Introduction à la Logique

A chaque proposition / énoncé / assertion, on associe une valeur de vérité soit vrai = true = 1 = oui soit faux = false = 0 = non.

Ex :

Assertion	La valeur de VÉRITÉ
$2 \leq 2$	1
$2 < 2$	0
x = 1 est une racine de l'équation : $x^2 - 2x + 1 = 0$ $(1)^2 - 2(1) + 1 = 0$ $1 - 2 + 1 = 0$	1

Notation :

· OU = OR = v (vee) = +

· ET = AND= ^ (wedge) = *

· NON = NOT = ¬ = ‘

· → = IMPLICATION = si … alors … = if … then …

· ↔ = ÉQUIVALENCE = si et seulement si

Dorénavant, les symboles p, q, r représentent les propositions quelconques.

Les Tables de VÉRITÉ

v = ou = or = + :

p	q	p v q
0	0	0
0	1	1
1	0	1
1	1	1

^ = et = and = * :

p	q	p ^ q
0	0	0
0	1	0
1	0	0
1	1	1

¬ = non = not = ‘ :

p	¬ p
0	1
1	0

Implication → :

· si p alors q,

· ou p → q,

· if p then q,

· p seulement si q,

· q est une condition nécessaire pour p,

· p est une condition suffisante pour q.

La TABLE de VÉRITÉ de p → q :

p	q	p → q
0	0	1
0	1	1
1	0	0
1	1	1

Équivalence :

p	q	p ↔ q
0	0	1
0	1	0
1	0	0
1	1	1

· v est lu DISJONCTION

· ^ est lu CONJONCTION

· ¬ est lu NÉGATION

La TABLE DE VÉRITÉ de ou exclusive (NOTATION : ⊕ / xor)

p	q	p ⊕ q
0	0	0
0	1	1
1	0	1
1	1	0

· p ⊕ q = p + q % 2

MODULO :

m % n = le reste de la division de l'entier m par l'entier n.

Ex :

10 % 5 = 0
12 % 5 = 2
5 % 7 = 5
17 % 3 = 2

TAUTOLOGIE :

C’est une proposition qui prend toujours la valeur vrai = 1

Ex : p v ¬ p est une tautologie.

p	¬ p	p v ¬ p
0	1	1
1	0	1

Il n’y a que 1, donc tautologie.

CONTRADICTION :

C’est une proposition qui prend toujours la valeur faux = 0

Ex : p ^ ¬ p est contradiction.

p	¬ p	p ^ ¬ p
0	1	0
1	0	0

Il n’y a que 0, donc contradiction.

La Réciproque de l'Implication

$p \rightarrow q$ est $q \rightarrow p$
¬

p	q	$p \rightarrow q$	$q \rightarrow p$
0	0	1	1
0	1	1	0
1	0	0	1
1	1	1	1

Il n'y a aucune relation entre $p \rightarrow q$ et sa réciproque $q \rightarrow p$

priorité des opérateurs :

Non ¬ a une priorité plus grand que ^ (et), ^ a une priorité plus grand que v.

Ex :

¬ p ^ q = (¬ p) ^ q mais ¬ p ^ q ≠ ¬ (p ^ q)

Analogie :

-2 + 3 = (-2) + 3
-2 + 3 ≠ (2 + 3)
2 × 3 + 5 = (2 × 3) + 5
2 × 3 + 5 ≠ 2 × (3 + 5)

1. Idempotence

p v p = p, p ^ p = p
p + p = p, p × p = p

(Analogie : A u A = A, A ∩ A = A, A un ensemble)

2. Associativité :

(p v q) v r = p v (q v r)
(p + q) + r = p + (q + r)

(Analogie : (A u B) u C = A u (B u C)

3. Commutativité :

p v q = q v p
p ^ q = q ^ p
p + q = q + p
pq + qp

(Analogie : A u B = B u A, A ∩ B = B ∩ A)

4. Distributivité :

p v (q ^ r) = (p v q) ^ (p v r)
p + qr = (p + q)(p + r)

(Cette loi est nouvelle. Ce n'est pas valable pour les nombres.)

