Simplification des FNC

- La FNC de A est

$$(a \vee b \vee \neg c) \wedge (a \vee \neg b \vee \neg c) \wedge (\neg a \vee \neg b \vee \neg c)$$

- Il y a 3 termes :
  - pour les termes 1 et 2 :

$$(a \vee b \vee \neg c) \wedge (a \vee \neg b \vee \neg c) \ \underline{eq} \ (a \vee \neg c)$$

  - la formule A est donc équivalente à

$$(a \vee \neg c) \wedge (\neg a \vee \neg b \vee \neg c)$$

  - Finalement $\quad A \ \underline{eq} \ (a \vee \neg c) \wedge (\neg b \vee \neg c)$
- En factorisant la FNC, on retrouve la FND $(\neg c \vee (a \wedge \neg b))$.
- La simplification des FND et FND n'est pas toujours facile, on utilise des méthodes spécifiques comme les *tables de Karnaugh*.

Algèbre de Boole

- Le nom d'**algèbre de Boole** ou **calcul booléen** est en l'honneur du mathématicien britannique George Boole (1815-1864), considéré comme créateur de la logique moderne.
- Le calcul booléen appliqué au calcul des propositions permet une approche algébrique pour traiter les formules logiques.
- On introduit les opérateurs arithmétiques binaires "+" pour $\vee$ et "." pour $\wedge$. L'opérateur $\bar{a}$ représente l'opération unaire $\neg a$ du calcul des propositions.

  Exemple : $(a \vee \neg c) \wedge (\neg a \vee \neg b \vee \neg c)$ s'écrit $(a + \bar{c}).(\bar{a} + \bar{b} + \bar{c})$
- La structure d'algèbre de Boole s'applique dans d'autres cadres, pas étudiés dans ce cours.
  Un exemple est l'*algèbre de Boole des parties d'un ensemble E*, $\mathcal{P}(E)$. L'opération"+" est alors la réunion $\cup$, l'opération "." est l'intersection $\cap$ et $\bar{A}$ désigne le complément de A dans E.

Propriétés de base

L'ensemble $\{0,1\}$ est muni des opérations "+", "." et "‾" vérifiants :

- associativité : $(a+b)+c = a+(b+c)$ et $(a.b).c = a.(b.c)$
- commutativité : $a+b = b+a$ et $a.b = b.a$
- éléments neutre : $0+a = a$ et $1.a = a$
- idempotence : $a+a = a$ et $a.a = a$
- involution : $\bar{\bar{a}} = a$
- complémentarité : $a.\bar{a} = 0$ et $a+\bar{a} = 1$
- éléments absorbants : $a+1 = 1$ et $a.0 = 0$
- distributivité de . par rapport à + : $a.(b+c) = a.b + a.c$
- distributivité de + par rapport à . : $a+(b.c) = (a+b).(a+c)$

Attention à utiliser ces notations et règles dans le bon contexte !

En particulier ne pas confondre avec les lois sur le corps F_2, vues en calcul modulaire.

Propriétés de base

- Les lois de De Morgan s'écrivent :

$$\overline{a+b} = \bar{a}.\bar{b} \qquad \text{et} \qquad \overline{a.b} = \bar{a}+\bar{b}\,.$$

- Les simplifications suivantes peuvent être utiles

$$a+\bar{a}.b = a+b \qquad \text{et} \qquad (a+b).(a+c) = a+b.c$$

- Le calcul booléen est utilisée en électronique pour simplifier des circuits logiques ou en programmation pour simplifier des tests logiques.
- Suivant le langage de programmation, le contexte,. . . , les opérations sont notées de différentes façons :

  le "." est aussi noté "$\wedge$", "&", "&&" ou "AND" ;
  le "+" est aussi noté "$\vee$", "|", "||" ou "OR" ;
  le "‾" est aussi noté "$\neg$", " !", "NOT" ;

Fonction booléenne

- Les formules du calcul des propositions deviennent des fonctions booléennes c'est-à-dire des applications de $\{0,1\}^n \longrightarrow \{0,1\}$ où n est le nombre de variables.
  On parle aussi de fonctions logiques.
- Pour une fonction booléenne de n variables $f(x_1,\ldots,x_n)$
  - on appelle minterme un *produit* m qui contient chaque variable x_i $(1 \leq i \leq n)$, ou sa négation, une seule fois et tel que $m = 1$ entraîne que $f(x_1,\ldots,x_n)$ est *vraie*.
  - on appelle maxterme une *somme* M qui contient chaque variable x_i $(1 \leq i \leq n)$, ou sa négation, une seule fois et telle que $M = 0$ entraîne que $f(x_1,\ldots,x_n)$ est *fausse*.
- Pour une fonction logique f on peut dire que
  - la **FND** de f est la disjonction des mintermes de f ;
  - la **FNC** de f est la conjonction des maxtermes de f.

