Chapitre 23. Boole (hors-programme)

Dans ce chapitre nous allons étudier dans un premier temps les algèbres de Boole et les fonctions booléennes, dans un second temps la logique propositionnelle, aussi appelée logique des énoncés ou encore logique booléenne.

Partie I : Algèbre de Boole & fonctions booléennes

L'intérêt d'étudier l'algèbre de Boole en informatique, est assez évident si l'on se souvient qu'un ordinateur est un système électronique fonctionnant à l'électricité. De ce fait, il est facile de comprendre que le courant circule ou non dans un circuit. Qu'à partir de cette information présence ou absence de courant il est possible de représenter des nombres, des caractères, d'effectuer des calculs numériques ou de faire de la logique. Ceci permet de disposer de langage de programmation permettant d'utiliser des tests, de faire des opérations numériques ou de décider s'il faut ou non continuer les calculs, dans ce dernier cas cela a pour conséquence une accélération des calculs en utilisant le principe de l'évaluation paresseuse « lazy evaluation ».

23.1 Algèbres de Boole

Rappelons qu'une opération $\bullet$ est commutative si $x \bullet y = y \bullet x$; elle est associative si $x \bullet (y \bullet z) = (x \bullet y) \bullet z$; idempotente si $x \bullet x = x$; distributive par rapport à une opération $\otimes$ si $x \bullet (y \otimes z) = (x \bullet y) \otimes (x \bullet z)$.

Définition 23.1.1 Une algèbre de Boole c'est :

- un ensemble E ;
- deux éléments disctincts de E, notés $\bot$ et $\top$;
- deux opérations binaires sur E, notées $\sqcap$ et $\sqcup$;
- une opération unaire sur E, notée $\bar{}$.

Qui vérifient les points suivants :

1. les opérations $\sqcap$ et $\sqcup$ sont idempotentes, associatives, commutatives et distributives l'une par rapport à l'autre ;
2. $x \sqcap (x \sqcup y) = x = (y \sqcap x) \sqcup x$;
3. $x \sqcap \bot = \bot$, $x \sqcup \bot = x$, $x \sqcap \top = x$, $x \sqcup \top = \top$;
4. $x \sqcap \bar{x} = \bot$, $x \sqcup \bar{x} = \top$.

◀

On écrira une telle algèbre sous la forme B = (E,$\sqcap$, $\sqcup$, $\bar{}$, $\bot$, $\top$). Certaines propriétés peuvent être déduites des autres (voir feuille d'exercices).

Propriété 23.1 Dans une algèbre de Boole chaque élément x, possède un unique élement complémentaire, noté $\bar{x}$. ♦

Propriété 23.2 (loi de De Morgan) Dans une algèbre de Boole $B = (E, \sqcap, \sqcup, \bar{}, \bot, \top)$ on a les propriétés suivantes :

- $\forall e \in E, \forall e' \in E, \overline{e \sqcap e'} = \bar{e} \sqcup \overline{e'}$
- $\forall e \in E, \forall e' \in E, \overline{e \sqcup e'} = \bar{e} \sqcap \overline{e'}$

♦

23.1.1 Algèbre de Boole minimale

Il existe une algèbre de Boole particulière, c'est l'algèbre minimale, notée $\mathbb{B}$. En effet, $\mathbb{B}$ ne contient que deux éléments, notés classiquement 0 et 1, de même les opérations $\sqcap$ et $\sqcup$ sont notées . et +. La table 23.2 décrit le résultat des opérations :

x	y	$x+y$	$x.y$	$\bar{x}$
0	0	0	0	1
0	1	1	0	1
1	0	1	0	0
1	1	1	1	0

TABLE 23.1 – Opérations dans $\mathbb{B}$

On donne très souvent l'interprétation [1] suivante pour cette algèbre : 0 correspond à `faux`, 1 à `vrai`, + est le `ou logique` qui se note $\vee$, . est le `et logique` qui se note $\wedge$, enfin $\bar{}$ correspond à la négation $\neg$. ces notations seront reprises dans la section 23.2.1 sur la logique propositionnelle.

Simplifications d'écriture
Nous allons explorer, quelques règles de simplification dans le cas particulier de l'algèbre minimale. Ces simplifications découlent directement de la définition 23.1.1. Et seront utilisées dans la suite.

1. $x.(x+y) = x = (y.x) + x$
2. $x.0 = 0, x + 0 = x, x.1 = x, x + 1 = 1$
3. $x.\bar{x} = 0, x + \bar{x} = 1, \bar{\bar{x}} = x$

23.2 Fonctions booléennes

On considère dans ce qui suit l'algèbre de Boole $\mathbb{B}$. Une fonction boolénne à n arguments est une fonction de $\mathbb{B}^n$ dans $\mathbb{B}$. Cette fonction est entièrement définie par les n-uplets de $\mathbb{B}^n$ où elle prend la valeur 1.

Exemple 23.1
La fonction $f(x, y)$ définie par $f(0, 1) = f(1, 0) = f(1, 1) = 1$ correspond à l'opération +; de même l'opération . correspond à la fonction $g(x, y)$ définie par $g(1, 1) = 1$. †

Définition 23.2.1 (Forme polynomiale) Une fonction booléenne est polynomiale si elle vaut 0 en tout point ou si elle peut s'écrire comme une combinaison de ses arguments à l'aide des trois opérations de base. ◀

1. une interprétation est une application qui permet de donner un sens (une signification) à quelque chose

Théorème 23.1 *Toute fonction booléenne est polynomiale* ◊

La preuve de ce théorème est laissée en exercice (voir fiche d'exercices), et repose sur la propriété 23.3 suivante :

Propriété 23.3 Soit f une fonction booléenne à $k+1$ arguments, on a :

$$f(x_0, x_1, \ldots, x_k) = \overline{x_0} f(0, x_1, \ldots, x_k) + x_0 f(1, x_1, \ldots, x_k)$$

♦

Cette propriété est connue aussi sous le nom de « décomposition de Shannon » ; plus précisément cette décomposition s'écrit :

$$\begin{aligned} f(x_0, \ldots, x_{i-1}, x_i, x_{i+1}, \ldots, x_k) = \quad & \overline{x_i} f(x_0, \ldots, x_{i-1}, 0, x_{i+1}, \ldots, x_k) \quad + \\ & x_i f(x_0, \ldots, x_{i-1}, 1, x_{i+1}, \ldots, x_k) \end{aligned}$$

Exemple 23.2
Soit la fonction $f(x, y)$ définie par $f(0,0) = f(0,1) = f(1,0) = 1$. D'après la propriété 23.3, $f(x,y) = \bar{x} f(0,y) + x f(1,y)$; $f(0,y) = \bar{y} f(0,0) + y f(0,1) = \bar{y} + y$ et $f(1,y) = \bar{y} f(1,0) + y f(1,1) = \bar{y}$ donc $f(x,y) = \bar{x}.\bar{y} + \bar{x}.y + x\bar{y}$. †

> **Attention** le théorème 23.1 n'a pas comme objectif de trouver la plus petite expression possible sous forme polynomiale. La fonction présentée dans l'exemple 23.2 s'appelle communément `nand` pour « *non et* » et peut s'écrire $\overline{x.y} = \bar{x} + \bar{y}$.

Remarque 23.2.1
Il est facile d'exprimer une fonction booléenne f à n arguments, sous forme polynomiale. Pour cela on considère tous les n-uplets pour lesquels f vaut 1 ; soit $D_f = \{(x_1, \ldots, x_n) \in \mathbb{B}^n$ tel que $f(x_1, \ldots, x_n) = 1\}$. Si D_f est vide, la fonction est nulle ; sinon pour chaque n-uplets $\overrightarrow{z}$ de D_f on construit la fonction $h_{\overrightarrow{z}} = y_1 y_2 \ldots y_n$, avec $y_i = x_i$ si $x_i = 1$ et $y_i = \overline{x_i}$ sinon. On vérifie aisément que $f(x_1, \ldots, x_n) = \sum_{\overrightarrow{z} \in D_f} h_{\overrightarrow{z}}(x_1, \ldots, x_n)$, en remarquant que dans $\mathbb{B}$ une somme vaut 1 dès que l'un des facteurs vaut 1 ; et que $\sum_{\overrightarrow{z} \in D_f} h_{\overrightarrow{z}}(x_1, \ldots, x_n)$ vaut 1 si et seulement si $(x_1, \ldots, x_n)$ appartient à D_f et donc si et seulement si $f(x_1, \ldots, x_n) = 1$ ∘

Exemple 23.3
Reprenons l'exemple 23.2, soit la fonction $f(x, y)$ définie par $f(0,0) = f(0,1) = f(1,0) = 1$. Selon la remarque 23.2.1, $f(x,y) = \bar{x}.\bar{y} + \bar{x}.y + x.\bar{y}$.
En factorisant $\bar{x}$ sur les deux premiers termes, et en factorisant $\bar{y}$ sur le premier et le troisième, on obtient $f(x,y) = \bar{x}(\bar{y}+y) + \bar{y}(\bar{x}+x)$; comme $z + \bar{z} = 1$, pour tout z, on obtient $f(x,y) = \bar{x} + \bar{y}$. †

Définition 23.2.2 (fonction duale) Soit f une fonction booléenne à n arguments, sa fonction duale notée $\hat{f}$ est définie par :

$$\hat{f}(x_1, \ldots, x_n) = \overline{f(\overline{x_1}, \ldots, \overline{x_n})}$$

◄

Exemple 23.4
La duale de $y + \bar{x}$ est $\overline{\bar{y} + \bar{\bar{x}}}$
La duale de $\bar{x}$ est $\bar{\bar{x}} = \bar{x}$ †

Propriété 23.4 Cette propriété tient en deux points :

1. Si g est la duale de f, alors f est la duale de g.
2. Si $f(x_1, \ldots, x_p) = g(g_1(x_1, \ldots, x_p), \ldots, g_k(x_1, \ldots, x_p))$ alors
   $\hat{f}(x_1, \ldots, x_p) = \hat{g}(\hat{g_1}(x_1, \ldots, x_p), \ldots, \hat{g_k}(x_1, \ldots, x_p))$

♦

La preuve de la propriété 23.4 est laissée en exercice (voir fiche d'exercices). Une conséquence immédiate est que *si f est sous forme polynomiale, alors pour trouver sa duale, il suffit de remplacer toutes les sommes par des produits et tous les produits par des sommes.*

Exemple 23.5
La duale de $x.y + \bar{x}$ est $(x + y).\bar{x} = y.\bar{x}$. Pour s'en convaincre, le lecteur calculera les valeurs des trois fonctions $\overline{\bar{x}.\bar{y} + \bar{\bar{x}}}$; $(x+y).\bar{x}$ et $y.\bar{x}$; il vérifiera que pour toute valeur de x et y les trois fonctions sont identiques. †

23.2.1 Mintermes, maxtermes

Cette partie introduit la notion de mintermes et de maxtermes afin de permettre un calcul plus rapide de la mise en forme des fonctions booléennes sous forme polynômiale [6] pp129–148. L'idée est de faire le lien entre fonction booléenne et mot binaire. Nous noterons $\mathcal{B} = \{0, 1\}$, $\mathcal{B}^n$ désignera le produit cartésien $\overbrace{\mathcal{B} \times \ldots \times \mathcal{B}}^{n \text{ fois}}$.
$(\mathcal{B}^n, \preceq)$ où $\preceq$ est l'ordre lexicographique induit par $0 \leq 1$. Ainsi, pour $n = 3$ l'ordre est $000, 001, 010, 011, 100, 101, 110, 111$.