Ex : $2 + 3 \times 4 \neq (2 + 3)(2 + 4)$

p ^ (q v r) = (p ^ q) v (p ^ r)
p(q + r) = pq + pr

Analogie :

A u (B ∩ C) = (A u B) ∩ (A u C)
A ∩ (B u C) = (A ∩ B) u (A ∩ C)

5. Absorption :

p v 1 = 1, p ^ 0 = 0
p + 1 = 1, p0 = 0

Analogie :

A u Ω = Ω, A ∩ Ø = Ø

6. Identité :

p ^ 1 = p, p u 0 =p
p1 = p, p + 0 = p

Analogie :

A ∩ Ω = A, A u Ø = A

7. Complémentation :

p v ¬ p = 1, p ^ ¬ p = 0
p + p' = 1, pp' = 0

Analogie :

0 = Ø, 1 = Ω

8. Involution :

¬ ¬ p = p
(p')' = p
p'' = p

Analogie : -(-2) = 2

9. Lois de Morgan :

¬ (p ^ q) = (¬ p) v (¬ q)
¬ (p v q) = (¬ p) ^ (¬ q)

(pq)’ = p’ + q’
(p + q)’ = p’q’

Ex : Montrer par la méthode ALGÉBRIQUE :

p v ¬ (p ^ q) = 1, TAUTOLOGIE

Solution : Montrons que p + (pq)’ = 1

p + (pq)’ = 1

p + (pq)’ = p + (p’ + q’) (Loi de Morgan)
= (p + p’) + q’ (Associativité)
= 1 + q’ = 1 (Absorption)

Argument valable / Fallacieux :

Notation : Un argument P1, P2, … , Pn |- Q (Conclusion)
Hypothèses

(P1, P2, … , Pn, Q sont des propositions composées est VALABLE si Q est vrai quant toutes les hypothèses sont vraies simultanément. Sinon, l’argument est fallacieux.

Exemple (Valable) :

Montrer que l’argument :

p, p → q |-- q (Loi de détachement) est VALABLE

P1, P2 |-- Q (Conclusion)
Hypothèses

La Table de Vérité :

P1 = p	**Q** = q	**P2** = p → q
0	0	1
0	1	1
1	0	0
1	1	1

Grace à la 4^{e} ligne l’argument est VALABLE.

Exemple (Fallacieux) :

Montrer que l'argument :

p → q, q |-- p est fallacieux.

P1, P2 |-- Q (Conclusion)
Hypothèses

La Table de VÉRITÉ :

Q = p	**P2** = q	**P1** = p → q
0	0	1
0	1	1
1	0	0
1	1	1

À cause de la 2e ligne, l'argument est FALLACIEUX.

Méthode Algébrique / Table de VÉRITÉ :

Théorème : L'argument P1, P2, … , Pn |-- Q est valable

ssi (P1 ^ P2 ^ … ^ Pn) → Q est une TAUTOLOGIE

Méthode Algébrique :

Montrons : (P1 ^ P2) → Q = 1

[p (p → q)] → q = 1

= p (p' + q) → q

= (pp' + pq) → q

= (0 + pq) → q

= (pq) → q

= (pq)' + q

= p' + (q +q')

= p + 1

= 1

Exemple (Valable) :

Montrer que l’argument :

p, p → q |-- q (Loi de détachement) est VALABLE

P1, P2 |-- Q

P1 = p	Q = q	P2 = p → q	P1 ^ P2	(P1 ^ P2) → Q
0	0	1	0	1
0	1	1	0	1
1	0	0	0	1
1	1	1	1	1

Exemple (Fallacieux) :

Montrer que l’argument :

p → q, q |-- p est fallacieux

La table de VÉRITÉ :

Q = p	P2 = q	P1 = p → q	P1 ^ P2	(P1 ^ P2) → Q
0	0	1	0	1
0	1	1	1	0
1	0	0	0	1
1	1	1	1	1

Définition (Théorème) :

Un théorème est une implication p → q qui est vrai (p, q sont des propositions qui sont vrai / faux). Notons que si p est faux, alors p → q est vraie.

Définition (Équivalence Logique) :

Deux propositions (composées) P1 et P2 sont logiquement équivalentes, si elles ont la même table de vérité.

Exemple :

p, $p \rightarrow q = \neg p \vee q$ P2 = p + q

p	q	$p \rightarrow q$	$\neg p$	$\neg p \vee q$
0	0	1	1	1
0	1	1	1	1
1	0	0	0	0
1	1	1	0	1

Fonction de proposition : p(x) :

A, le quantificateur universel
E, le quantificateur existentiel

A lire ‘pour tout’
E lire ‘il existe’

Ex :

Soit N = {0, 1, 2, 3, … }
Soit p(x) (une proposition p dépendant d’un élément x de N)

Soit p(x) : x + 2 > 7 (x + 2 > 7 (ou) x > 7 – 2 = 5)
Alors E = {6, 7, 8, … }

Fonction de proposition :

p(x) est une propriété concernant l’élément x

Notation : ($\forall x \in A$) p(x) est vrai (Lire pour tout élément x dans A p(x) est vrai)

= ($\forall x \in A$) p(x) (Notons que « est vrai » est implicite)