Fonction booléenne. Exemple

On considère un fonction $f : \{0,1\}^3 \longrightarrow \{0,1\}$ définie par

a	b	c	$f(a,b,c)$
0	0	0	1
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0

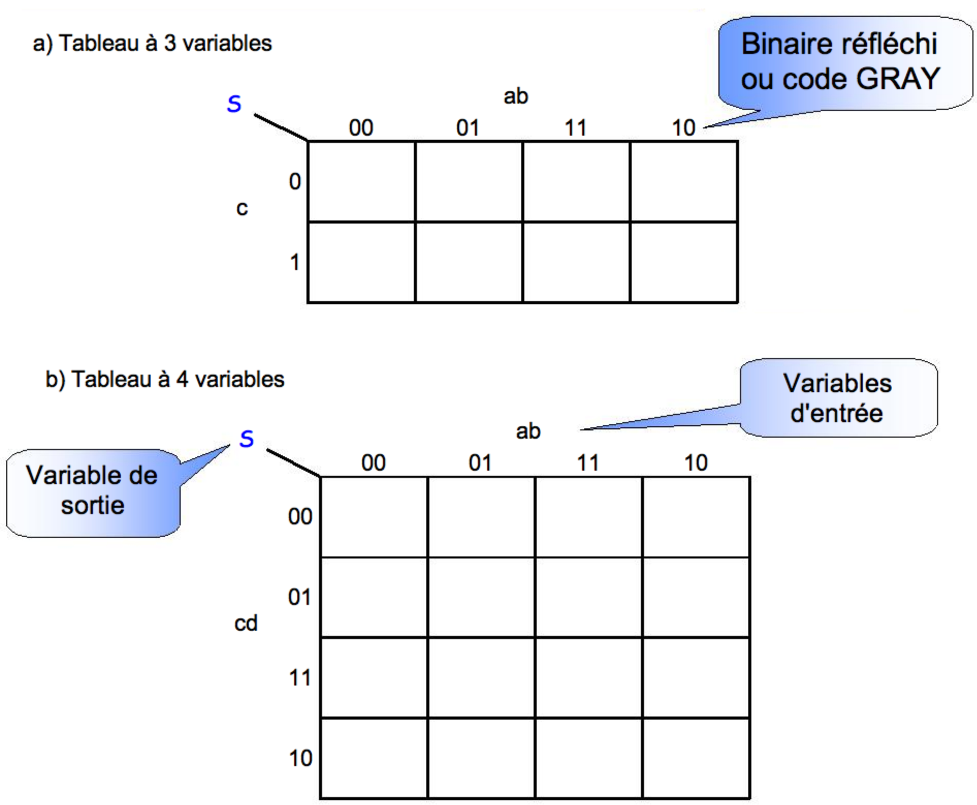
On obtient

$$\begin{aligned}
\text{FND de } f &= (\bar{a}.\bar{b}.\bar{c}) + (\bar{a}.b.\bar{c}) + (a.\bar{b}.\bar{c}) + (a.\bar{b}.c) + (a.b.\bar{c}) \\
&= \textit{«somme des mintermes»} \\
\text{FNC de } f &= (a + b + \bar{c}).(a + \bar{b} + \bar{c}).(\bar{a} + \bar{b} + \bar{c}) \\
&= \textit{«produit des maxtermes»}
\end{aligned}$$

Table de Karnaugh

- La simplification d'une expression logique par le **tableau de Karnaugh** est une méthode développé en 1953 par Maurice Karnaugh, ingénieur en télécommunications au laboratoires Bell.
- La méthode de Karnaugh consiste à présenter les états d'une fonction logique, non pas sous la forme d'une table de vérité, mais en utilisant un **tableau à double entrée**.
- Chaque case du tableau correspond à une combinaison des variables d'entrées, donc à une ligne de la table de vérité.
- Le tableau de Karnaugh aura autant de cases que la table de vérité possède de lignes.
- Les lignes et les colonnes du tableau sont numérotées selon le code binaire réfléchi (code de Gray) :
  à chaque passage d'une case à l'autre, une seule variable change d'état.

