Informatique Théorique

Informatique Théorique 3

(MAM3-SI3)

October 4, 2021

1 Deux propositions

On considére une logique du premier ordre dont les seuls prédicats sont les deux propositions P et Q (pas de variables, pas de fonction). Le but de l'exercice est de montrer que toute formule peut s'écrire en forme normale disjonctive en utilisant au plus 3 connecteurs logiques binaires (forcement des $\wedge$ et des $\vee$)

1. Toute formule pouvant être ici assimilée à sa table de vérité, combien de formules différentes devez vous considérer ?
2. La manière la plus simple d'obtenir une forme normale disjonctive pour la formule est de dire que chaque ligne de la table où la formule est vrai (minterm) correspond à un $\wedge$ et de faire un $\vee$ de toutes les lignes où la formule est vraie, par exemple, la formule dont la table de vérité est :

P	Q	Φ
1	1	0
1	0	1
0	1	0
0	0	1

   correspond à $(P \wedge \neg Q) \vee (\neg P \wedge \neg Q)$. C'est ce qu'on appelle la forme canonique disjonctive.

   Pouvez vous faire mieux pour cet exemple ?
3. Si vous utilisez ces formes canoniques, quelles sont les tables de vérité pour lesquelles vous obtenez une solution avec plus de trois connecteurs logiques ?
4. Pour chacune de ces tables , donner une solution avec au plus trois connecteurs logiques binaires.
5. De combien de connecteurs logiques binaires avez vous besoin pour exprimer le xor ?
6. Pouvez vous en déduire que dans ce cadre, on peut aussi écrire toute formule en forme normale conjonctive avec au plus trois connecteurs logiques binaires?

2 Simplification "à la main"

Simplifier les formules suivantes

1. $(A \wedge B \wedge C) \vee (A \wedge B \wedge \neg C) \vee (\neg A \wedge B \wedge \neg C) \vee (\neg A \wedge B \wedge C)$
2. $(A \wedge \neg B \wedge \neg C) \vee (A \wedge B \wedge \neg C) \vee (A \wedge B \wedge C) \vee (A \wedge B \wedge \neg C)$
3. $(\neg A \wedge \neg C) \vee (A \wedge B \wedge \neg C) \vee (A \wedge B \wedge D) \vee (A \wedge B \wedge C \wedge \neg D)$

3 Algorithme de Quine Mc Cluskey

1. Lister tous les minterms de f dans une table en les convertissant en mots de $\{0,1\}^*$
2. Les grouper selon leur poids, c'est à dire par le nombre de 1 dans chaque minterm
3. Unir les termes deux à deux, c'est à dire :
   - Comparer les termes d'un groupe avec ceux du groupe adjacent pour essayer de les combiner
   - Créer une nouvelle table avec les combinaisons trouvées : 0100 +0101 = 010-

- Rayer chaque terme utilisé pour la combinaison et passer à la table suivante

4. Répéter l'étape (3) autant de fois que c'est possible
5. Identifier les impliquants premiers (qui correspondent aux termes non rayés)
6. Identifier les impliquants premiers essentiels
7. Vérifier si la fonction est entièrement exprimée par ses impliquants essentiels, auquel cas arrêter
8. Si on n'a pas fini à l'étape (7), choisir les impliquants premiers appropriés

Utilisez cet algorithme pour simplifier les fonctions logiques suivantes exprimées sous la forme d'une somme (représentant des $\vee$) de produits logique (représentant des $\wedge$) des variables a,b,c,d,e éventuellement niées, et toujours dans le même ordre . Chaque produit logique est représenté par la valeur en base 10 de l'ecriture binaire correspondante, par exemple a.b.c.d est identifié à 1111 puis à l'entier quinze, tandis que $\neg a \neg b \neg cd$ est identifié à 0001 puis à l'entier 1

1. $F_1 = \Sigma(0, 2, 5, 6, 7, 8, 10, 12, 13, 14, 15)$ plusieurs solutions sont possibles
2. $F_2 = \Sigma(0, 1, 2, 3, 7, 14, 15, 22, 23, 29, 31)$ une seule solution possible
3. $F_3 = \Sigma(0, 1, 2, 5, 6, 7, 9, 10, 11, 13, 14, 15)$ plusieurs solutions sont possibles

4 Avec ou sans algorithme de Quine Mc Cluskey

1. Écrire une expression logique minimale qui calcule la majorité de 4 bits (en cas d'égalité, c'est 0 qui l'emporte).
2. Écrire une expression logique minimale qui calcule le plus grand de 4 bits (un peu long avec l'algo ...).