= $\forall \in x \in p(x)$

Si {x} $x \in A$, {(x)} = A alors
$\forall x$ p(x) sinon $\forall x$ p(x) est faux

Ex : E = L’ensemble d’étudiants présents

p(e) : l’étudiant e a moins de 40 ans

Donc E_v =E

(ou) $\{c \in E \mid p(c)\} = E$

Notation :

$(\exists\ e \in E)\ p(e)$ est vrai s'écrit simplement $(\exists\ e \in E)\ p(e)$ (est vrai est implicite)

Négation :

$\neg\ (\forall e \in E)$ (e est un garçon)

$= (\exists e \in E)$ (e n'est pas un garçon)

$= (\exists e \in E)\ p(e)$

$\neg\ (\exists e \in E)\ p(e)$

$= (\forall e \in E) \neg\ p(e)$

Remarque : Négation change $\forall$ en $\exists$
$\exists$ en $\forall$

Ex :

$\neg\ (\forall x\ \exists y\ \exists z,\ p(x, y, z))$

$= (\exists x\ \forall y\ \forall z,\ p(x, y, z))$

Négation :

Ex :

$\lim\limits_{n \to \infty} a_n = l$ Définition : $\forall \varepsilon > 0, \exists n_0 \in N, n_0 \in 12$

$\varepsilon = 0{,}0001$

$\lim\limits_{n \to \infty} a_n \neq l$ Définition : $\exists \varepsilon > 0, \exists n_0 \in N, n_0 \in 12$

$\forall n > n_0, |\ a_n - l\ | < \varepsilon$

$\exists n > n_0, |\ a_n - l\ | \geq \varepsilon$

Énoncé	Négation
∀	∃
∃	∀
$< \varepsilon$	$\geq \varepsilon$
$> \varepsilon$	$\leq \varepsilon$
$=$	$\neq$

Algèbre de Boole

(George Boole)

Définition :

Une algèbre de Boole est un ensemble avec aurons 2 éléments notés : 0 (élément zéro), 1 (élément unité) ($0 \neq 1$) avec 2 opérations Binaires.

+ (Addition), (Multiplication), et son opération binaire (noté ‘) complémentation avec les 4 propriétés suivantes :

1. Commutativité :

$a + b = b + a$, $\forall\ a,b \in B$
$ab = ba$

2. Associativité :

$a + (b + c) = (a + b) + c$ $\forall\ a,b,c \in B$
$a\ (bc) = (ab)\ c$

3. Distributivité :

$a\ (b + c) = ab + ac$
$a + bc = (a + b)\ (a + c)$

4. Élément neutre ou identité :

$a + 0 = a$
$a \times 1 = a$

5. Complémentation :

$a + a' = 1$
$aa' = 0$

On désigne l'algèbre de Boole par B (0, 1, +, ×, ')

Ex : (Bébé Boole)

B = {0, 1} Définition : +, ×, '

+	0	1
0	0	1
1	1	1

×	0	1
0	0	0
1	0	1

'	0	1
	1	0

Ex :

$B^2 = B \times B = \{00, 01, 10, 11\}$ ← tous les doublets de 0 et 1

$B^3 = B \times B \times B = \{000, 001, 010, 011, 100, 101, 110, 111\}$ ← tous les triplets de 0 et 1

Notons que B^3 possède 2^3 éléments.

D'après Boole : 010 + 110 = $\begin{array}{ccc} \underline{1} & \underline{1} & \underline{0} \\ 0+1 & 1+1 & 0+0 \end{array}$

110 + 001 = 111

110 × 001 = 000

111 × 010 = 010

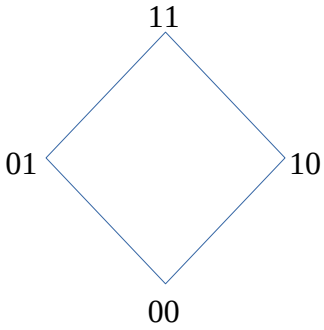
(010)' = 101

Élément zéro = 000
Élément unité = 111

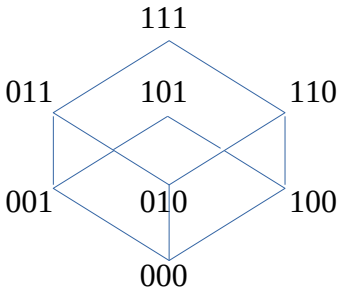
Diagramme de Hasse pour représenter l'algèbre de bool :

B = {0, 1}

B^2



B^3



Je trace un segment entre abc et def si : $a \leq d$; $b \leq e$; $c \leq f$

Pour calculer la somme, je prends l'ancêtre commun aux 2 nombres.

