

# LOGIQUE

## 1 introduction

La logique booléenne est à la base de tout système informatique. Il repose sur le fait que l'on eut facilement concevoir une système qui peut avoir 2 états. Par exemple dans un circuit électrique :

- Le courant passe,
- Le courant ne passe pas

Avant l'avènement des ordinateurs, des études théoriques sur le raisonnement logique avaient été menées avec plusieurs objectifs. Les premiers principes du raisonnement mathématique ont été érigés en une science par Aristote (384-322 av. J.C.). Des travaux complémentaires ont été faits durant l'antiquité par Zénon d'Élée, Autolique de Pitane, Eudème de Rhodes, Xénocrate.

Certaines parties du raisonnement aristotélicien restent floues. Leibniz (1646-1716) est le premier mathématicien à essayer de créer un système universel de raisonnement. Son but est de formaliser le raisonnement logique en introduisant des concepts simples. Son travail est incomplet et n'aboutit à rien de satisfaisant. Il vantera les mérites du calcul en base 2 mais ne sera pas suivi.

Il faudra attendre le 19<sup>ième</sup> siècle avant Boole (1815-1864) et De Morgan (1806-1871) pour voir la naissance de la logique mathématique indépendamment de la philosophie. Des études théoriques de Turing dans la première moitié du 20<sup>ième</sup> siècle montrent que toute automate à  $n$  états peut se ramener à un automate à 2 états.

## 2 Connecteurs logiques

### 2.1 Variables propositionnelles

définition : Les *variables propositionnelles* sont des propositions qui peuvent prendre la valeur **Vrai** ou **Faux** (**true** ou **false** en anglais).

exemple : «Les andorrans sont des européens» est une proposition vraie ne contenant aucune variable propositionnelle.

«L'élève  $x$  a une moyenne générale supérieure ou égale à 10» est une variable propositionnelle qui prend la valeur **Vrai** ou **Faux** suivant l'élève  $x$ .

«Je vous mens» n'est pas une variable propositionnelle quelque soit l'individu qui dit cette phrase.

définition : On appelle *connecteur logique* toute application de  $E^n$  dans  $E$  où  $E = \{\mathcal{V}, \mathcal{F}\}$ .

exemple :  $\begin{pmatrix} \text{Vrai} & \mapsto & \text{Vrai} \\ \text{Faux} & \mapsto & \text{Vrai} \end{pmatrix}$  est un connecteur logique de  $E$  dans  $E$ .

définition : Si  $f$  est un connecteur logique de  $E^n$  dans  $E$ ,  $n$  est appelé l'arité de  $f$ . Elle correspond au nombre de paramètres de la fonction  $f$ .

propriété: le nombre de connecteurs logiques d'arité  $n$  est égal à :

$$\text{Card}(\mathcal{F}(E^n, E)) = \text{Card}(E)^{\text{Card}(E^n)} = 2^{2^n}$$

Exemples Il existe  $2^{2^1} = 4$  connecteurs logiques de  $E$  dans  $E$  d'arité 1.

a	Vrai(a)	$Id_E(a)$	Non(a)	Faux(a)
$\mathcal{V}$	$\mathcal{V}$	$\mathcal{V}$	$\mathcal{F}$	$\mathcal{F}$
$\mathcal{F}$	$\mathcal{V}$	$\mathcal{F}$	$\mathcal{V}$	$\mathcal{F}$

On a donc deux fonctions constantes, l'identité et la fonction de négation appelé NON (ou bien NOT en anglais).

Il existe  $2^{2^2} = 2^4 = 16$  connecteurs logiques d'arité 2. Seules quelques unes sont d'un usage courant et ont reçu un nom.