Une fonction booléenne ayant n paramètres peut être vue comme une fonction associant à un mot binaire de longueur n, une valeur booléenne. On note dans la suite $\mathbb{F}_n$, l'ensemble des fonctions booléennes à n paramètres.
Puisqu'une fonction booléenne est à valeur dans $\{0, 1\}$ il est possible de l'interpréter comme la fonction caractéritique d'une partie de $\{0, 1\}^n$.

Propriété 23.5 Une fonction booléenne de $\mathbb{F}_n$ est complètement définie par un mot binaire de longueur 2^n. ♦

Une fonction booléenne est définie par sa table de vérité, cette table possède 2^n lignes pour représenter toutes les valeurs que peuvent prendre les n paramètres, chaque ligne indiquant la valeur 0 ou 1 pour la fonction. D'où la propriété.

Exemple 23.6
Soit la fonction f telle que $f(1, 0) = f(1, 1) = 1$; que l'on aurait aussi pu décrire par $f(1, x) = 1$.
Sa table de vérité est donc :

x	y	f(x,y)
0	0	0
0	1	0
1	0	1
1	1	1

Le mot binaire associé à f sera 0011. †

Propriété 23.6 $(\mathbb{F}_n, \hat{+}, \hat{.}, \bar{}, 0^n, 1^n)$ est une algèbre de Boole.
Soit $f_1 = x_1x_2 \dots x_n$ et $f_2 = y_1y_2 \dots y_n$

1. 0^n désigne le mot binaire de longueur n ne contenant que des 0
2. 1^n désigne le mot binaire de longueur n ne contenant que des 1
3. $f_1 \hat{+} f_2 = z_1z_2 \dots z_n$ avec $z_i = x_i + y_i - x_iy_i$ calculé dans $\mathbb{N}$
4. $f_1 \hat{.} f_2 = z_1z_2 \dots z_n$ avec $z_i = x_iy_i$ calculé dans $\mathbb{N}$
5. $\overline{f_1} = z_1z_2 \dots z_n$ avec $z_i = 1 - x_i$ calculé dans $\mathbb{N}$

♦

Définition 23.2.3 (minterme, maxterme)

1. Dans $\mathbb{F}_n$ on appelle *minterme* la fonction qui vaut 0 partout sauf en une unique valeur.
2. Dans $\mathbb{F}_n$ on appelle *maxterme* la fonction qui vaut 1 partout sauf pour une unique valeur.

◄

Notation 23.1 On considère l'ensemble $\mathbb{F}_n$. On note m_i le ième minterme dont le 0 est en ième position dans le mot binaire associé. On notera M_j le jème maxterme dont le 1 est en jème position dans le mot binaire associé. La numérotation se fait de la gauche vers la droite. Ainsi $m_1 = 01111\dots1$; $M_2 = 010\dots0$.

Attention L'ordre des lignes doit être uniforme pour toutes les fonctions de $\mathbb{F}_n$. On utilisera l'ordre lexicographique de $\mathcal{B}^n$.

Exemple 23.7
Dans l'exemple 23.6 $\mathbb{F}_2$ admet
• les mintermes $1000, 0100, 0010, 0001$
• les maxtermes $0111, 1011, 1101, 1110$. †

Remarque 23.2.2
En toute généralité, il faudrait rajouter un indice supplémentaire pour indiquer que l'on travaille dans $\mathbb{F}_n$, il faudrait par exemple écrire $m_i^{(n)}$ (respectivement $M_i^{(n)}$) le ième minterme (respectivement le maxterme) de $\mathbb{F}_n$. Dans la suite, nous fixerons préalablement l'ensemble $\mathbb{F}_n$ afin d'alléger la notation des mintermes, et maxtermes. ○

Propriété 23.7 Dans $\mathbb{F}_n$, $m_i = \overline{M_i}$ ♦

$| m_i | = | M_i |$; la ième position dans m_i vaut 0 ; dans M_i elle vaut 1. Toutes les autres positions dans m_i sont à 1, tandis que dans M_i elles sont à 0.

Propriété 23.8 Dans $\mathbb{F}_n$ il y a 2^n mintermes (respectivement maxtermes) ♦

Une fonction de $\mathbb{F}_n$ est définie par un mot binaire de longueur 2^n, un minterme (resp. maxterme) est un mot binaire contenant une unique valeur à 0 (resp. 1). Il y a donc 2^n façons différentes de placer le 0 (resp. le 1).

Propriété 23.9 Dans $\mathbb{F}_n$ il existe 2^{2^n} fonctions booléennes différentes. ♦

À chaque fonction de $\mathbb{F}_n$ on peut faire correspondre un mot binaire de longueur 2^n. Il y a 2^k mots binaires de longueur k ; il y a donc ici 2^{2^n} mots binaires différents. D'où 2^{2^n} fonctions différentes dans $\mathbb{F}_n$.

Propriété 23.10 Si f est une fonction booléenne de $\mathbb{F}_k$ alors, $\forall n \geq k$, f est une fonction de $\mathbb{F}_n$. ♦

Pour établir cette propriété, il suffit d'établir que si $f \in \mathbb{F}_k$, alors $f \in \mathbb{F}_{k+1}$. En utilisant la propriété de Shannon, on sait que $f(x_1, x_2, \ldots, x_p) = x_1.f_1(x_2, \ldots, x_p) + \overline{x_1}.f_2(x_2, \ldots, x_p)$ si on prend $f_1 = f_2$ on obtient :
$f(x_1, x_2, \ldots, x_p) = (x_1 + \overline{x_1}).f_1(x_2, \ldots, x_p) = f_1(x_2, \ldots, x_p)$
Ceci nous permet, étant donnée une fonction à k paramètres de l'étendre à k+1 paramètres.

Exemple 23.8
Soit f définie par la table de vérité suivante, et que l'on pourrait définir par $f(x, 0) = 1$:

x	y	f(x,y)
0	0	1
0	1	0
1	0	1
1	1	0

Pour passer à trois paramètres on recopie deux fois la table (une fois quand le nouveau paramètre z=0, une fois quand z=1). Afin de préserver l'ordre lexicographique, il faudra ensuite réorganiser les lignes.

1. Première étape duplication et ajout de la colonne :

x	y	z	f(x,y,z)
0	0	**0**	1
0	1	**0**	0
1	0	**0**	1
1	1	**0**	0
0	0	**1**	1
0	1	**1**	0
1	0	**1**	1
1	1	**1**	0

2. Seconde étape réordonnancement :

x	y	z	f(x,y,z)
0	0	**0**	1
0	0	**1**	1
0	1	**0**	0
0	1	**1**	0
1	0	**0**	1
1	0	**1**	1
1	1	**0**	0
1	1	**1**	0

Bien entendu, il est possible d'aller plus vide en dupliquant chaque ligne, et en rajoutant une troisième colonne dans laquelle on alterne les valeurs **0** et **1**.
Qui est, dans $\mathbb{F}_3$ la fonction $f(x, 0, z) = 1$ †

Propriété 23.11

1. Toute fonction booléenne de $\mathbb{F}_n$ peut s'exprimer comme une somme de mintermes.
2. Toute fonction booléenne de $\mathbb{F}_n$ peut s'exprimer comme un produit de maxtermes.

⧫

Pour établir cette propriété, il suffit de passer à l'expression de la fonction sous la forme d'un mot binaire et utiliser les opérations bit à bit tels que présenter dans la propriété 23.6.

Remarque 23.2.3
Un minterme (respectivement un maxterme) est un élément de $\mathbb{F}_n$, que l'on peut donc définir en fonction de n paramètres appartenant à $\mathcal{B}$, et utilisant les opérations $+$, $.$, $^-$définie dans $\mathcal{B}$.
Soit le minterme 01000000 c'est la fonction $m_2(x, y, z)$ telle que $m_2(0, 0, 1) = 1$ que l'on peut écrire $(\bar{x}.\bar{y}.z)$
Soit le maxterme 11101111 c'est la fonction $M_4(x, y, z) = \overline{m_4(x, y, z)} = \overline{\bar{x}yz} = (x + \bar{y} + \bar{z})$ ∘

Retour sur les fonctions booléennes en général

Propriété 23.12 (formes canoniques)

1. Toute fonction booléenne peut s'exprimer comme une somme de produits de variables. C'est l'écriture sous forme *canonique disjonctive*.
2. Toute fonction booléenne peut s'exprimer comme un produit de sommes de variables. C'est l'écriture *canonique conjonctive*.

⧫

Les fonctions dans $\mathbb{F}_n$ étant un cas particulier de fonctions booléennes, on étend la propriété de $\mathbb{F}_n$ à l'ensemble des fonctions booléennes. Soit une fonction booléenne f, il existe un ensemble $\mathbb{F}_k$ auquel elle appartient, de là on exprime f comme une somme (respectivement un produit) de mintermes (respectivement maxtermes). Chaque minterme (respectivement maxterme) est une fonction booléenne que l'on peut traduire aisément sous forme de produit (respectivement somme) de variables. Il n'y a plus qu'à faire le lien.