Tableaux de Karnaugh. Exemple



On remplit le tableau grâce à la fonction booléenne $S = f(a, b, c, d)$.

Table de Karnaugh. Somme

Pour obtenir une somme on procède comme suit

- **Regrouper** les cases adjacentes de “1” par paquets de taille des puissance de 2.

  Pour minimiser le nombre de paquets, prendre les rectangles le plus grand possible : $2^n, \ldots, 16, 8, 4, 2{,}1$.
- Une même case peut faire partie de plusieurs regroupements.
- Les regroupements peuvent se faire au delà des bords : les côtés/coins ont des codes Gray voisins.
- Toute case contenant “1” doit faire partie d’au moins un regroupement, mais aucun “0” ne doit y être.
- Pour chaque rectangle, on **élimine** les variables qui changent d’état, l’on ne conserve que celles qui restent fixes.
  On **multiplie** les variables fixes par “.”
  afin d’obtenir des **mintermes** de S.
- Les produits obtenus sont ensuite **sommés** avec “+”
  et l’on obtient une **FND** de S.

Table de Karnaugh

Exemple :
Simplifier la fonction $S = \bar{a}.b.\bar{c}.\bar{d} + a.b.c.d + a.\bar{b}.c.d + a.b.\bar{c}.\bar{d}$

La table de Karnaugh associée est

S — cd \ ab	00	01	11	10
00	0	1	1	0
01	0	0	0	0
11	0	0	1	1
10	0	0	0	0

- 1^{er} regroupement : a change d’état et est éliminé, b vaut 1, c et d valent 0, d’où le minterme $b\bar{c}\bar{d}$
- 2^{eme} regroupement : b change d’état et est éliminé, a, c et d valent 1, d’où le minterme acd
- On fait la somme des mintermes et $S = (acd) + (b\bar{c}\bar{d})$

Table de Karnaugh

Exemple : Soit

$$W = \bar{a}\bar{b}\bar{c}\bar{d} + \bar{a}\bar{b}\bar{c}d + \bar{a}\bar{b}cd + \bar{a}\bar{b}c\bar{d}$$

On dresse la table de Karnaugh :

W	ab: 00	01	11	10
cd: 00	1	0	0	0
01	1	0	0	0
11	1	0	0	0
10	1	0	0	0

$\boxed{W = \bar{a}\,\bar{b}}$

Table de Karnaugh

Exemple : Soit

$X = \bar{a}\bar{b}\bar{c}\bar{d} + \bar{a}\bar{b}\bar{c}d + \bar{a}\bar{b}cd + \bar{a}\bar{b}c\bar{d} + a\bar{b}\bar{c}\bar{d} + a\bar{b}\bar{c}d + a\bar{b}cd + a\bar{b}c\bar{d}$

On dresse la table de Karnaugh :

X	ab: 00	01	11	10
cd: 00	1	0	0	1
01	1	0	0	1
11	1	0	0	1
10	1	0	0	1

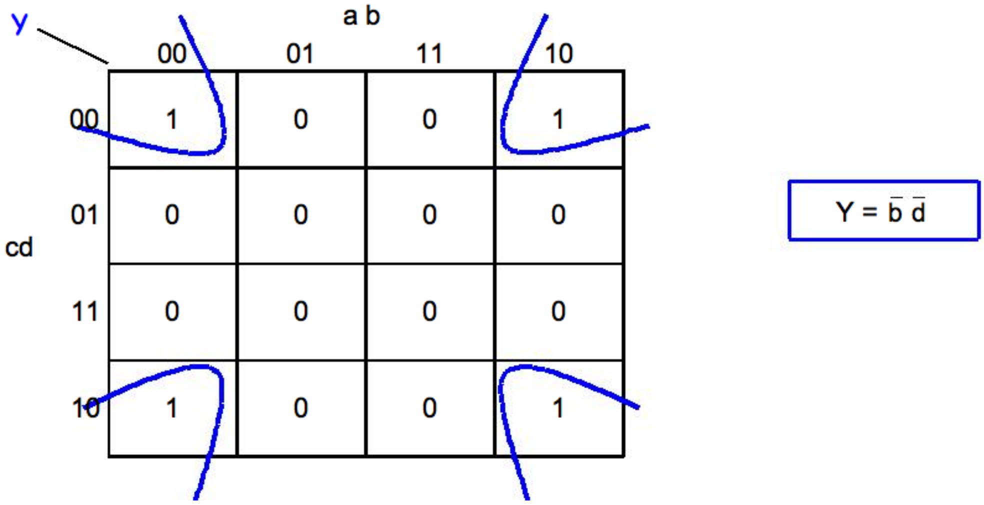
$\boxed{X = \bar{b}}$

Table de Karnaugh

Exemple : Soit

$$Y = \bar{a}\bar{b}\bar{c}\bar{d} + \bar{a}\bar{b}c\bar{d} + a\bar{b}\bar{c}\bar{d} + a\bar{b}c\bar{d}$$

On dresse la table de Karnaugh :



Récréation

Résolution de problèmes par le calcul des propositions
en utilisant la simplification par table de Karnaugh.