Exemple : 001 + 010 = 011

Pour calculer le produit, je prends leur descendant.

Exemple : 001 × 010 = 000

Considérons l'ensemble de tous les diviseurs positifs de l'entier 70 :

D_{70} = {1, 2, 5, 7, 10, 14, 35, 70}

L'ensemble comporte 8 éléments, c'est une puissance de 2. Lorsque les éléments sont une puissance des 2, je peux procéder en algèbre de bool.

On définit sur l'ensemble D_{70}, les opérations +, ×, unaire.

On a alors : $a + b = ppcm(a, b)$ $\quad a, b \in D_{70}$
$a \times b = pgcd(a, b)$
$\neg a = 70 \div a$

Exemple :

10 v 14 = 10 + 14 = ppcm(10, 14)
↔ 2 × 5 = 10 2 × 7 = 14
↔ 2 × 5 × 7 = 70
↔ ppmc(10, 14) = 70

10 ^ 14 = 10 × 14 = pgcd(10, 14)
↔ 2 × 5 = 10 2 × 7 = 14
↔ pgcd(10, 14) = 2

¬ 14 = (14)’ = 70 ÷ 14 = 5

10 v 5 = 10 + 5 = ppmc(10, 5)
↔ 10 = 2 × 5 5 = 1 × 5
↔ ppmc(10, 5) = 5 × 1 × 2
↔ ppmc(10, 5) = 10

10 ^ 5 = 10 × 5 = pgcd(10, 5)
↔ 10 = 2 × 5 5 = 1 × 5
↔ pgcd(10, 5) = 5

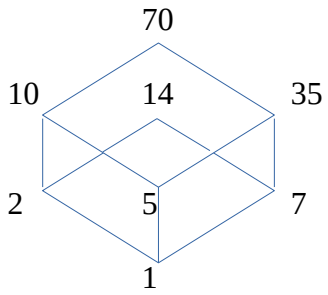
¬ 5 = 70 ÷ 5 = 14

2 v 35 = 2 + 35 = ppcm(2, 35)
↔ 2 = 2 × 1 35 = 5 × 7
↔ ppcm(2, 35) = 2 × 1 × 5 × 7
↔ ppcm(2, 35) = 70

2 ^ 35 = 2 × 35 = pgcd(2, 35)
↔ 2 = 2 × 1 35 = 7 × 5
↔ pgcd(2, 35) = 1

2’ = 70 ÷ 2 = 35

Traçons le diagramme de Hasse correspondant :



Il y a 2^3 éléments donc 3 étages.

Je relie a et b si a | b

Par convention, a est considéré comme ancêtre et descendant de lui-même.

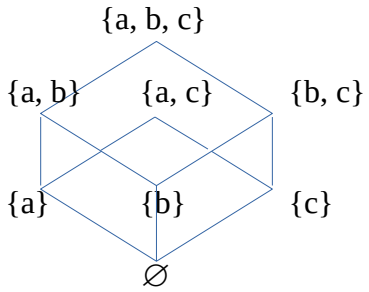
Théorème de Stone :

Un algèbre de Boole fini possède exactement 2^n éléments avec $n \geq 1$. Il n'y a qu'un seul algèbre de Boole d'après le principe d'isomorphisme sur 2^n éléments. L'ensemble d'éléments de 1 étage constitue l'ensemble d'atomes. La puissance n représente le nombre d'étages.

Exemple : Soit A = {a, b, c}

P(A) = {∅, {a}, {b}, {c}, {a, b}, {a, c}, {b, c}, {a, b, c}}

Puisqu'il y a 8 éléments soit 2^3, je peux définir les opérations +, ×, ' dans l'ensemble.



Donc l'élément zéro est ∅ et l'élément neutre {a, b, c}.
L'élément atome est le 1er étage {{a} , {b}, {c}}

Dualité

La dualité d'un énoncé dans l'algèbre de Boole est obtenue en échangeant :
+ par × ; × par + ; 1 par 0 ; 0 par 1

Exemple :

$(1 + a) \times (b + a) = b \rightarrow (0 \times a) + (b \times 1) = b$
$(0 \times a) + (1 \times b) = b \rightarrow (1 + a) \times (0 + b) = b$

Théorème (Principe de la dualité) :

La dualité d'un théorème dans l'algèbre de Boole est aussi est un théorème.
Un algèbre de Bool admet les propriétés suivantes :

· Idempotence

$a + a = a$ $\forall a \in B$
$aa = a$

· Absorption

$a + 1 = 1$ $\forall a \in B$
$a \times 0 = 0$
$a + ab = a$
$a (a + b) = a$

· Associativité

$a + (b + c) = (a + b) + c$ $\forall a, b, c \in B$
$a (bc) = (ab) c$

· Morgan

$(a + b)' = a'b'$ $\forall a, b \in B$
$(ab)' = a' + b'$

$(a')' = a$
$0' = 1$
$1' = 0$

$a + x = 1 \leftrightarrow x = a'$
$ax = 0$

$a + b = 0$
$ab' = 0$
$a = b$

La forme disjonctive normale ou la somme des produits vise à transformer une expression en expression plus simple.