Exemple 23.9
Considérons la fonction booléenne $f \in \mathbb{F}_3$ définie par $f(1, y, z) = 1 = f(x, y, 0)$. Sa table de vérité possède $2^3 = 8$ lignes et 4 colonnes.

i	x	y	z	$f(x, y, z)$
1	0	0	0	1
2	0	0	1	0
3	0	1	0	1
4	0	1	1	0
5	1	0	0	1
6	1	0	1	1
7	1	1	0	1
8	1	1	1	1

Elle correspond aux mintermes $m_1, m_3, m_5, m_6, m_7, m_8$ et aux maxtermes M_2, M_4
L'idée pour construire un minterme est d'utiliser les noms des paramètres et mettre un $\bar{}$ sur la lettre si sa valeur est 0, de combiner cette expression avec des produits, d'où pour m_1 on trouve $(\bar{x}.\bar{y}.\bar{z})$
L'idée pour construire un maxterme est d'utiliser les noms des paramètres, de mettre un $\bar{}$ sur la lettre si la valeur est 1, de combiner cette expression avec des sommes, d'où pour M_2 on trouve $(x + y + \bar{z})$

1. La forme canonique disjonctive est donc

$$f(x, y, z) = (\bar{x}.\bar{y}.\bar{z}) + (\bar{x}.y.\bar{z}) + (x.\bar{y}.\bar{z}) + (x.\bar{y}.z) + (x.y.\bar{z}) + (x.y.z)$$

2. La forme canonique conjonctive est donc

$$f(x, y, z) = (x + y + \bar{z}).(x + \bar{y} + \bar{z})$$

†

Remarque 23.2.4
Les formes canoniques, ne sont pas les formes les plus petites pour exprimer les formes conjonctives ou disjonctives.
Si on reprend l'exemple 23.9, il est possible d'exprimer de manière plus concise, la fonction $f(x, y, z)$ comme $(x + \bar{z})$. Ou, si l'on veut faire apparaître la variable y dans une somme $xy + x\bar{y} + y\bar{z} + \bar{y}\bar{z}$ ou dans un produit $(x + y + \bar{z}).(x + \bar{y} + \bar{z})$ ∘

Définition 23.2.4 (fonction duale) La *duale* de f, notée $\widehat{f}$ est :

$$\widehat{f}(x_1, \ldots, x_p) = \overline{f(\overline{x_1}, \ldots, \overline{x_p})}$$

◀

Propriété 23.13 (dualité)

1. La duale d'une somme est le produit des duales de ses termes.
2. La duale d'un produit est la somme des duales de ses termes.
3. La duale d'une duale est la fonction initiale.

⧫

Exemple 23.10
Soit la fonction de l'exemple 23.9 $f(x, y, z) = x + \bar{z}$

- $\overline{f(x, y, z)} = \overline{x + \bar{z}} = \bar{x}z$
- $f(\bar{x}, \bar{y}, \bar{z}) = \bar{x} + \bar{\bar{z}} = \bar{x} + z$
- $\hat{f}(x, y, z) = \bar{\bar{x}}\bar{z} = x\bar{z}$

†

Partie II : Introduction à la logique propositionnelle

Dans cette partie, nous abordons la logique et plus particulièrement la logique **propositionnelle** dite aussi booléenne ou encore calcul des énoncés. Sans logique pas de raisonnement, pas de langage de programmation, pas de vérification de programmes, pas de démonstration. La logique en informatique va nous permettre de distinguer la notion de syntaxe (manipulation de symboles) de la notion de sémantique (sens, signification, interprétation). Cette partie est construite à partir des ressources suivantes [1, 2, 4, 6]. Dans le livre de Smullyan [5] vous trouverez des applications ludiques tels que puzzles et énigmes.

Selon le dictionnaire Larousse, la logique est la science du raisonnement en lui-même, abstraction faite de la matière à laquelle il s'applique et de tous processus psychologiques. La logique constitue une langue, c'est-à-dire un système de symboles et de variables liés par des opérateurs qui déterminent la structure interne des propositions et des relations entre les propositions. Historiquement, la logique remonterait à Aristote (IV^e siècle avant J.C.) qui posa les bases du syllogisme, qui ne sera formalisé qu'au cours du moyen-âge. Au XIX^e siècle la logique devient mathématique suite aux travaux de Bolzano, Boole et De Morgan, pour devenir formelle avec Frege, Russel et Wittgenstein.

Dans l'Antiquité, l'objet de la logique était le discours, afin d'étudier la validité de l'argumentation présentée :
« *Tout le monde le sait, les politiciens sont véreux. Je suis issu de la société civile, en me confiant ce mandat ...* ». Cet extrait fictif, n'est pas sans rappeler certains propos tenus ici ou là et ouvre sur une argumentation oiseuse.
« Tout le monde le sait » est un élément du discours visant à englober l'auditoire dans une croyance, faisant (ac)croire que « [tous] les politiciens sont véreux » est un énoncé **vrai** – l'énoncé est **faux** pour plusieurs raisons [2] et a pour conséquence que la suite n'a aucun intérêt.
En supposant qu'effectivement « [tous] les politiciens sont véreux » soit **vrai**. L'expression « société civile » à la mode est une opposition de la « société politique », l'auteur (fictif) exprime qu'il n'est pas *politicien* et incite l'auditoire à penser qu'en conséquence « je ne suis pas véreux » est un énoncé **vrai** – là encore, la logique met en évidence un raisonnement **faux**, le fait de ne pas être politicien n'a pas d'impact sur le fait d'être (ou pas) véreux, il n'y a pas d'équivalence, le seul raisonnement correct est « je ne suis pas véreux, donc je ne suis pas un politicien ». qui est la contraposée de l'assertion.
Admettons, cependant que l'auteur soit effectivement une personne de qualité irréprochable. Quand il brigue un mandat (politique), il indique le souhait de rentrer dans la société politique, devenant ainsi un politicien, d'où il s'ensuivra, en accord avec le début de l'extrait, *véreux ...*
Conclusion l'extrait pourrait être réduit à « Je ne suis pas politicien et je souhaite devenir véreux, en m'élisant ... »

23.3 Langage

La logique est un *langage* qui contient des règles et des signes. Ce langage est constitué d'un ensemble de symboles et de variables liés par des « connecteurs » qui déterminent la structure interne des propositions et les relations entre ces différentes propositions.

On considère un alphabet $\mathcal{A}$ constitué de 3 sous-ensembles disjoints :

2. Premier argument JE, auditeur particulier, ne le sais pas ; second argument Mme Unetelle, politicienne notoire n'est pas véreuse.

1. $\mathcal{P} = \{p, q, r, \ldots\}$ un ensemble fini ou dénombrable de variables propositionnelles, éventuellement indicées.
2. Ponct = { (,) }
3. Conn = $\{\neg, \wedge, \vee, \supset, \equiv\}$, ces connecteurs se lisent respectivement `non`, `et`, `ou`, `implique`, `équivalent à`. Nous reviendrons ultérieurement (section 23.4) sur la signification à accorder à ces différents connecteurs, et aux différences notables avec le langage naturel.

à partir de cet alphabet, on construit des mots qui appartiennent à $\mathcal{A}^*$. Sur ces mots, on définit un sous ensemble noté $\mathcal{F}$ constituant les formules propositionnelles.
Par « mot » on entend ici simplement une succession de symboles appartenant à l'alphabet, *sans se préoccuper du sens* que pourrait avoir ce mot. La définition 23.3.1 a pour objectif de définir les mots qui vont nous intéresser, c'est-à-dire ceux auxquels on pourra donner un sens, une signification.

Définition 23.3.1 L'ensemble des formules propositionnelles $\mathcal{F}$ construites sur $\mathcal{P}$ est le plus petit ensemble tel que :

- Si $F \in \mathcal{F}$ alors $\neg F \in \mathcal{F}$
- Si F et G sont des formules alors $(F \wedge G)$, $(F \vee G)$, $(F \supset G)$ et $(F \equiv G)$ appartiennent à $\mathcal{F}$.

◀

De cette définition, on peut tirer plusieurs remarques. La première est que l'on a $\mathcal{P} \subseteq \mathcal{F}$, la seconde que $\mathcal{F}$ est bien définie, car $\mathcal{A}^*$ vérifie la définition 23.3.1, que $\mathcal{P}$ appartient à l'intersection de tous les ensembles vérifiant cette définition.

Il est aussi possible de définir les formules propositionnelles par induction :

Définition 23.3.2

1. $\mathcal{F}_0 = \mathcal{P}$
2. $\mathcal{F}_{n+1} = \mathcal{F}_n \cup \{\neg F \,/\, F \in \mathcal{F}_n\} \cup \{(F \bowtie G) \,/\, F, G \in \mathcal{F}_n, \bowtie \in \{\wedge, \vee, \supset, \equiv\}\}$

◀

De la définition 23.3.2 on constate que la suite $(\mathcal{F}_n)_{n \in \mathbb{N}}$ est croissante. Les définitions 23.3.1 et 23.3.2 définissent le même objet.

La proposition suivante établit l'égalité :

Propriété 23.14

$$\mathcal{F} = \bigcup_n \mathcal{F}_n$$

⧫

Pour faire la démonstration on prouve la double inclusion. Dans un sens, il faudra faire une preuve par induction, dans l'autre sens il suffira d'établir que l'union des $\mathcal{F}_n$ vérifie la définition 23.3.1.

Les **lois** sur lesquelles reposent le calcul des énoncés sont :

- Le principe d'identité : $(f \equiv f)$ « une formule est identique à elle-même »

- Le principe de non contradiction : $\neg(f \wedge \neg f)$ « on ne peut avoir une formule et son contraire »
- Le principe du tiers exclus : $(f \vee \neg f)$ « il n'y a rien d'autre qu'une formule ou son contraire »
- Le principe de la double négation : $(\neg\neg f \equiv f)$ « le contraire du contraire d'une formule est la formule »

Ces principes (ou lois) sont à comprendre comme des formules logiques toujours vraies.

Définition 23.3.3 On dira qu'une formule G est une sous-formule d'une formule F si elle vérifie l'un des points suivants :

1. F = G ;
2. F = $\neg$F$_1$ et G une sous-formule de F$_1$;
3. F = $(F_1 \bowtie F_2)$, avec $\bowtie \in \{\vee, \wedge, \supset, \equiv\}$ si G est une sous-formule de F_1 ou de F_2.

◀

L'intérêt de la notion de sous-formules est de pouvoir décomposer l'expression en sous-expressions plus simples à calculer, puis recombiner les différents résultats intermédiaires pour obtenir le résultat final, tout comme pour calculer, dans $\mathbb{Z}$, $(((5 \times 47) + (6 \times 23)) \times ((12 + ((8 - 3) \times 7)) - 42)) = \ldots$

Exemple 23.11
Ainsi soit $F = ((\neg p \vee q) \wedge r)$; p, $\neg p$, $(\neg p \vee q)$, r et F sont des sous-formules de F, mais pas $(q \wedge r)$. †

Lorsque l'on désire étudier une formule logique, on peut le faire selon deux approches : (1) la *théorie des modèles* et (2) la *théorie de la démonstration*. La première a pour but d'établir un mécanisme sémantique d'évaluation des formules, il s'agit d'une formalisation de la notion intuitive que nous avons de la vérité ou de la fausseté d'une phrase en fonction de ses constituants. La seconde est un mécanisme purement syntaxique d'évaluation. Il s'agit, dans ce cadre, d'appliquer des règles mécaniques sans s'intéresser au sens véhiculé par les composants. Les règles sont appelées, *règles d'inférence* et portent sur des *jugements* que l'on souhaite prouver mécaniquement – dans ce cadre un jugement porte sur la véracité d'une formule.