Lors d'une enquête de l'inspecteur Maigret, les personnes A, B, C et D sont suspectées. Il est établi que :

1. Si A et B sont coupables, il en est de même de C.
2. Si A est coupable, l'un au moins de B et C est aussi coupable.
3. Si C est coupable, D l'est aussi.
4. Si A est innocent, D est coupable.

Peut-on établir la culpabilité de l'un ou plusieurs des suspects ?

On définit les propositions :

a = « A est coupable », b = « B est coupable » ,
c = « C est coupable » et d = « D est coupable ».

Application (suite)

On traduit les quatre affirmations en langage des propositions
et l'on écrit leur conjonction :

$$M = [(a \wedge b) \to c] \wedge [a \to (b \vee c)] \wedge [c \to d] \wedge [\neg a \to d]$$

On dresse la table de Karnaugh de M :

cd \ ab	00	01	11	10
00	0	0	0	0
01	1	1	0	0
11	1	1	1	1
10	0	0	0	0

D'où $M = \bar{a}.d + c.d$,

en langage des propositions on a $M = (\neg a \wedge d) \vee (c \wedge d)$.
On peut donc conclure que D est certainement coupable !

Table de Karnaugh. Produit

Pour obtenir un produit on procède comme suit

- **Regrouper** les cases adjacentes de "0" par paquets de taille des puissance de 2.

  Pour minimiser le nombre de paquets, prendre les rectangles le plus grand possible : $2^n, \ldots, 16, 8, 4, 2{,}1$.
- Une même case peut faire partie de plusieurs regroupements.
- Les regroupements peuvent se faire au delà les bords :
  les côtés/coins ont des codes Gray voisins.
- Toute case contenant "0" doit faire partie d'au moins un regroupement, mais aucun "1" ne doit y être.
- Pour chaque rectangle, on **élimine** les variables qui changent d'état, l'on ne conserve que celles qui restent fixes.
  On **somme** les variables fixes avec "+"
  afin d'obtenir des **maxtermes** de S.
- Les sommes obtenus sont ensuite **multipliées** avec "."
  et l'on obtient une **FNC** de S.

Table de Karnaugh

Exemple : On reprend la table de Karnaugh de

$X = \bar{a}\bar{b}\bar{c}\bar{d} + \bar{a}\bar{b}\bar{c}d + \bar{a}\bar{b}cd + \bar{a}\bar{b}c\bar{d} + a\bar{b}\bar{c}\bar{d} + a\bar{b}\bar{c}d + a\bar{b}cd + a\bar{b}c\bar{d}$

X: cd \ ab	00	01	11	10
00	1	0	0	1
01	1	0	0	1
11	1	0	0	1
10	1	0	0	1

$X = \bar{b}$

1. Soit on regroupe les “0” et on fait le **produit des maxtermes** de variables qui ne changent pas et qui rendent faux X.
2. Soit on regroupe les “1”et on fait la somme des mintermes de variables qui ne changent pas et qui rendent vrai X.

Circuits logiques et booléens

- Les *circuits logiques*, composants de base des ordinateurs, sont conçus à partir de *circuits élémentaires* correspondants aux opérations booléennes “.”, “+” et “‾”.
- Un circuit logique peut être vu comme une boîte noire ayant
  $n \geq 1$ ports d'entrée $e_1, e_2, \ldots, e_n$
  $m \geq 1$ ports de sortie $s_1, s_2, \ldots, s_m$
- Il traite des informations codées sur n bits et donne des informations codées sur m bits.
- Le codage de l'information, en entrée ou sortie, est représenté par l'absence (0) ou la présence (1) d'une tension électrique.
- On appelle ces circuits *logiques* car un bit d'information 0 est assimilé à la valeur de vérité *faux* et 1 à *vrai*
- On représente ainsi une application de $\{0,1\}^n$ dans $\{0,1\}^m$