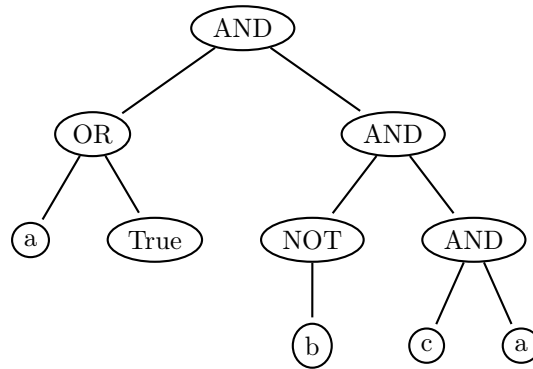
$$\begin{aligned}
 ET = AND &= \begin{pmatrix} (\mathcal{V}, \mathcal{V}) \rightarrow \mathcal{V} \\ (\mathcal{V}, \mathcal{F}) \rightarrow \mathcal{F} \\ (\mathcal{F}, \mathcal{V}) \rightarrow \mathcal{F} \\ (\mathcal{F}, \mathcal{F}) \rightarrow \mathcal{F} \end{pmatrix} & OU = OR &= \begin{pmatrix} (\mathcal{V}, \mathcal{V}) \rightarrow \mathcal{V} \\ (\mathcal{V}, \mathcal{F}) \rightarrow \mathcal{V} \\ (\mathcal{F}, \mathcal{V}) \rightarrow \mathcal{V} \\ (\mathcal{F}, \mathcal{F}) \rightarrow \mathcal{F} \end{pmatrix} \\
 NONET = NAND &= \begin{pmatrix} (\mathcal{V}, \mathcal{V}) \rightarrow \mathcal{F} \\ (\mathcal{V}, \mathcal{F}) \rightarrow \mathcal{V} \\ (\mathcal{F}, \mathcal{V}) \rightarrow \mathcal{V} \\ (\mathcal{F}, \mathcal{F}) \rightarrow \mathcal{V} \end{pmatrix} & OUX = XOR &= \begin{pmatrix} (\mathcal{V}, \mathcal{V}) \rightarrow \mathcal{F} \\ (\mathcal{V}, \mathcal{F}) \rightarrow \mathcal{V} \\ (\mathcal{F}, \mathcal{V}) \rightarrow \mathcal{V} \\ (\mathcal{F}, \mathcal{F}) \rightarrow \mathcal{F} \end{pmatrix} \\
 \Rightarrow &= \begin{pmatrix} (\mathcal{V}, \mathcal{V}) \rightarrow \mathcal{V} \\ (\mathcal{V}, \mathcal{F}) \rightarrow \mathcal{F} \\ (\mathcal{F}, \mathcal{V}) \rightarrow \mathcal{V} \\ (\mathcal{F}, \mathcal{F}) \rightarrow \mathcal{V} \end{pmatrix} & \iff &= \begin{pmatrix} (\mathcal{V}, \mathcal{V}) \rightarrow \mathcal{V} \\ (\mathcal{V}, \mathcal{F}) \rightarrow \mathcal{F} \\ (\mathcal{F}, \mathcal{V}) \rightarrow \mathcal{F} \\ (\mathcal{F}, \mathcal{F}) \rightarrow \mathcal{V} \end{pmatrix}
 \end{aligned}$$

définition: Une proposition logique (ou formule logique) est :

- Une constante égale à **vrai** ou **faux**,
- une variable propositionnelle qui pourra prendre la valeur **vrai** ou **faux**
- une expression de la forme :
  1. NOT( $E$ ) où  $E$  est une proposition logique
  2.  $E$  AND  $F$  où  $E$  et  $F$  sont des propositions logiques
  3.  $E$  OR  $F$  où  $E$  et  $F$  sont des propositions logiques

La notion de connecteur logique permet de construire de nouvelles propositions logiques. Chaque proposition logique peut se représenter sous la forme d'un arbre où les feuilles sont soit des constantes soit des variables propositionnelles et les nœuds ont une ou deux branches contenant un connecteur logique.

exemple: (a OR true) AND ((NOT b) AND (c AND a)) se représente sous forme arborescente :



Une proposition logique est en fait une application de  $E^p$  dans  $E$  où  $p$  est le nombre de variables propositionnelles distinctes dans la proposition logique  $P$ . Elle est caractérisée par sa table de vérité qui contient  $(p+1)$  lignes et  $2^p$  colonnes.

$a_1$	$a_2$	...	$a_p$	$P(a_1, \dots, a_p)$
$\mathcal{V}$	$\mathcal{V}$	...	$\mathcal{V}$	$P(\mathcal{V}, \mathcal{V}, \dots, \mathcal{V})$
$\mathcal{V}$	$\mathcal{V}$	...	$\mathcal{F}$	$P(\mathcal{V}, \mathcal{V}, \dots, \mathcal{F})$
$\vdots$	$\vdots$		$\vdots$	$\vdots$
$\mathcal{F}$	$\mathcal{F}$	...	$\mathcal{F}$	$P(\mathcal{F}, \mathcal{F}, \dots, \mathcal{F})$

réciroquement : toute fonction booléenne est une proposition logique. Pour fabriquer la proposition logique à partir d'une fonction booléenne de  $E^p$  à valeur dans  $E$ , on représente notre fonction sous forme de table de vérité en notant  $a_1, a_2, \dots, a_p$  les  $p$  composantes des éléments de l'ensemble de départ  $E^p$ . On garde tous les cas où le résultat est vrai que l'on traduit sous forme conjonctive (  $a_1$  et non( $a_2$ ) et...  $a_p$  par exemple) que l'on relie entre elles par des «ou».

