur 7 segments affichant un chiffre hexadécimal (0-9, A-F).