Exemple :

E1 = (x + y'z)' + (xyz' + x'y)' x, y, z ∈ B

E2 = ((xy'z' + y)' + x'z)' x, y, z ∈ B

Littérale : Une variable x ou sa complémentation x' sont appelés littérales : x + x' = 1 ; xx' = 0

Produit fondamental : produit sans deux littérales de la même variable.

Exemple :

xz' ; xy'z ; xy' ; x'yz

Contre-exemple :

xyx'z (car il y a x et x')

xyzy (car il y a y et y)

Mais un produit non fondamental se transforme soit en 0 soit en 1 : un produit fondamental

Exemple :

· xyx'z (non-fondamental)

· x(yx')z (commutativité)

· (xx')yz = 0yz = 0

· xyzy = xyyz

= xyz par idempotence

· P1 = x'z P2 = x'yz

P1 et P2 sont fondamentaux

Notons que P1 est contenu dans P2 où que P2 contient P1

La somme P1 + P2 = x'z + x'yz
= x'z (car x'yz < x'z)

C'est la loi d'absorption : a + ab = a
une expression booléenne est dans la forme disjonctive normale si E = P1 + P2 + … + P_n où chaque produit P est fondamental et aucun P_i n'est dans l'autre.

Algorithme

Input (les ingrédients) : une expression booléenne E

Résultat : E dans la forme FCN

Algorithme : Recette

Step 1 : Utilisons les lois de Morgan et l'involution afin d'amener l'opération complémentation ' ou ¬ pour que chaque variable soit une littérale.

Step 2 : Utilisons la distributivité

Step 3 : Utilisons la commutativité, idempotence, complémentation

Step 4 : Utilisons l'absorption, l'identité

Application de l'algorithme

Écrire E = ((xy)'z)' ((x' + z) (y' + z'))' dans la forme FCN

Step 1 : E = ((xy)'z)' ((x' + z) (y' + z'))'

E = (((xy)')' + z') ((x' + z)' + (y' + z')')

E = (xy + z') (xz' + yz)

E = (xy + z') (xz' + yz)

Step 2 : xyxz' + xyyz + z'xz' + z'yz

Step 3 : yxxz' + xyyz + xz'z' + yzz'

yxz' + xyz + xz'

xz' + xyz

Step 4 : xyz + xz'

xy (z + z') + xz'

xy + xz'

Remarque : l'écriture FCN n'est pas unique.

La forme disjonctive normale complète est une expression booléenne E dans la forme FDN est dans la forme FDNC si chaque produit de E contient toutes les littérales.

Exemple : x'yz + xyz' est FDN mais pas FDNC

Théorème : **La FDNC est unique.**

Exemple :

Écrire sous la forme FDNC : E (x, y, z) = x (y'z)'

Step 1 : E (x, y, z) = x (y'z)'
= x (y'' + z')
= x (y + z')
= **xy + xz'** c'est la forme FDN

Step 2 : E (x, y, z) = xy + xz'
= xy (z + z') + xz' (y + y')
= xyz + xyz' + xyz' + xy'z'
= xyz + xyz' + xy'z'

Un produit dans lequel toutes les littérales sont présentes est un mintern.

Remarque : le nombre de minternes avec n variables x_1, x_2, … x_n est 2^n

Exemple : Écrivons toutes les 8 (2^3) minternes avec 3 variables x, y, z.

Pour ceci nous avons besoin de l'impliquant premier :

Un produit fondamental p est un impliquant premier de l'expression E si :
· E + P = E | · Aucun sous produit p' de p vérifie E + P = E.

Exemple : E = xy' + xyz' + x'yz'

Montrer que P = xz'

E + P = xy' + xyz' + x'yz' + xz' = **xy' + xz' + x'yz'**

Pour tester l'égalité, je complète :

xy' + xyz' + x'yz' et xy' + xz' + x'yz'

· xy' + xyz' + x'yz'
= xy' (z + z') + xyz' + x'yz'
= xy'z + xy'z' + xyz' + x'yz (FDNC)

· xy' + xz' + x'yz'
= xy' (z + z') + xz' (y + y') + x'yz'
= xy'z + xy'z' + xyz' + x'yz'

Théorème (Écriture Minimale) :

Une écriture minimale de somme de produits (Ex : E = xy + y'z + x'z') d'une expression booléenne et une somme des Ips. Fin de l'énoncé du Théorème.