exemple : considérons la fonction  $f$  de  $E^2$  à valeurs dans  $E$  représentée par le tableau :

$a_1$	$a_2$	$f(a_1, a_2)$
$\mathcal{V}$	$\mathcal{V}$	$\mathcal{F}$
$\mathcal{V}$	$\mathcal{F}$	$\mathcal{V}$
$\mathcal{F}$	$\mathcal{V}$	$\mathcal{V}$
$\mathcal{F}$	$\mathcal{F}$	$\mathcal{F}$

La deuxième ligne du tableau nous donne : ( $a_1$  et non( $a_2$ )).

La troisième ligne du tableau nous donne : (non( $a_1$ ) et  $a_2$ ).

En final, on a :  $f(a_1, a_2) = (a_1$  et non( $a_2$ )) ou (non( $a_1$ ) et  $a_2$ ).

Vérifier qu'une proposition logique est vraie, c'est vérifier qu'elle est toujours vraie quelles que soient les valeurs des variables propositionnelles qui la composent. Si la proposition contient  $p$  variables distinctes, on doit évaluer  $2^p$  expressions booléennes.

définition : On appelle *tautologie* toute proposition logique vraie quelles que soient les valeurs des variables propositionnelles qui la composent.

exemple:  $(a \text{ ou } \text{non}(a))$  est une tautologie.

## 2.2 Complétude

définition : On dit qu'un système de connecteurs logiques est complet si toute proposition logique peut s'exprimer comme proposition logique utilisant ces connecteurs et seulement ces connecteurs.

remarque: On peut aussi utiliser les constantes vrai et faux dans les propositions logiques.

exemple: par définition  $\{\text{AND}, \text{OR}, \text{NOT}\}$  forme un système complet de connecteurs.

propriété: Pour montrer qu'un système de connecteurs est complet, il faut et il suffit de prouver que les fonctions logiques  $\text{non}(a)$ ,  $(a \text{ et } b)$  et  $(a \text{ ou } b)$  peuvent s'exprimer en utilisant que les connecteurs du système.

théorème:  $\{\text{NAND}\}$  est un système complet de connecteur

démonstration:  $\text{NOT}(a) \equiv a \text{ NAND vrai}$

$$a \text{ AND } b \equiv \text{NOT}(a \text{ NAND } b) \equiv (a \text{ NAND } b) \text{ NAND vrai}$$

$$a \text{ OR } b \equiv \text{NOT}(\text{NOT}(a) \text{ AND } \text{NOT}(b)) \equiv ((a \text{ NAND vrai}) \text{ NAND } (b \text{ NAND vrai}) \text{ )NAND true NAND } b \text{ NAND vrai}$$

## 2.3 Représentation des expressions logiques en Caml

```

1  #type constante = Vrai | Faux;;
2  Le type constante est défini.
   #Vrai;;
4  - : constante = Vrai

   #type proposition =
6    Cst of constante
   | variable of string
8   | Non of proposition
   | Et of (proposition*proposition)
10  | Ou of (proposition*proposition);;
   Le type proposition est défini.

12  #let et a b = Et(a,b);;
   et : proposition -> proposition -> proposition = <fun>
14  ##infix "et";;
   #let a = variable "a";;
16  a : proposition = variable "a"

```

```

#let vrai = Cst Vrai;;
18 vrai : proposition = Cst Vrai
#a et vrai;;
20 - : proposition = Et (variable "a", Cst Vrai)

```

### 3 Tableau récapitulatif des expressions logiques.

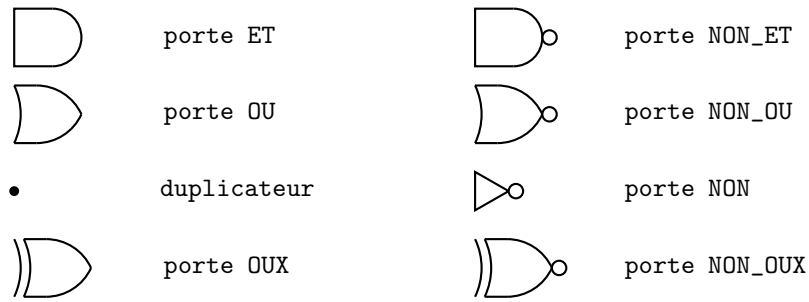
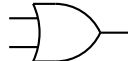