23.4 Sémantique dans le calcul des propositions

Le rôle de la *syntaxe* est de différencier une formule d'un mot quelconque de $\mathcal{A}^*$. La tâche dévolue à la *sémantique* est d'attribuer une *signification* aux énoncés, plus particulièrement une valeur de vérité.

Une proposition est soit vraie, soit fausse. On peut alors lui attribuer un domaine sémantique que l'on notera { **vrai**, **faux** } – ou encore $\{0, 1\}$, 0 correspondant à **faux**, 1 à **vrai**, lorsque l'on se place dans le cadre de l'algèbre de Boole (et des fonctions booléennes). Interpréter une formule revient à lui associer une valeur de vérité. Une sémantique est une loi compositionnelle qui dépend de ses constituants. Les connecteurs logiques sont considérés comme vérifonctionnels puisqu'une table de vérité définit complètement leur comportement. Ci-après, nous donnons la table de vérité associée à chaque connecteur (**À savoir par cœur**) :

p	q	$\neg p$	$(p \wedge q)$	$(p \vee q)$	$(p \supset q)$	$(p \equiv q)$
0	0	1	0	0	1	1
0	1	1	0	1	1	0
1	0	0	0	1	0	0
1	1	0	1	1	1	1

Avant d'aller plus loin, il semble nécessaire de comparer la sémantique attribuée aux connecteurs logiques et leurs significations en langage naturel (français, anglais, ...).

Et logique vs et français
Le `et` logique est commutatif, ce qui n'est pas le cas en français puisqu'il peut y avoir un lien de causalité entre les deux propositions. On pourra, pour s'en assurer, comparer les deux phrases suivantes : « Il prit peur et le tua », « Il le tua et prit peur ». Dans la même veine on pourra étudier les deux phrases : « Elle eut un enfant et se maria », « Elle se maria et eut un enfant ».

Ou logique vs ou français
Très souvent le `ou` français possède une connotation d'exclusivité, c'est-à-dire que les deux propositions ne peuvent être vraies simultanément, alors que le `ou` logique est par nature inclusif. à titre d'exemple, considérez les phrases : « peste ou choléra », « fromage ou dessert », « le pistolet ou la corde », « vous montez ou vous descendez ? »...

Implication logique vs implication française
L'implication française qui se traduit par l'usage de la syntaxe « **si** hypothèse **alors** conclusion » indique une relation de causalité entre hypothèse et conclusion. L'implication logique [3] $(p \supset q)$ est en fait une notation commode pour exprimer l'alternative [4] $(q \vee \neg p)$ qui indique qu'aucune relation n'est exigée entre les deux propositions (ou formules logiques) représentant l'hypothèse (p) et la conclusion (q).

Exercice 23.4.1
Construisez les tables de vérité des formules suivantes $(p \supset q)$, $(\neg p \vee q)$, $\neg(\neg p \vee q)$, $(p \wedge \neg q)$, $(q \supset p)$ et $(\neg q \supset \neg p)$.
On s'attend à ce que les colonnes 1, 2 et 6 soient identiques. Que les colonnes 3 et 4 soient similaires et en opposition avec la colonne 1. Enfin que la colonne 5 soit différente de toutes les autres.

p	q	$(p \supset q)$	$(\neg p \vee q)$	$\neg(\neg p \vee q)$	$(p \wedge \neg q)$	$(q \supset p)$	$(\neg q \supset \neg p)$
0	0						
1	0						
0	1						
1	1						

23.5 formules et fonctions booléennes

Avant de commencer l'étude des interprétations que l'on peut associer à une formule, il semble utile de regarder plus précisément, quelles sont les fonctions à 2 variables que l'on peut définir. Comme chaque variable peut prendre 2 valeurs, il y a donc 4 couples distincts de

3. On dit aussi « implication matérielle ».
4. Pour s'assurer de la correction de cette expression, il suffit de construire les tables de vérité associées à chacune des expressions et de vérifier que les résultats sont identiques.

valeurs, à chaque couple on peut associer (on dit aussi assigner) 2 réponses (0 ou 1), il y a ainsi 16 fonctions à deux variables possibles qui sont résumées dans la table de la figure 23.1. Nous avons vu qu'il existait 2^{2^n} fonctions booléennes à n variables.

p	q	f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8	f_9	f_{10}	f_{11}	f_{12}	f_{13}	f_{14}	f_{15}	f_{16}
0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
0	1	0	1	0	0	1	1	0	1	0	1	0	0	1	1	0	1
1	0	0	0	1	0	1	0	1	1	0	0	1	0	1	0	1	1
1	1	0	0	0	1	0	1	1	1	0	0	0	1	0	1	1	1

FIGURE 23.1 – 16 fonctions à 2 variables

à partir de la figure 23.1, on reconnaît les identités suivantes (le lecteur n'a qu'à trouver les formes polynomiales de chaque fonction) : $f_1(p,q) = 0$ (fonction uniformément fausse), $f_2(p,q) = (\neg p \wedge q)$, $f_3(p,q) = (p \wedge \neg q)$, $f_4(p,q) = (p \wedge q)$, $f_5(p,q) = (p \equiv \neg q) = (\neg p \equiv q) = \neg(p \equiv q)$, $f_6(p,q) = q$, $f_7(p,q) = p$, $f_8(p,q) = (p \vee q)$, $f_9(p,q) = (p \downarrow q)$, $f_{10}(p,q) = \neg p$, $f_{11}(p,q) = \neg q$, $f_{12}(p,q) = (p \equiv q)$, $f_{13}(p,q) = (p \uparrow q)$, $f_{14}(p,q) = (p \supset q)$, $f_{15}(p,q) = (q \supset p)$, $f_{16} = 1$. La fonction f_1 est appelée **contradiction** dont l'expression type est $(p \wedge \neg p)$, tandis que la fonction f_{16} est appelée **tautologie** dont l'expression type est $(p \vee \neg p)$. Il en résulte que la négation d'une contradiction est une tautologie et réciproquement.

23.5.1 formules et interprétation

Dans cette partie, nous allons voir comment faire le lien entre une formule de la logique propositionnelle et une fonction booléenne.

Définition 23.5.1 Un fonction d'interprétation [5] (ou interprétation) est une application **i** qui à toute variable propositionnelle p associe une valeur de vérité. ◀

On étend facilement cette notion aux formules, par le biais des tables de vérité. Cette extension noté **I** est aussi une interprétation.

Définition 23.5.2 Une interprétation qui rend une formule F vraie est appelée un *modèle* de F. ◀

Exemple 23.12
Prenons la formule $F = (p \vee q)$, et considérons l'interprétation **I** vérifiant, $\mathbf{I}[\![p]\!] = 1, \mathbf{I}[\![q]\!] = 0$, c'est un modèle de F, puisque $\mathbf{I}[\![F]\!] = 1$.
Ce n'est pas la seule possible, combien y-a-t-il d'interprétations de F qui soit un modèle de F ?
†

Exercice 23.5.1 (∗)
Étant donnée une formule F contenant k variables propositionnelles, combien existe-t-il d'interprétations différentes de F ?

Donnons une autre façon de faire le lien entre interprétation d'une formule et fonction boolénne.

Définition 23.5.3 Soit F une formule, on va considérer l'interprétation de F, $\mathbf{I}[\![F]\!]$ de manière inductive :

5. On parle parfois de fonction de valuation.

- Si F $\in \mathcal{P}$, $\mathbf{I}[\![F]\!] = \mathbf{i}(\mathrm{F})$;
- Si F = $\neg$G, $\mathbf{I}[\![F]\!] = 1$ - $\mathbf{I}[\![G]\!]$
- Si F = $(F_1 \bowtie F_2)$, $\mathbf{I}[\![F]\!] = f_{\bowtie}(\mathbf{I}[\![F_1]\!], \mathbf{I}[\![F_2]\!])$, avec $\bowtie \in \{\vee, \wedge, \supset, \equiv\}$ et $f_{\bowtie}$ désigne la fonction booléenne correspondant au connecteur logique.

◀

Exemple 23.13
Par exemple la fonction $f_\vee$ est la fonction f_8 de la figure 23.1 et correspond à la définition de « + » utilisée dans $\mathbb{B}$. La fonction $f_\wedge$ est la fonction f_4 de la figure 23.1 et correspond à la définition de « . » utilisée dans $\mathbb{B}$ †

Définition 23.5.4 Un *littéral* est soit une variable propositionnelle, soit la négation d'une variable propositionnelle. En d'autre terme, si $p \in \mathcal{P}$, alors p et $\neg p$ sont des littéraux, le premier sera dit *positif* et le second *négatif*. ◀

Définition 23.5.5 (formes normales) Une *forme normale conjonctive* (fnc) est soit un littéral, soit une conjonction de formules écrites comme disjonctions de littéraux.
Une *forme normale disjonctive* (fnd) est soit un littéral, soit une disjonction de formules écrites comme conjonctions de littéraux. ◀

Définition 23.5.6 (forme clausale) Une *clause* est une disjonction de littéraux, que l'on peut représenter comme un ensemble des littéraux qui la compose. ◀

Définition 23.5.7 (subsomption) Une clause c_0 subsume une autre clause c_1 si $c_0 \subseteq c_1$ ◀

Propriété 23.15 Algorithme de réduction

1. Si une clause contient un littéral et son opposé, on peut supprimer la clause
2. Dans une clause, on supprime les multiples occurrences d'un littéral
3. Si une clause c contient une clause c', on supprime la clause c

⧫

Soit $\mathcal{C} = \{c_1, \dots c_k\}$ un ensemble de clauses, une clause $c_i \in \mathcal{C}$ est de la forme $\{l_{1_i}, \dots l_{k_i}\}$. Un modèle pour un ensemble de clauses est un modèle pour chacune des clauses constituantes. Par conséquent, un modèle d'un ensemble de clauses $\mathcal{C}_0$ est un modèle pour tout sous-ensemble de $\mathcal{C}_0$.

1. Une clause contenant un littéral et son opposé est une tautologie (vraie quelque soit l'interprétation choisie).
2. Une clause étant un disjonction de littéraux, et le connecteur $\vee$ étant absorbant, supprimer au sein d'une clause les occurrences multiples d'un littéral n'influe pas sur le modèle.
3. Soient $c_i, c_j \in \mathcal{C}$ telles que $c_j \subseteq c_i$, et soit I un modèle de c_j cela signifie qu'il existe $q_j \in \{1_j, \dots k_j\}$ tel que $\mathbf{I}[\![l_{q_j}]\!] = 1$, comme $l_{q_j} \in c_i$, I est un modèle de c_i.

Ces 3 points démontrent que tout modèle de l'ensemble réduit est un modèle de l'ensemble initial.