Pour trouver les Ips d'une expression booléenne, nous avons besoin de la notion de « consensus de 2 produits Fondamentaux »

Ex (Consensus de 2 produits) :

Ex1 :

P1 = xyz's P2 = xy't

Compatible (consensus existe)

Le consensus de P1 et P2 = (xz's) (yt) = xz'st

Notation :

Un consensus est noté par Q ou Q1, Q2, …

Ex2 :

P1 = x'yz, P2 = x'yt

Non compatible

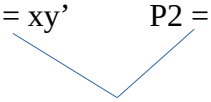
Définition (Consensus de z produits) :

Considérons z produits P1, P2 fondamentaux tels que exactement une littérale non-complémenté dans l'un des produits et complément dans l'autre.

Alors le consensus Q de P1 et P2 est obtenu par le produit de P1 et P2 après avoir éliminé les littérales compléments dans l'un et non-complémenté dans l'autre.

Ex :

P1 = xy' P2 = y1



Q = x1 = x

Ex :

P1 = x'yz, P2 = xyz'
Non-compatible = Q n'existe pas

Remarque :

Soient P1 = x, P2 = x Alors Q n'existe pas pour ces produits.

P1 = x, P2 =

Lemme (A faire en TD) :

Soit Q le consensus de P1 et P2.
Alors P1 + P2 + Q = P1 + P2

Algo permettant de trouver les Ips d'une expression F.

Input :

E = P1 + P2 + P3 + … + Pn (FDN)
où chaque Pi (i = 1, 2, … n) est un produit fondamental

Output :

E = Q1 + Q2 + … + Qk
(E comme somme des Ips)

Algo :

Step 1 : Recette, supprimer un produit Pi qui contient un autre produit Pj.
(Ex : xyz contient xz) (Petit mange costaud)

Step 2 : Ajout de consensus de 2 produits (grâce au Lemme).

Step 3 : Répétez Step 1 et Step 2 jusqu'à ce que on ne puisse plus appliquer ni Step 1 ni Step 2.

Ex (Ips) :

Trouvons les Ips de

E = xyz + x’z’ + xyz’ + x’y’z + x’yz’

Mange

E = xyz + x’y’ + xyz’ + x’yz

Q = (xy) (xy) = xy

E = xyz + x’z’ + xyz’ + x’y’z + xy (Ajout de Q)

Mange

E = x’z’ + x’y’z + xy

Q = (x’) (x’y’) = x’y

E = x’z’ + x’y’z + xy + x’y (Ajout de Q)

Mange

E = x’z’ + xy + x’y

Q = (z’) (y) = yz’

IP1 = x’z’, IP2 = xy,

IP3 = x’y, IP4 = yz’

L’Écriture Minimale :

Algo :

Input : E = IP1 + IP2 + … + Ipk

Où IP1, IP2, IPk, sont les IPs de E.

Output : Éliminons les IPs superflus et écrivons E avec les IPs qui restent.

Algo :

Step 1 : Écrivons chaque IP en FDNC. Autrement dit, complétons chaque IP.

Step 2 : Éliminons un par un les IPs dont les termes se trouvent dans d’autres IPs complétés.

Ex (Écriture Minimale) :

E = x’z’ + xy + x’y’ + yz’
E = IP1 + IP2 + IP3 + IP4 (k = 4)

Complétons chaque IPi.

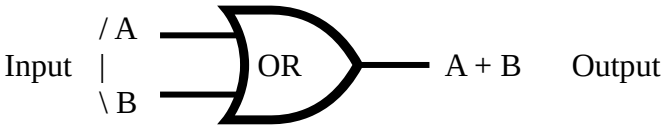
~~x’z’ = x’z’ (y + y’) = x’yz’ + x’y’z’~~ (Simplifie)
xy = xy (z + z’) = xyz + xyz’
x’y’ = x’y’ (z + z’) = x’y’z + x’y’z’
yz’ = yz’ (x + x’) = xyz’ + x’yz’

E = xy + x’y’ + yz’

Circuits Logiques :

Un circuit logique est une abstraction d'un circuit électronique d'un ordinateur.
Un circuit logique est constitué (Lego Blocks) de portes (Gates).
Chaque porte calcule une fonction particulière.