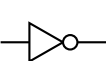




FIG. 1 – Symbologie des portes logiques.

nom français	OU	ET	NON	OUX	NONOU
nom anglais	OR	AND	NOT	XOR	NOR
symbole					
symbole	$\vee$	$\wedge$	$\neg$	$\otimes$	

### 4 formes disjunctive et conjonctive

définition : On appelle *disjonction* toute formule logique  $F$  s'écrivant  $F_1 \vee F_2 \cdots \vee F_n$  où chaque  $F_i$  est une variable propositionnelle  $p_i$  ou la négation d'une variable propositionnelle  $\neg p_i$ .

définition : On appelle *conjonction* toute formule logique  $F$  s'écrivant  $F_1 \wedge F_2 \cdots \wedge F_n$  où chaque  $F_i$  est une variable propositionnelle  $p_i$  ou la négation d'une variable propositionnelle  $\neg p_i$ .

remarque : Si une variable propositionnelle  $p$  apparaît plusieurs fois dans une disjonction ou dans une conjonction, on peut transformer la formule logique  $F$  correspondante en une formule dans laquelle  $p$  n'apparaît qu'une seule fois ou pas du tout en appliquant les règles suivantes :

$$\begin{array}{lll}
 p \wedge p \equiv p & p \wedge \neg p \equiv \mathcal{F} & \neg p \wedge \neg p \equiv \neg p \\
 p \vee p \equiv p & p \vee \neg p \equiv \mathcal{V} & \neg p \vee \neg p \equiv \neg p
 \end{array}$$

Règles de simplification :

Si  $F_1$  et  $F_2$  sont deux formules logiques, on a :

- $\neg(\neg F_1) \equiv F_1$
- $\neg(F_1 \wedge F_2) \equiv (\neg F_1) \vee (\neg F_2)$  et  $\neg(F_1 \vee F_2) \equiv (\neg F_1) \wedge (\neg F_2)$  (lois de Morgan)
- $\neg(F_1) \wedge F_1 \equiv \mathcal{F}$  et  $\neg(F_1) \vee F_1 \equiv \mathcal{V}$

propriété Si  $F$  est une conjonction alors  $\neg F$  est une disjonction

théorème I. Toute formule logique  $F$  est équivalente à une conjonction de disjonctions, c'est à dire à une formule  $F' = C_1 \vee \dots \vee C_n$  où les  $C_i$  sont des conjonctions. On parle alors de *forme normale conjonctive*.

théorème II. Toute formule logique  $F$  est équivalente à une disjonction de conjonctions, c'est à dire à une formule  $F' = C_1 \wedge \dots \wedge C_n$  où les  $C_i$  sont des disjonctions. On parle alors de *forme normale disjonctive*.

démonstration d'après le théorème I,  $\neg F$  est de la forme  $C_1 \vee \dots \vee C_n$  où les  $C_i$  sont des conjonctions. D'où  $F \equiv (\neg C_1) \wedge \dots \wedge (\neg C_n)$  et chaque  $\neg C_i$  est une disjonction comme négation d'une conjonction.

exemple : Considérons la fonction logique suivante :

$a$	$b$	$c$	$f(a, b, c)$
$\mathcal{V}$	$\mathcal{V}$	$\mathcal{V}$	$\mathcal{F}$
$\mathcal{V}$	$\mathcal{V}$	$\mathcal{F}$	$\mathcal{V}$
$\mathcal{V}$	$\mathcal{F}$	$\mathcal{V}$	$\mathcal{V}$
$\mathcal{V}$	$\mathcal{F}$	$\mathcal{F}$	$\mathcal{F}$
$\mathcal{F}$	$\mathcal{V}$	$\mathcal{V}$	$\mathcal{V}$
$\mathcal{F}$	$\mathcal{V}$	$\mathcal{F}$	$\mathcal{F}$
$\mathcal{F}$	$\mathcal{F}$	$\mathcal{V}$	$\mathcal{V}$
$\mathcal{F}$	$\mathcal{F}$	$\mathcal{F}$	$\mathcal{V}$

Elle a pour forme normale disjonctive :

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

Le calcul de la forme normale disjonctive de  $\neg f(a, b, c)$  nous donne :

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

Ce qui nous donne en en prenant la négation la forme normale conjonctive suivante :

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