Exemple 23.14
Soit $F = ((A \wedge C) \vee (A \wedge \neg B \wedge \neg C) \vee (\neg A \wedge B \wedge \neg C))$, sous forme disjonctive. On peut la mettre sous forme conjonctive en utilisant la distributivité de $\vee$ par rapport à $\wedge$. Ce qui va nous donner des clauses de taille 3, il y en a $2 \times 3 \times 3 = 18$

$$\begin{array}{lc|l}
((A \vee A \vee \neg A) & \wedge & (c_1) \\
(A \vee A \vee B) & \wedge & (c_2) \\
(A \vee A \vee \neg C) & \wedge & (c_3) \\
(A \vee \neg B \vee \neg A) & \wedge & (c_4) \\
(A \vee \neg B \vee B) & \wedge & (c_5) \\
(A \vee \neg B \vee \neg C) & \wedge & (c_6) \\
(A \vee \neg C \vee \neg A) & \wedge & (c_7) \\
(A \vee \neg C \vee B) & \wedge & (c_8) \\
(A \vee \neg C \vee \neg C) & \wedge & (c_9) \\
(C \vee A \vee \neg A) & \wedge & (c_{10}) \\
(C \vee A \vee B) & \wedge & (c_{11}) \\
(C \vee A \vee \neg C) & \wedge & (c_{12}) \\
(C \vee \neg B \vee \neg A) & \wedge & (c_{13}) \\
(C \vee \neg B \vee B) & \wedge & (c_{14}) \\
(C \vee \neg B \vee \neg C) & \wedge & (c_{15}) \\
(C \vee \neg C \vee \neg A) & \wedge & (c_{16}) \\
(C \vee \neg C \vee B) & \wedge & (c_{17}) \\
(C \vee \neg C \vee \neg C)) & & (c_{18})
\end{array}$$

Les clauses $c_1, c_4, c_5, c_7, c_{10}, c_{12}, c_{14}, c_{15}, c_{16}, c_{17}, c_{18}$ sont des tautologies
Les clauses c_2, c_3, c_9 contiennent plusieurs occurrences d'un même littéral.

$$\begin{array}{lc|l}
((A \vee B) & \wedge & (c_2) \\
(A \vee \neg C) & \wedge & (c_3) \\
(A \vee \neg B \vee \neg C) & \wedge & (c_6) \\
(A \vee \neg C \vee B) & \wedge & (c_8) \\
(A \vee \neg C) & \wedge & (c_9) \\
(C \vee A \vee B) & \wedge & (c_{11}) \\
(C \vee \neg B \vee \neg A)) & & (c_{13})
\end{array}$$

Les clauses c_3 et c_9 sont identiques, supprimer les multiples occurrences d'une clause, ne change pas le modèle.
La clause c_2 subsume la clause c_{11}. La clause c_3 subsume la clause c_6, la clause c_3 subsume la clause c_8.

$$\begin{array}{lc|l}
((A \vee B) & \wedge & (c_2) \\
(A \vee \neg C) & \wedge & (c_3) \\
(C \vee \neg B \vee \neg A)) & & (c_{13})
\end{array}$$

†

Exercice 23.5.2 ($*$)
Trouver la forme normale conjonctive simplifiée de la formule $((p \supset (q \supset r)) \supset ((p \wedge s) \supset r))$.

Définition 23.5.8 Une formule sera dite sémantiquement consistante (ou plus simplement consistante) si elle admet un modèle[6]. Une formule qui n'est pas consistante est dite inconsistante[7]. Une formule est dite valide[8] si elle est toujours vraie. ◀

Dire qu'un ensemble $\mathcal{E}$ de formules est cohérent (ou consistant), c'est dire que tous les élements de $\mathcal{E}$ admettent le même modèle (ou encore qu'il existe un modèle pour la conjonction des formules de $\mathcal{E}$). Enfin, dire que F est conséquence logique d'un ensemble de formules $\mathcal{E}$, c'est établir que $\mathcal{E} \cup \{\neg F\}$ est incohérent (ou inconsistant). Cette dernière approche est à la base de la *déduction* qui consiste à déterminer si une formule F est conséquence logique d'un ensemble de formules $\{F_1, F_2, \ldots, F_k\}$.

Les questions que l'on peut vouloir résoudre sont :

1. Déterminer si un mot $m \in \mathcal{A}^*$ est une formule. Il s'agit d'un problème trivial.
2. Déterminer si une formule $F \in \mathcal{F}$ est
   - consistante, (il existe un modèle) ;
   - valide, (toute interprétation est un modèle) ;
   - inconsistante, (il n'existe pas de modèle).

   sont des problèmes qui sont soient difficiles, soient fastidieux puisqu'il y a un nombre exponentiel de cas à tester.

Remarque 23.5.1
Du fait du lien existant entre formules logiques propositionnelles et fonction booléennes, les simplifications des fonctions booléennes sont directement applicable sur les formules logiques. ○

Exemple 23.15
La formule logique $(p \vee \neg p)$ a son pendant en fonction booléenne $(x + \bar{x})$ †

23.5.2 Puissance d'expression

Dans cette partie nous allons voir, que les cinq connecteurs ne sont pas nécessaires pour exprimer une formule dans le langage propositionnel. L'idée est de pouvoir simplifier les expressions, en minimisant le nombre de connecteurs différents.
Pour s'assurer qu'un ensemble de connecteurs forme une base, on se restreindra à la comparaison des tables de vérité de chaque expression. Dans la suite nous utiliserons le signe $=$[9] entre deux expressions, pour indiquer qu'elles ont même table de vérité.

Définition 23.5.9 (adéquat et base) On dira qu'un ensemble de connecteurs est *adéquat* si toute formule logique peut s'exprimer en n'utilisant que les connecteurs de cet ensemble.
On dira qu'un ensemble de connecteurs forme une *base* pour la logique propositionnelle, si c'est un ensemble adéquat *minimal*. ◀

Remarque 23.5.2
L'ensemble des connecteurs $\{\neg, \vee, \wedge, \supset, \equiv\}$ est adéquat, puisque, par définition les formules logiques sont définies à partir de ces 5 connecteurs. ○

6. satisfaisable
7. insatisfaisable
8. tautologie.
9. Le signe $=$ appartient au méta-langage et non au langage propositionnel. Dire que $F = G$ c'est dire que $(F \equiv G)$ est une tautologie.

Exemple 23.16
Montrez que l'ensemble de connecteurs $\{\neg, \vee, \wedge, \supset\}$ est adéquat. †

Schéma de solution 6
L'idée est assez simple, il faut trouver le moyen d'exprimer le connecteur $\equiv$ en utilisant une combinaison des autres connecteurs. En l'occurrence, on va montrer que $(p \equiv q)$ et $((p \supset q) \wedge (q \supset p))$ sont similaires.

p	q	$(p \equiv q)$	$(p \supset q)$	$(q \supset p)$	$((p \supset q) \wedge (q \supset p))$
0	0	1	1	1	1
0	1	0	1	0	0
1	0	0	0	1	0
1	1	1	1	1	1

Pour chaque ligne de la table, il y a correspondance, ce qui veut dire que une interprétation de $(p \equiv q)$ est un modèle de $(p \equiv q)$ si et seulement si c'est un modèle de $((p \supset q) \wedge (q \supset p))$. Toute formule $(F \equiv G)$ peut être remplacée par la formule $((F \supset G) \wedge (G \supset F))$.

D'autres simplifications sont possibles et seront l'objet d'exercices en TD.

Algorithme de normalisation

Il existe deux méthodes simples pour mettre une formule quelconque sous forme normale, la première passe par une suite d'équivalence (au sens du signe $=$) en remplaçant le membre gauche par le membre droit. Cet méthode peut se résumer par les règles suivantes :

Suppression de $\equiv$
$(F \equiv G) = ((F \supset G) \wedge (G \supset F))$
Suppression de $\supset$
$(F \supset G) = (\neg F \vee G)$
Déplacement de $\neg$ àl'intérieur des formules
$\neg(F \vee G) = (\neg F \wedge \neg G)$
$\neg(F \wedge G) = (\neg F \vee \neg G)$
$\neg\neg F = F$
Distributivité des opérateurs $\vee$ et $\wedge$
$(F \wedge (G \vee H)) = ((F \wedge G) \vee (F \wedge H))$
$(F \vee (G \wedge H)) = ((F \vee G) \wedge (F \vee H))$

La seconde méthode consiste à déterminer la fonction booléenne associée, puis à construire une formule sous forme normale. Les étapes peuvent se résumer de la manière suivante :

(1) On détermine les interprétations I telles que $\mathbf{I}[F] = 1$
(2) à chaque I_k telle que $I_k[F] = 1$ est associée une formule g_k
qui est une conjonction de littéraux (minterme)
(3) La formule g est finalement obtenue par la disjonction des g_k.

(1) On détermine les interprétations I telles que $\mathbf{I}[F] = 0$
(2) à chaque I_k telle que $I_k[F] = 0$ est associée une formule h_k
qui est une disjonction de littéraux (maxterme)
(3) La formule h est finalement obtenue par la conjonction des h_k.

Afin de clarifier ces méthodes considérons l'exemple suivant de normalisation.

Exemple 23.17
Soit à normaliser la formule $\neg(p \equiv (q \supset r))$. Par la première méthode on obtient :
$\neg((p \supset (q \supset r)) \wedge ((q \supset r) \supset p))$

$\neg((\neg p \vee (\neg q \vee r)) \wedge (\neg(\neg q \vee r) \vee p))$
$(\neg(\neg p \vee (\neg q \vee r)) \vee \neg((\neg\neg q \wedge \neg r) \vee p))$
$((\neg\neg p \wedge \neg(\neg q \vee r)) \vee (\neg(\neg\neg q \wedge \neg r) \wedge \neg p))$
$((\neg\neg p \wedge (\neg\neg q \wedge \neg r)) \vee ((\neg\neg\neg q \vee \neg\neg r) \wedge \neg p))$
$((p \wedge q \wedge \neg r) \vee ((\neg q \vee r) \wedge \neg p))$
$((p \wedge q \wedge \neg r) \vee (\neg q \wedge \neg p) \vee (r \wedge \neg p))$
qui est une fnd.