Porte : OR = OU

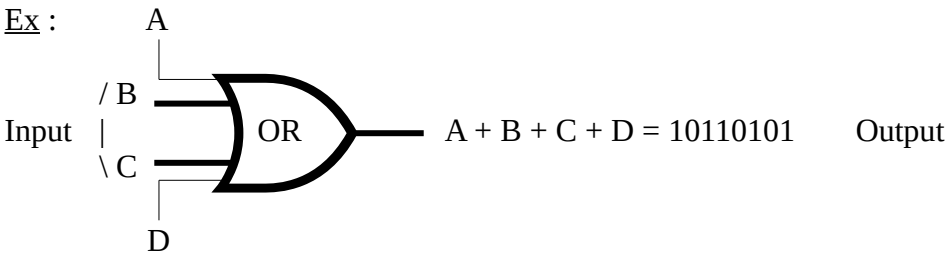


Manipuler bit (binary digit)

Byte = Octet

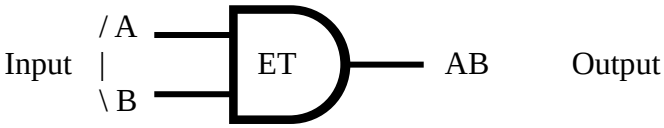
1 Byte = 8 Bits = Un caractère
4 Bytes = Un entier

Ex :

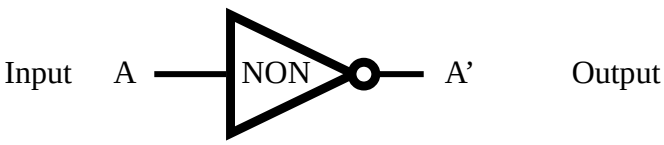


A = 10000101
B = 10100001
C = 00100100
D = 10010101

Porte : AND = ET



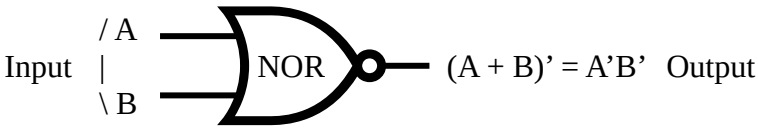
Porte : NOT = NON



Il n'y a que 3 portes de BASES.

Porte : NAND = NOT AND = NON ET

Porte : NOR = NOT OR = NON OU



A	B	NAND	NOR
0	0	1	1
0	1	1	0
1	0	1	0
1	1	0	0

Ex : Conséquence un circuit logique de l'expression y = ABC + AB'C + A'B

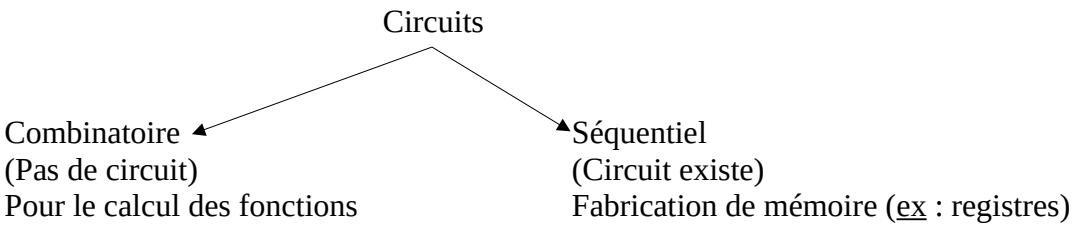
Input : A, B, C (3 Inputs)

y = la somme des produits

(schéma à insérer)

Circuits :

Les portes sont combinées pour former des circuits en connectant les sorties de certaines portes aux entrées d'autres portes.



Le graphe d'un circuit :

○ = petit cercle = une porte

Karnaugh Maps nous permet de trouver les Implicants premiers et l'écriture minimale d'une expression FDN complète.

Cas de 2 variables x, y :

	y	y'
x	xy	xy'
x'	x'y	x'y'

Les Bases de Rectangles :

1x1

	1x2

2x1

Ex : Trouver les IPS et l'écriture minimale de E(x, y) = xy + xy'

	y	y'
x	×	×
x'		

Step 1 : Cochons les cases correspondants aux termes de E.

Step 2 : Recouvrons les cases cochées par les Rectangles de Bases Maximales.

