TP 2 – Fonctions booléennes

Téléchargez le fichier `tp2.c`. Il contient les entêtes des fonctions que vous complèterez.

1 Préliminaires

Pour $1 \leq p \leq 32$, on souhaite générer l'ensemble de tous les vecteurs possibles de p booléens, c'est à dire :
— (0,0,...,0),
— (0,0,...,1),
— ...
— (0,1,...,1),
— (1,1,...,1).
En remarquant que chacun de ces vecteurs correspond à un entier n compris entre 0 et $2^p - 1$, dîtes comment générer un tel ensemble.
1 – La fonction `long long int P2(int n)` renvoie la n-ième puissance de 2, et la directive `BIT(A,B)` définie par

```
#define BIT(A,B) ((A>>B)&0x1)
```

renvoie la valeur du `B`-ième bit (en partant du bit de poids le plus faible) de l'entier `A`. Utilisez ces deux fonctions pour afficher l'ensemble des vecteurs de p booléens pour p=2,3 et 4.

2 Fonctions booléennes

Pour $1 \leq p \leq 32$, on propose de représenter un vecteur v de p booléens $v = (X_{p-1}, ..., X_1, X_0)$ par l'entier dont les chiffres en écriture binaire sont les composantes de v. Par exemple, ici v sera représenté par l'entier $X_{p-1} * 2^{p-1} + X_{p-2} * 2^{p-2} + ... + X_1 * 2 + X_0$. Ainsi, une fonction de p variables booléennes pourra être représentée par une fonction ayant pour argument un entier, renvoyant un entier.
2 – En utilisant la directive `BIT(A,B)`, implémentez les fonctions :
— `int F2(int)` de deux variables booléenne A et B valant $A \oplus B$.
— `int F3(int)` de trois variables booléennes A, B, C valant $A \cdot \overline{B} + B \cdot \overline{C} + \overline{B} \cdot C$.
— `int F4(int)` de quatres variables booléennes A, B, C, D valant $(A \cdot B \cdot \overline{C} \oplus A \cdot \overline{B} \cdot D) + C \cdot \overline{D}$.

3 Tables de vérité

En language C, on a la possibilité d'écrire des fonctions ayant comme argument une (ou plusieurs) fonctions, en passant un pointeur sur cette fonction. Par exemple, la fonction :

```
void Racine( int (*func) (int) ) {
  int i;
  for(i=-10,i<=10;i++) if (func(i)==0) printf("Solution: %d\n", i);
}
```

cherche et affiche les zéros dans l'intervalle $[-10, 10]$ de la fonction `func` passée en argument.
3 – Écrivez la fonction `void verite2var( int (*func) (int) )` affichant à l'écran la table de vérité de la fonction de deux variables `func`, sous la forme :

A	B	$F(A,B)$

Testez votre fonction avec `F2`.
4 – Écrivez la fonction `void veriteNvar(int N, int (*func) (int) )` affichant à l'écran la table de vérité de la fonction de `N` variables `func`, sous la forme :

X_0	X_1	...	X_{N-1}	$F(X_0, ..., X_{N-1})$

Testez votre fonction avec `F3` et `F4`.

4 Formes normales

Comment obtenir la forme normale disjonctive (somme de produits) d'une expression booléenne à partir de sa table de vérité ? Même question pour la forme normale conjonctive.

5 – Écrivez les fonctions `void conjonctive(int N, int (*func) (int))` et `void disjonctive (int N, int (*func) (int))` affichant les formes normales conjonctives et disjonctives de fonctions booléennes de `N` variables. (On utilisera la notation `!A` pour exprimer la négation de la variable `A`). Comment vérifier le résultat obtenu ?