Par la seconde méthode on peut construire la table de vérité de la formule :

p	q	r	$(q \supset r)$	$(p \equiv (q \supset r))$	f	
0	0	0	1	0	1	$\neg p \wedge \neg q \wedge \neg r$
0	0	1	1	0	1	$\neg p \wedge \neg q \wedge r$
0	1	0	0	1	0	
0	1	1	1	0	1	$\neg p \wedge q \wedge r$
1	0	0	1	1	0	
1	0	1	1	1	0	
1	1	0	0	0	1	$p \wedge q \wedge \neg r$
1	1	1	1	1	1	

La fnd que l'on trouve est donc $((\neg p \wedge \neg q \wedge \neg r) \vee (\neg p \wedge \neg q \wedge r) \vee (\neg p \wedge q \wedge r) \vee (p \wedge q \wedge \neg r))$. Elle semble à priori différente de la première. Nous allons voir que ces deux formes sont équivalentes : à partir des lignes 1 et 2 on peut construire la formule $(\neg p \wedge \neg q)$, à partir des lignes 2 et 4 on peut construire la formule $(\neg p \wedge r)$, la formule 7 restant isolée. La disjonction des interprétations nous redonne la forme normale trouvée précédemment.
La table de vérité permet de construire une fnc, en utilisant les maxtermes

$$((p \vee \neg q \vee r) \wedge (\neg p \vee q \vee r) \wedge (\neg p \vee q \vee \neg r) \wedge (\neg p \vee \neg q \vee \neg r))$$

Exercice 23.5.3 ($*$)
Normalisez les formules suivantes

1. $(A \equiv B)$
2. $((A \vee B) \supset (B \wedge A))$
3. $(A \equiv (B \vee C))$
4. Généralisez pour n quelconque $(A \equiv (F_1 \vee \ldots F_n))$
5. $(A \equiv \neg(B \equiv C))$

23.5.3 formes duales

Définition 23.5.10 Soit une formule F ne contenant que des connecteurs pris dans $\{\neg, \vee, \wedge\}$, et soit $\overline{F}$ la formule obtenue à partir de F en interchangeant les connecteurs $\wedge$ et $\vee$ et en remplaçant les littéraux par leurs opposés. $\overline{F}$ est la forme `duale` de F. ◀

Propriété 23.16 Pour toute formule $F \in \mathcal{F}$ construit sur $\{\neg, \vee, \wedge\}$, on a $\overline{F} = \neg F$. ♦

La démonstration se fait par induction sur le nombre de connecteurs apparaissant dans la formule F.

Propriété 23.17 Soient $F, G, H \in \mathcal{F}$ des formules construites sur $\{\neg, \vee, \wedge\}$, on a les propriétés suivantes :

1. Si $F = G$ alors $\overline{F} = \overline{G}$
2. Si $F = (G \vee H)$ alors $\overline{F} = (\overline{G} \wedge \overline{H})$
3. Si $F = (G \wedge H)$ alors $\overline{F} = (\overline{G} \vee \overline{H})$.

♦

La preuve est immédiate, puisqu'elle repose sur le lien entre formules logiques et fonctions booléennes.

23.6 Logique et démonstration

Dans cette section, nous allons faire le lien entre la logique (propositionnelle) et les preuves. Dans la partie précédente nous avons dit que chercher un modèle peut s'avérer long et fastidieux, en effet il faut associer à chaque variable une interprétation puis regarder la valeur de la formule. S'il n'existe pas de modèle, ou si l'on n'a pas de chance il faut examiner 2^k interprétations différentes pour répondre. Il est bien plus facile de répondre à la question « Est-ce que telle interprétation est un modèle de la formule ? », dans ce cas il est juste nécessaire de faire la calcul en utilisant les tables de vérité. Répondre à la question « Est-ce que cette formule est une tautologie ? » ou « Est-ce que cette formule est inconsistante ? » est parfois plus facile. La technique utilisée est celle de la preuve par l'absurde. Si l'on s'intéresse aux trois connecteurs $\{\vee, \wedge, \supset\}$, on s'aperçoit que 3 cas sur 4 sont identiques, l'idée est de faire l'hypothèse que l'on est dans le cas qui n'arrive qu'une fois sur 4, et voir si cette hypothèse conduit à une contradiction. Cette technique que l'on nomme algorithme de réduction s'appuye sur la notion de *substiution uniforme*

Définition 23.6.1 (substitution uniforme) Le principe de la substitution uniforme consiste à remplacer, dans une formule logique, toutes les occurrences d'une proposition par une formule, on notera $F_{[p/f]}$ cette opération, où F est la formule sur laquelle on applique la substitution, p est la proposition remplacée et f la formule de remplacement. ◄

La substitution uniforme permet de générer autant de tautologies que l'on veut. En effet, si F est une tautologie, si p est une proposition et si f est une formule alors $F_{[p/f]}$ est une tautologie.

Exemple 23.18
Considérons la formule $F = (p \vee \neg p)$ et la formule $f = (q \wedge \neg r)$. F est une tautologie comme l'est aussi $F_{[p/f]} = ((q \wedge \neg r) \vee \neg(q \wedge \neg r))$ †

Exemple 23.19
Montrez que la formule $((p \wedge q) \supset (p \vee q))$ est une tautologie. †

Quand on regarde la table de $\supset$, on s'aperçoit qu'il n'y a qu'un seul cas où cette formule est fausse. Faisons donc l'hypothèse que nous soyons dans cette situation que peut-il advenir ?

1. On trouve que c'est possible, dans ce cas nous sommes contents puisque nous avons trouvé un exemple montrant que la formule de départ n'est pas vraie, et par conséquent nous avons une preuve que ce n'est pas une tautologie

2. On aboutit à une contradiction, dans ce cas, une fois de plus nous sommes satisfaits, nous avons fait une preuve « ex absurdo » établissant qu'il ne pouvait pas y avoir de situation dans laquelle la formule était fausse, c'est donc une tautologie.

Schéma de solution 7
Hypothèse Il existe une interprétation $\mathbf{I}$ telle que $\mathbf{I}[\![F]\!] = 0$. Grâce à la table de vérité de $\supset$, et en appliquant le mécanisme de substitution uniforme, on en déduit que $\mathbf{I}[\![(p \wedge q)]\!] = 1$ et que $\mathbf{I}[\![(p \vee q)]\!] = 0$. En examinant la table de $\wedge$, on constate que la seule possibilité est que $\mathbf{I}[\![p]\!] = \mathbf{I}[\![q]\!] = 1$, en examinant la table de $\vee$, on constate que dans ce cas, $\mathbf{I}[\![(p \vee q)]\!] = 1$ ce qui est contraire à ce que nous avons déduit de notre hypothèse, en conséquence notre hypothèse était fausse et donc la formule initiale est **toujours** vraie.

Exemple 23.20
Soit la formule $F = (((p \wedge q) \supset r) \supset (p \supset (q \supset r)))$. †

Schéma de solution 8

- Posons $F_1 = ((p \wedge q) \supset r)$, et $F_2 = (p \supset (q \supset r))$. Et faisons l'hypothèse que $\mathbf{I}[\![F]\!] = 0$. Grâce au principe de substitution, on établit que $\mathbf{I}[\![F_1]\!] = 1$ et $\mathbf{I}[\![F_2]\!] = 0$. D'où l'on tire que $\mathbf{I}[\![p]\!] = 1$ et $\mathbf{I}[\![(q \supset r)]\!] = 0$, c'est-à-dire que $\mathbf{I}[\![q]\!] = 1$ et $\mathbf{I}[\![r]\!] = 0$.
- Remplaçons ces valeurs dans F_1, on aboutit à $\mathbf{I}[\![F_1]\!] = 0$, ce qui est contraire aux hypothèses.
- On en déduit que F est une tautologie.

Exemple 23.21
Montrez que la formule $((p \vee q) \supset (p \wedge q))$ est une tautologie. †

Schéma de solution 9
Hypothèse Il existe une interprétation $\mathbf{I}$ telle que $\mathbf{I}[\![F]\!] = 0$. En appliquant principe de substitution uniforme, $\mathbf{I}[\![(p \vee q)]\!] = 1$ et que $\mathbf{I}[\![(p \wedge q)]\!] = 0$.
Là, nous n'avons pas de chance, il va falloir faire une étude de cas

1. Supposons $\mathbf{I}[\![p]\!] = \mathbf{I}[\![q]\!] = 0$, on obtient $\mathbf{I}[\![(p \wedge q)]\!] = \mathbf{I}[\![p \vee q)]\!] = 0$, ce cas n'est pas en accord avec nos hypothèses
2. Supposons $\mathbf{I}[\![p]\!] = 0, \mathbf{I}[\![q]\!] = 1$, on obtient $\mathbf{I}[\![(p \wedge q)]\!] = 0$, $\mathbf{I}[\![p \vee q)]\!] = 1$. Ce cas est conforme à nos hypothèses. Inutile de regarder les autres cas

Nous avons pu trouver une interprétation telle que $\mathbf{I}[\![F]\!] = 0$ ($\mathbf{I}[\![p]\!] = 0$ et $\mathbf{I}[\![q]\!] = 1$), la formule $((p \vee q) \supset (p \wedge q))$ **n'est pas une tautologie**.

23.6.1 Méthode de résolution de Robinson

Cette méthode s'applique sur les ensembles de clauses, c'est-à-dire les formes normales conjonctives. C'est une méthode *syntaxique*, la preuve de son fonctionnement s'appuye sur la *sémantique*. On montre que chaque étape préserve les modèles de la formule.

Définition 23.6.2 Soient deux clauses c_1 et c_2 et soit l un littéral tel que $l \in c_1$ et $\neg l \in c_2$. La résolvante est obtenue en ajoutant les littéraux de c_1 et de c_2 et en faisant disparaître les occurrences (positive et négative) de l, plus formellement $r = (c_1 \setminus \{l\}) \cup (c_2 \setminus \{\neg l\})$. ◀

La propriété suivante établit que chercher un modèle pour deux clauses c_1 et c_2, revient à chercher un modèle pour la résolvante r.

Propriété 23.18 Soient deux clauses $c_1 = \{l_{1_1}, \ldots, l_{k_1}\}$ et $c_2 = \{l_{1_2}, \ldots, l_{k_2}\}$ supposons (sans perte de généralité) que $l_{1_1} = \neg l_{1_2}$. Alors tout modèle de $\{c_1, c_2\}$ est un modèle de la clause $\{l_{2_1}, \ldots, l_{k_1}, l_{2_2}, \ldots, l_{k_2}\}$ ♦

Soit I un modèle de $\{c_1, c_2\}$, il y a deux cas à considérer. Soit $\mathbf{I}[\![l_{1_1}]\!] = 1$, donc $\mathbf{I}[\![l_{1_2}]\!] = 0$, comme I est un modèle de c_2, il existe un $i \in 2_2..k_2$ tel que $\mathbf{I}[\![l_i]\!] = 1$ d'où I est un modèle de r. Le deuxième cas, symmétrique du premier est $\mathbf{I}[\![l_{1_1}]\!] = 0$, donc $\mathbf{I}[\![l_{1_2}]\!] = 1$; par le même type de raisonnement, on aboutit à la même conclusion. Donc on a établi la proposition.

