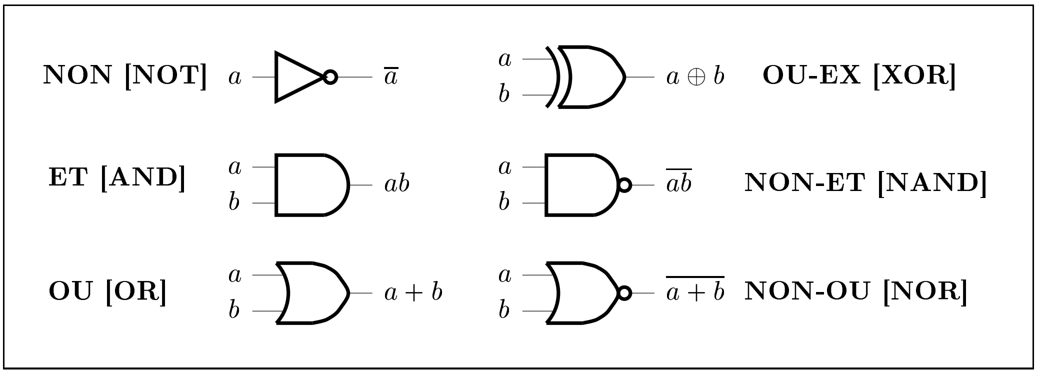
Travaux Dirigés n°1
Architecture des ordinateurs
—IR 2000—

Circuits combinatoires, algèbre de Boole, Tables de Karnaugh

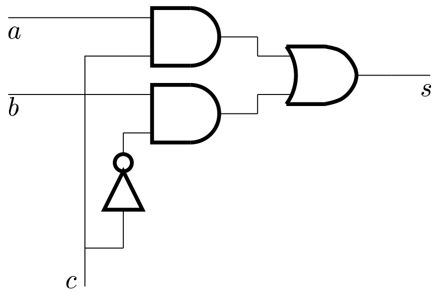
Dans cette séance de travaux dirigés, nous allons étudier le calcul des formules logiques (algèbre de Boole). En particulier, nous allons voir les différentes représentations d'une expression logique (circuits combinatoires, expressions algébriques, tables de vérité) et comment passer de l'une à l'autre (méthodes minterms, maxterms, tables de Karnaugh).

▸ Exercice 1. Opérations logiques élémentaires

La figure suivante donne le schéma des principaux opérateurs logiques :



1. Donner la table de vérité de chacun de ces opérateurs.
2. Donner la table de vérité du circuit ci-contre.
3. Combien y a-t-il d'opérateurs logiques unaires (à une seule entrée), binaires (à deux entrées) ? Donner leurs tables de vérité et proposer des noms.
4. Combien y a-t-il d'opérateurs logiques n-aires (à n entrées) ?



▸ Exercice 2. Relations fondamentales de l'algèbre de Boole

On rappelle que 1 désigne la constante VRAI, 0 la constante FAUX et que l'on note indifféremment $a \times b$ ou ab le produit logique (ET) de a et b, la somme logique (OU) étant notée $a + b$.

Règles des constantes	$a + 0 = a$	$a \times 0 = 0$
	$a + 1 = 1$	$a \times 1 = a$
Idempotence (duplication)	$a + a = a$	$a \times a = a$
Complémentation	$a + \overline{a} = 1$	$a \times \overline{a} = 0$
Commutativité	$a + b = b + a$	$a \times b = a \times b$
Distributivité	$a + (bc) = (a + b)(a + c)$	
	$a(b + c) = (ab) + (ac)$	
Associativité	$a + (b + c) = a + (b + c) = a + b + c$	
	$a(bc) = a(bc) = abc$	
Théorèmes de De Morgan	$\overline{ab} = \overline{a} + \overline{b}$	$\overline{a + b} = \overline{a}\,\overline{b}$
Autres relations	$\overline{\overline{a}} = a$	$(a + b)(a + \overline{b}) = a$

1. Vérifier l'associativité ainsi que la règle de De Morgan sur les tables de vérité.
2. Vérifier algébriquement les relations suivantes :

$$(a + b)(a + \overline{b}) = a\,, \qquad a + (ab) = a\,, \qquad a + (\overline{a}b) = a + b\,, \qquad a(a + b) = a\,.$$

▸ Exercice 3. Minterms-Maxterms

On appelle **minterm** le produit logique (ET) de plusieurs variables ou de leurs compléments (exemple : $ab\overline{c}$). On appelle **maxterm** la somme logique (OU) de plusieurs variables ou de leurs compléments (exemple : $a + b + \overline{c}$). On veut exprimer n'importe quelle fonction logique en utilisant seulement les fonctions ET, OU, et NON. On va tout d'abord étudier l'exemple de la fonction XOR.

1. Dans la méthode des minterms, on utilise le fait que la fonction peut s'exprimer comme la somme logique de tous les **minterms** correspondant à chaque sortie 1 dans la table de vérité. Donner l'expression du XOR par la méthode des minterms.
2. Dans la méthode des maxterms, on utilise le fait que la fonction peut s'exprimer comme le produit logique de tous les **maxterms** correspondant à chaque sortie 0 dans la table de vérité. Donner l'expression du XOR par la méthode des maxterms.

Les expressions ainsi obtenues seront dites sous **forme normale**.