Step 3 :

xy ∩ xy' = x
IP1 = x
E = x

Ex : Trouver les Ips de E(x, y) = xy + x'y + x'y'

	y	y'
x	×	
x'	×	×

IP1 = xy ∩ x'y = y
IP2 = x'y ∩ x'y' = x'
E = y + x' = x' + y

Ex :

	y	y’
x	×	
x’		×

IP1 = x’y’
IP2 = xy

Ex :

E(x, y) = xy + x’y + xy’ + x’y’

	y	y’
x	×	×
x’	×	×

Ou

	y	y’
x	×	×
x’	×	×

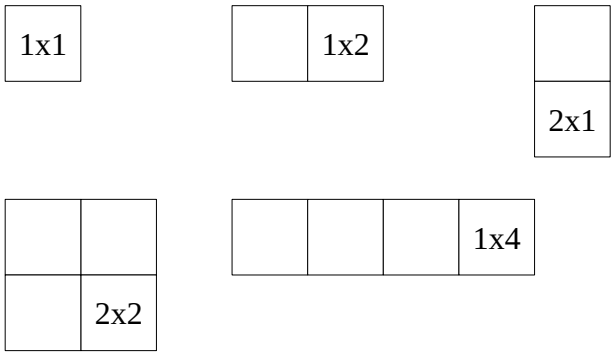
IP1 = xy ∩ xy’ = x
IP2 = x’y ∩ x’y’ = x’
E = x + x’ = 1

Karnaugh Maps

Cas 3 variables x, y, z :

	yz	yz’	y’z’	y’z
x	xyz	xyz’	xy’z’	xy’z
x’	x’yz	x’yz’	x’y’z’	x’y’z

Les Bases de Rectangles :



Ex :

E(x, y, z) = xy + x’yz’ + x’y’z’ + x’y’z

= xy (z + z’) + NPSSTP

= xyz + xyz’ + NPDSTP

	yz	yz’	y’z’	y’z
x	×	×		
x’		×	×	×

IP1 = xyz ∩ xyz’ = xy
IP2 = xyz’ ∩ x’yz’ = yz’
IP3 = x’yz’ ∩ x’y’z’ = x’z’
IP4 = x’y’z’ ∩ x’y’z = x’y’

Rq : 3 var x, y, z

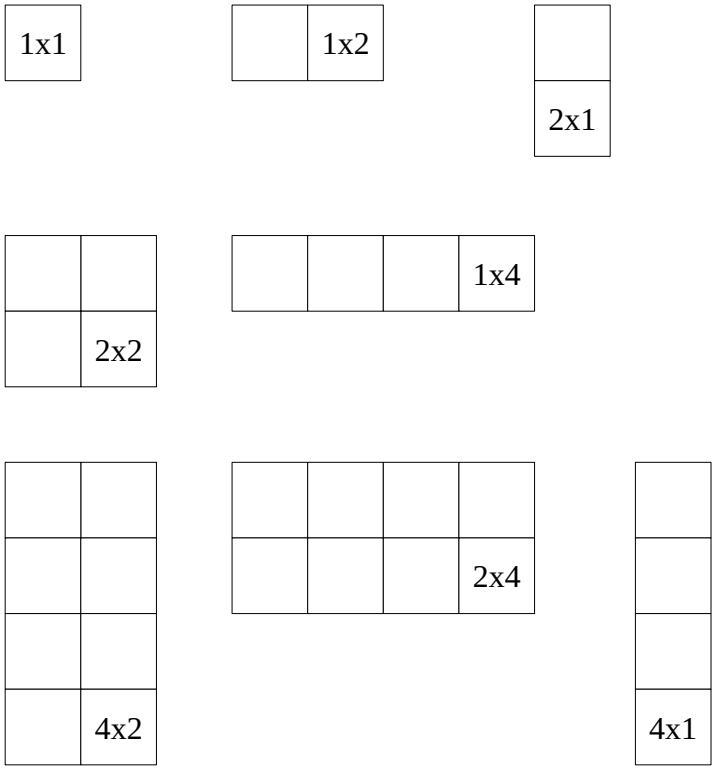
	yz	yz’	y’z’	y’z
x	×			×
x’	×			×

E = xyz + x’yz + xy’z + x’y’z IP1 = xyz ∩ x’yz ∩ xy’z ∩ x’y’z = z

Cas 4 variables x, y, z, t :

	zt	zt’	z’t’	z’t
xy				
xy’				
x’y’				
x’y				

Les Bases de Rectangles :



Rq :

Les cases de gauches et droites sont adjacentes.
Les cases de haut et bas sont adjacentes.

E = xy’ + xyz + x’y’z’ + x’yzt’ Trouver les IPs et écriture minimale

Complétons.

E = xy’ (z + z’) (t + t’) + xyz (t + t’) + x’y’z’ (t + t’) + NPDSTP

E = xy’zt +xy’zt’ + xy’z’t + xy’z’t’ + xyzt + xyzt’ + x’y’z’t + x’y’z’t’ + x’yzt’

	zt	zt’	z’t’	z’t
xy	×	×		
xy’	×	×	×	×
x’y’			×	×
x’y		×		