Exemple 23.22
Considérons 4 clauses $c_1 = \{p, q, r\}, c_2 = \{\neg p, q\}, c_3 = \{\neg q, r\}, c_4 = \{\neg r, p\}$. à partir de c_1 et c_2, on peut construire $c_{12} = \{q, r\}$ en combinant ensuite avec c_3 on obtient $c_{123} = \{r\}$. Si on combine ensuite avec c_4 on aboutit à $c_{1234} = \{p\}$. Enfin en combinant de nouveau avec c_2 on trouve $c_{12342} = \{q\}$. Ce qui revient à dire qu'un modèle de $\{c_1, c_2, c_3, c_4\}$ est un modèle de $(p \wedge (q \wedge r))$ ou que $\{c_1, c_2, c_3, c_4\} \cup \{\{\neg p, \neg q, \neg r\}\}$ est incohérent.

On constate que si le principe de cette résolution est simple, il n'en demeure pas moins qu'elle est longue et fastidieuse, car elle nécessite de combiner toutes les clauses. Néanmoins le processus peut être amélioré, en évitant la création de tautologies, en suivant une résolution linéaire, ou en restreignant le type de clauses (clauses de Horn [10], par exemple). La résolution sera dite linéaire si la résolvante construite à la $k+1$ème étape utilise la résolvante trouvée à l'étape k. Certaines de ces améliorations ont permis la mise en place du langage PROLOG

23.6.2 Méthode de résolution sémantique

Dans ce cadre, établir la cohérence d'un ensemble de clauses, revient à chercher un modèle, c'est-à-dire assigner une valeur de vérité à chaque variable propositionnelle. Ce principe d'affectation est un problème que l'on peut présenter sous forme récursive :

```
choisir un littéral l
I[l] = 1
Si l'affectation rend vraie toutes les clauses Alors
   fin un modèle a été trouvé
Sinon Si l'affectation provoque une incohérence Alors
   une clause ne contient que des littéraux faux
   il faut soit changer l'affectation soit revenir à un choix antérieur
   Si toutes les alternatives sont épuisées Alors
       fin l'ensemble de clauses est incohérent
   Fsi
Sinon
   Le processus d'affectation doit continuer
Fsi
```

Exemple 23.23
Soit $\mathcal{C} = \{\{\neg p\}, \{p, q\}, \{p, r\}, \{p, \neg s\}\}$ et la formule $F = ((p_1 \wedge \neg p) \vee \neg p_1)$. On construit le nouvel ensemble de clauses $\mathcal{C} \cup \{\neg F\} = \mathcal{C} \cup \{\{\neg p_1, p\}, \{p_1\}\} = \mathcal{C}'$.
On pose $\mathbf{I}[\![p_1]\!] = 1$, cette affectation ne pose pas de problème pour $\mathcal{C}$, pour que $\mathcal{C}'$ admette un modèle il faut que $\mathbf{I}[\![p]\!] = 1$ et que $\mathcal{C}$ admette un modèle. Ce n'est pas le cas à cause de la clause $\{\neg p\}$. Il faut revenir sur un choix antérieur, si $\mathbf{I}[\![p]\!] = 0$ alors $\mathcal{C}'$ est incohérent. Il faut revenir sur l'affectation de p_1 et choisir $\mathbf{I}[\![p_1]\!] = 0$, ce qui rend $\mathcal{C}'$ incohérent. Donc on établit que $\mathbf{I}[\![\mathcal{C}]\!] = \mathbf{I}[\![F]\!]$.

10. Clause de Horn : clause ayant au plus un littéral positif.

Nous allons maintenant étudier l'algorithme proposé par Davis et Putnam (1960) et proposer quelques heuristiques (astuces ne modifiant pas la validité du processus mais améliorant sa rapidité).

23.6.3 Algorithme de Davis et Putnam

Cette partie est directement inspirée de la thèse de doctorat d'A. Rauzy [3]

Définition 23.6.3 Soit $\mathcal{C}$ un ensemble de clauses réduit. On note $\mathrm{T}_{l_1,l_2,\ldots,l_n}(\mathcal{C})$ l'ensemble de clauses obtenu à partir de $\mathcal{C}$ en supprimant les clauses contenant l_i ou les occurrences opposées de l_i, pour tout $i \in 1..n$. ◀

Exemple 23.24
Soit $\mathcal{C}_0 = \{\{a, b, c\}, \{\neg a, b\}, \{\neg b, c\}, \{\neg c, a\}\}$.
$\mathrm{T}_a(\mathcal{C}_0) = \{\{b\}, \{\neg b, c\}\}$; $\mathrm{T}_{abc}(\mathcal{C}_0) = \emptyset$; $\mathrm{T}_{\neg a,b}(\mathcal{C}_0) = \{\{c\}\}$.

Remarque 23.6.1
L'ordre des opérations n'influe pas sur le résultat, on a les égalités suivantes :

$$\mathrm{T}_{abc}(\mathcal{C}) = \mathrm{T}_a(\mathrm{T}_{bc}(\mathcal{C})) = \mathrm{T}_{bc}(\mathrm{T}_a(\mathcal{C}))$$

∘

Propriété 23.19 Soit $\mathcal{C}$ un ensemble de clauses réduit. Un modèle de $\mathcal{C} \cup \{l_1\} \cup \{l_2\} \cup \{l_n\}$ est un modèle de $\mathrm{T}_{l_1,l_2,\ldots,l_n}(\mathcal{C})$. ♦

Soit I un modèle de $\mathcal{C} \cup \{l_1\} \cup \{l_2\} \cup \{l_n\}$. Par définition, I est un modèle de $\mathcal{C}$ et vérifie $\mathbf{I}[\![l_1]\!] = \mathbf{I}[\![l_2]\!] = \ldots = \mathbf{I}[\![l_n]\!] = 1$. I rend vraie toutes les clauses de $\mathcal{C}$ qui contienne un l_i (par subsomption entre autre), dans toutes les autres clauses de $\mathcal{C}$, on peut supprimer les occurrences négatives des l_i, les clauses résiduelles appartiennent à T (par construction).

Exemple 23.25
Soit $\mathcal{C}_0 = \{\{a, b, c\}, \{\neg a, b\}, \{\neg b, c\}, \{\neg c, a\}\}$, et considérons $\mathrm{T}_{\neg a,b}(\mathcal{C}_0) = \{\{c\}\}$. Soit I telle que $\mathbf{I}[\![\neg a]\!] = \mathbf{I}[\![b]\!] = \mathbf{I}[\![c]\!] = 1$ alors I est un modèle de $\mathcal{C}_0$.

Propriété 23.20 Soit $\mathcal{C}$ un ensemble de clauses réduit, et soient l_1, l_2, $\ldots l_n$ des littéraux distincts deux à deux et tels que $\nexists i \neq j$ avec l_i et l_j opposé. Alors l'ensemble $\mathrm{T}_{l_1,l_2,\ldots,l_n}(\mathcal{C})$ est équivalent à $\mathrm{T}^*_{l_1,l_2,\ldots,l_n}(\mathcal{C})$ obtenu par subsomption. ♦

Propriété 23.21 Soit $\mathcal{C}$ un ensemble de clauses réduit, et soient l_1, l_2, $\ldots l_n$ des littéraux distincts deux à deux et tels que $\nexists i \neq j$ avec l_i et l_j opposé. Alors $\mathrm{T}^*_{l_1,l_2,\ldots,l_n}(\mathcal{C})$ et $\mathcal{C} \cup \{l_1\} \cup \{l_2\} \cup \{l_n\}$ ont un même modèle. ♦

T^* et T admettent le même modèle.

Propriété 23.22 Soit $\mathcal{C}$ un ensemble de clauses et soit p une variable propositionnelle apparaissant dans $\mathcal{C}$. $\mathcal{C}$ est cohérent si et seulement si $\mathrm{T}^*_p(\mathcal{C})$ ou $\mathrm{T}^*_{\neg p}(\mathcal{C})$ est cohérent. ♦

Puisque p est une variable apparaissant dans $\mathcal{C}$, un modèle de $\mathcal{C}$ est une extension d'un modèle qui rend p ou $\neg p$ vrai. Donc soit $\mathcal{C} \cup \{p\}$ est cohérent, soit $\mathcal{C} \cup \{\neg p\}$ est cohérent. On peut alors établir (grâce à la propriété 23.21) que soit $\mathrm{T}^*_p(\mathcal{C})$ est cohérent, soit $\mathrm{T}^*_{\neg p}(\mathcal{C})$ est cohérent. La réciproque s'établissant par la même propriété.

On a donc un moyen constructif pour établir la cohérence d'un ensemble donné de clauses, c'est ce moyen qui est l'algorithme de Davis et Putnam.

```
Soit C un ensemble de clauses réduit
Si C = ∅ alors C est cohérent
Sinon Si C contient □ alors C est incohérent
      Sinon
        on choisit une variable propositionnelle p et
        Si T*_p(C) et T*_¬p(C) sont incohérents
        Alors C est incohérent.
        Fsi
      Fsi
Fsi
```

Exemple 23.26
Soit l'ensemble de clauses $\mathcal{C}_0 = \{\{p, q\}, \{p, r\}, \{\neg p, q\}, \{\neg q\}\}$.
Choisissons la variable p, on construit alors :
$\mathrm{T}^*_p(\mathcal{C}_0) = \{\{q\}, \{\neg q\}\}$; $\mathrm{T}^*_{\neg p}(\mathcal{C}_0) = \{\{q\}, \{r\}, \{\neg q\}\}$
Choisissons la variable q, on construit alors :
$\mathrm{T}^*_{p,q} = \{\square\}$; $\mathrm{T}^*_{p,\neg q} = \{\square\}$ qui sont tous les deux incohérents, donc T^*_p est incohérent.
$\mathrm{T}^*_{\neg p,q} = \{\{r\}, \square\}$; $\mathrm{T}^*_{\neg p,q} = \{\square, \{r\}\}$ qui sont tous les deux incohérents donc $\mathrm{T}^*_{\neg p}$ est incohérent.
Donc $\mathcal{C}_0$ est incohérent.

Un exemple de codage d'un ensemble de clauses

Il est possible de représenter un ensemble de clauses par le biais d'une matrice creuse, dont les lignes correspondent aux clauses et les colonnes aux littéraux.

Exemple 23.27
Regardons le codage de l'ensemble $\{\{p, q, \neg r\}, \{q, s\}, \{\neg p, q\}, \{\neg r, s\}, \{p, \neg r\}\}$.

$$\begin{pmatrix} p & q & \neg r & \\ & q & & s \\ \neg p & q & & \\ & & \neg r & s \\ p & & \neg r & \end{pmatrix}$$

Par rapport à l'algorithme, sélectionner une variable, revient à sélectionner une colonne. Choisir un littéral c'est mettre un 1 à sa place et un 0 à la place de sa forme négative. Simplifier une clause c'est supprimer toutes les lignes où il y a un 1.