3. Calculer avec les deux méthodes les expressions logiques dont la table de vérité est

a \ bc	00	01	10	11
0	1	0	1	0
1	0	1	1	0

4. Proposer une expression logique simplifiée, dessiner le circuit logique correspondant.

▸ Exercice 4. Opérateurs complets

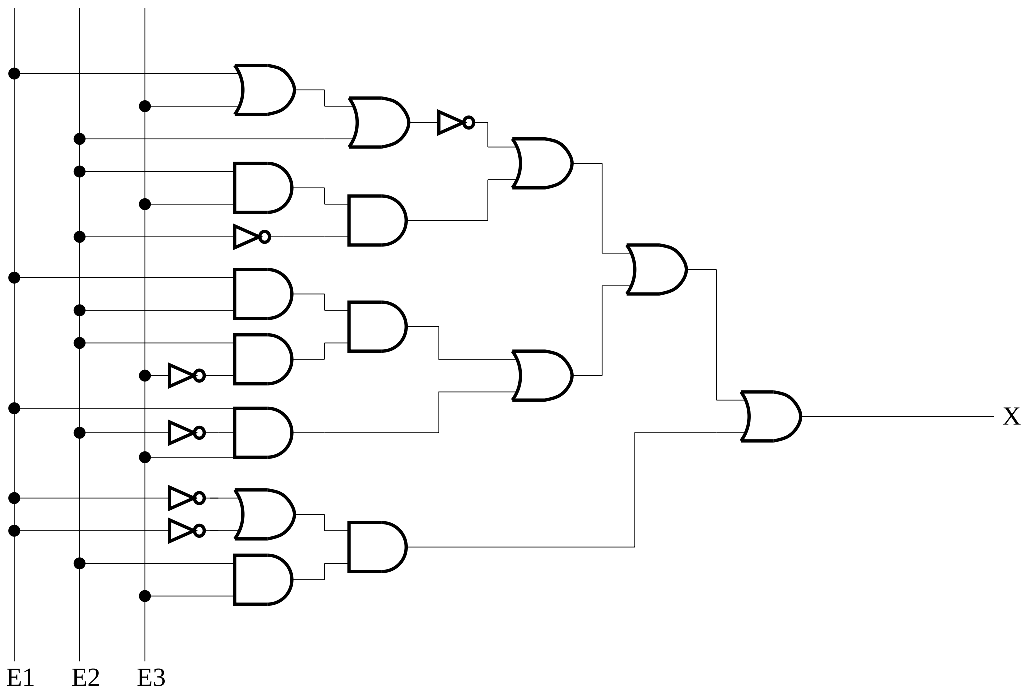
On vient de montrer que l'on peut réaliser n'importe quel opérateur logique en utilisant seulement des portes AND, OR, et NOT. En fait, cet ensemble n'est pas minimal :

1. Pourquoi ?

Nous allons montrer que l'on peut réaliser n'importe quel opérateur logique avec l'une seulement des deux fonctions suivantes : NAND (NON ET) et NOR (NON OU).

2. Donner les tables de vérité des opérateurs NAND et NOR.
3. Montrer que l'on peut réaliser les opérateurs NOT, OR et AND en utilisant seulement des NAND. Même question avec des NOR.
4. Conclure que l'on peut réaliser n'importe quel circuit.
5. Application : réaliser le XOR en utilisant seulement des portes NAND puis avec seulement des NOR.

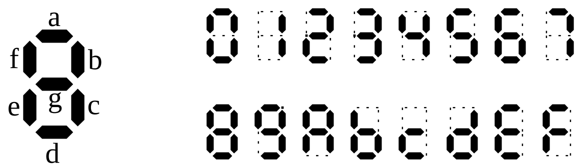
▸ Exercice 5. Analyse d'un circuit logique



1. Quelle est l'expression booléenne de la sortie X ?
2. Donner l'expression simplifiée et le circuit correspondant en n'utilisant que les opérateurs à deux entrées choisis parmis OR, AND, XOR et NOT.
3. Calculer la table de vérité de ce circuit.
4. Quel est son rôle ?

▸ Exercice 6. Simplification par la méthode des tables de Karnaugh : réalisation d'un afficheur

Une table de Karnaugh est une manière particulière de présenter les tables de vérité : au lieu d'écrire les entrées dans l'ordre usuel, on les écrit dans l'ordre 00, 01, 11, 10 où seul un bit change à chaque fois. Il est ainsi très facile de repérer les simplifications possibles. On cherche à réaliser un circuit afficheur hexadécimal pour une calculette. L'entrée est un nombre n en binaire sur 4 bits : b_0, b_1, b_2, b_3. Les 7 sorties sont appelées a, b, c, d, e, f, g. Une sortie est à 1 si le segment correpondant est noir.



1. Écrire les tables de vérité des 7 sorties.
2. En déduire le circuit correspondant.

▸ Exercice 7. Tables de Karnaugh partielles

On veut maintenant réaliser un afficheur **décimal**. Les sorties qui ne correspondent pas à une entrée décimale sont indéfinies. On peut donc choisir ce qui nous arrange pour avoir le circuit le plus simple.

1. Écrire les tables de vérité partielles des 7 sorties et les compléter pour avoir les formules les plus simples.
2. En déduire le circuit correspondant.

▸ Exercice 8. Le code Gray

Le but de cet exercice est de trouver un moyen pour énumérer toutes les configurations possibles des entrées d'un circuit logique de manière à ne changer qu'une seule entrée entre deux configurations consécutives.

1. Montrer que l'on peut écrire les quatre configurations de deux entrées en ne changeant qu'une entrée à chaque fois.
2. On définit alors **récursivement** le code Gray de la manière suivante :
   - Le code Gray d'une entrée seule est 0, 1 ;
   - Supposons que l'on sait calculer le code Gray d'un circuit à n entrées. On écrit alors les 2^n configurations correspondantes précédées par un 0 puis, les 2^n configurations précédées par un 1, **dans l'ordre inverse**.
3. Écrire le code Gray sur 2, 3 et 4 entrées, vérifier que cela répond bien au problème.
4. Comment calculer directement la p-ième configuration ?