Exemple 23.28
En continuant l'exemple 23.27, et en supposant que l'on ait choisi le littéral p on obtient :

$$\begin{pmatrix} 1 & q & \neg r & \\ & q & & s \\ 0 & q & & \\ & & \neg r & s \\ 1 & & \neg r & \end{pmatrix}$$

Ce qui donne après simplification :

$$\begin{pmatrix} q & & s \\ q & & \\ & \neg r & s \end{pmatrix}$$

En prenant le littéral q, après simplification on obtient :

$$\begin{pmatrix} \neg r & s \end{pmatrix}$$

Si on choisit ensuite le littéral $\neg r$, le processus s'arrête (suppression de toutes les lignes), si par contre on choisit le littéral r, on se retrouve avec une matrice 1x1 où l'on est forcé de prendre le littéral s, sous peine d'incohérence.

Quelques heuristiques

Parmi les heuristiques possibles :

1. Privilégier le choix des variables propositionnelles correspondant à un littéral isolé dans une clause.
2. Choisir les littéraux apparaissant dans les clauses les plus courtes.
3. Choisir les littéraux qui n'apparaissent que positivement (resp. négativement), et parmi ceux-ci ceux qui sont présents le plus souvent.

à titre d'exemple, prenons l'ensemble de clauses $\{\{p, q, \neg r\}, \{q, s\}, \{\neg p, q\}, \{\neg r, s\}, \{p, \neg r\}\}$.
Sans heuristique, on choisit les littéraux par ordre alphabétique.
Choisir p : $\mathrm{T}^*_p = \{\{q\}, \{\neg r, s\}\}$
Choisir q : $\mathrm{T}^*_{p,q} = \{\{\neg r, s\}\}$
Choisir r : $\mathrm{T}^*_{p,q,r} = \{\{s\}\}$
Choisir s : $\mathrm{T}^*_{p,q,r,s} = \emptyset$
On a donc trouver un modèle de l'ensemble initial, pour trouver tous les modèles, il faut continuer le processus.

Si par contre on applique les heuristiques.
Choisir q : $\mathrm{T}^*_q = \{\{\neg r, s\}, \{p, \neg r\}\}$
Choisir $\neg r$: $\mathrm{T}^*_{q,\neg r} = \emptyset$
On a donc trouver un modèle de l'ensemble initial, bien entendu si l'on cherche tous les modèles le processus doit se poursuivre, mais le calcul demeure plus rapide.

23.7 Exercices de révision et compléments

La quotation des exercices « $*$ » signifie qu'il s'agit d'une application directe des définitions, « $**$ » désigne les exercices que vous devez savoir résoudre, « $***$ » se rapporte à des exercices plus délicats à résoudre.

algèbres de Boole & fonctions booléennes

Exercice 23.7.1 $(*)$
Soit $B = (E, \sqcap, \sqcup, \bar{}, \bot, \top)$ une algèbre de Boole. Donnez la preuve des lois de De Morgan (propriété 23.2).

Exercice 23.7.2 $(*)$
Soit $B = (E, \sqcap, \sqcup, \bar{}, \bot, \top)$ une algèbre de Boole.

- $\forall x \in E, \overline{x}$ est unique
- $\forall x \in E, \overline{\overline{x}} = x$

Exercice 23.7.3 (∗)
Soit $\mathbb{B} = (\mathcal{B}, ., +, \bar{}, 0, 1)$ l'algèbre de Boole minimale et soit $\mathbb{B}^3 = (E', \sqcap, \sqcup, \texttt{neg}, \bot, \top)$ défini par :

- $E' = \{(x, y, z) \ / \ x, y, z \in \mathcal{B}\}$;
- $\texttt{neg}(x, y, z) = (\bar{x}, \bar{y}, \bar{z})$;
- $(a, b, c) \sqcap (x, y, z) = (a.x, b.y, c.z)$;
- $(a, b, c) \sqcup (x, y, z) = (a + x, b + y, c + z)$.

1. Combien y a-t-il d'éléments dans E' ?
2. Exprimez les éléments $\bot$ et $\top$ de E' en fonction d'éléments de $\mathcal{B}$.
3. Établir que les opérations $\sqcup$ et $\sqcap$ sont idempotentes.

Exercice 23.7.4 (∗∗)
Soit $B = (E, ., +, \bar{}, 0, 1)$ une algèbre de Boole, et $e \neq 0$ un élément particulier de E. On construit $E' = \{x \in E \text{ tel que } x.e = x\}$, montrer que $(E', ., +, \hat{}, 0, e)$ est aussi une algèbre de Boole, avec $\hat{x} = \bar{x}.e$

Dans la suite, nous nous plaçons dans le cadre de l'algèbre de Boole minimale $\mathbb{B}$.

Exercice 23.7.5 (∗)
Soit la fonction f telle que $f(0, 0) = f(1, 0) = 1$; que l'on aurait aussi pu décrire par $f(x, 0) = 1$.

1. Écrire la table de vérité de f.
2. Quel est le mot binaire associé à f ?
3. Exprimer f comme somme de mintermes et comme produit de maxtermes.
4. Exprimer f sous forme conjonctive
5. Exprimer f sous forme disjonctive

Exercice 23.7.6 (∗)
Soient f et g deux fonctions booléennes vérifier que :

1. $\widehat{(f + g)} = (\widehat{f} \ . \ \widehat{g})$
2. $\widehat{(f.g)} = (\widehat{f} \ + \ \widehat{g})$
3. $\widehat{\widehat{f}} = f$
4. Calculer les duales de x, $\bar{x}$, $(x + \bar{y})$, $x\bar{y}$ et $(x + y + \bar{z}).\overline{(\bar{x} + y + \bar{z})}$

Exercice 23.7.7 (∗)
Vérifier les trois points suivants en utilisant les tables de vérité, les deux premiers points sont connus sous le nom de « loi de De Morgan » :

1. $\overline{x.y} = \bar{x} + \bar{y}$;
2. $\overline{x + y} = \bar{x}.\bar{y}$;
3. $\bar{\bar{x}} = x$.

Exercice 23.7.8 (∗)
On souhaite montrer que la duale de $x.y + \overline{x}$ est $y.\overline{x}$. On procèdera par étapes :

1. Faire les tables de vérité
2. En vous appuyant sur les propriétés de $\mathbb{B}$, montrer que :
   a) $(x + y).\overline{x} = y.\overline{x}$.
   b) $\overline{x} = \overline{x}.y + \overline{x}$
   c) $x.y + \overline{x}.y = y$
   d) $x.y + \overline{x} = \overline{x} + y$

Exercice 23.7.9 (∗∗)
En utilisant les exercices précédents, démontrer que si f est sous forme polynomiale, trouver une forme polynomiale de $\widehat{f}$ consiste simplement à inverser les . et +. Exemple $f(x, y) = \bar{x}.y$ alors $\widehat{f(x, y)} = \bar{x} + y$.

Exercice 23.7.10 (∗∗)
Montrer que :
Soit f une fonction booléenne à $k + 1$ arguments, on a :

$$f(x_0, x_1, \ldots, x_k) = \overline{x_0} f(0, x_1, \ldots, x_k) + x_0 f(1, x_1, \ldots, x_k)$$

En déduire que toute fonction booléenne est polynomiale i.e. peut s'écrire en n'utilisant que les opérations de l'algèbre de Boole minimale.

Exercice 23.7.11 (∗∗)
Montrer que :

1. Si g est la duale de f, alors f est la duale de g.
2. Si $f(x_1, \ldots, x_p) = g(g_1(x_1, \ldots, x_p), \ldots, g_k(x_1, \ldots, x_p))$ alors
   $\hat{f}(x_1, \ldots, x_p) = \hat{g}(\hat{g_1}(x_1, \ldots, x_p), \ldots, \hat{g_k}(x_1, \ldots, x_p))$
3. **Illustration** Calulez la duale de f définie par $f(x, y, z) = (x + y).(y + z)$ en donnant les fonctions booléennes $g, g_1, \ldots, g_k$

Exercice 23.7.12 (∗)
Quelles sont les fonctions duales de `nand`, `nor`, `xor` et `ite` décrites dans la table ci-dessous :

x	y	nand(x,y)	nor(x,y)	xor(x,y)	ite(x,y,z)
0	0	1	1	0	z
0	1	1	0	1	z
1	0	1	0	1	0
1	1	0	0	0	1

TABLE 23.2 – Opérations dans $\mathbb{B}$

Exercice 23.7.13 (∗)
Soit la fonction boolénne `xor`[11], définie par : $\texttt{xor}(x,y) = x\bar{y} + \bar{x}y$. Montrer, sans utiliser la table de vérité, que :

1. `xor` est commutative et associative ;
2. $\texttt{xor}(x, x) = 0$;
3. $\texttt{xor}(x, \bar{x}) = 1$;
4. $z.\texttt{xor}(x, y) = \texttt{xor}(xz, yz)$.

Exercice 23.7.14 (∗)
Soit la fonction définie par $f(0, 0) = f(0, 1) = f(1, 0) = 1$, montrer sans utiliser la table de vérité qu'elle peut s'écrire sous la forme $\bar{x} + \bar{y}$.

Exercice 23.7.15 (∗)
Exprimer la fonction $f(0, y) = 1,\ f(1, y) = y$ sous forme polynomiale.

Exercice 23.7.16 (∗∗)
Soit la fonction `ite` (pour if-then-else) définie par $\texttt{ite}(1, x, y) = x$; $\texttt{ite}(0, x, y) = y$.

1. Exprimer $\texttt{ite}(x,y,z)$ sous forme polynomiale ;
2. Exprimer $x + y$ en n'utilisant que la fonction `ite` ;
3. Exprimer $x.y$ en n'utilisant que la fonction `ite` ;
4. Exprimer $\bar{x}$ en n'utilisant que la fonction `ite` ;

introduction à la logique propositionnelle

Exercice 23.7.17 (∗)
Tiré de [6]. Exprimez, sans utiliser la forme « il est faux que », la négation des affirmations suivantes.

1. « S'il pleut demain ou s'il fait froid, je ne sortirai pas » ;
2. « Le nombre 522 n'est pas divisible par 3 mais il est divisible par 7 » ;
3. « fumer provoque le cancer » ;
4. « si l'on mange des bonbons, alors on a des caries » ;
5. « si Paul ne va pas travailler ce matin, il va perdre son emploi » ;
6. « tout nombre entier impair peut être divisible par 3 ou par 5 mais pas par 2 ».

Exercice 23.7.18 (∗)
Tiré de [6]. On note **p** l'affirmation « Jean est fort en maths » et **q** l'affirmation « Jean est fort en chimie ». En utilisant les connecteurs logiques et les variables propositionnelles p et q, exprimez les affirmations suivantes :

1. « Jean est fort en maths mais faible en chimie » ;

11. `xor` s'appelle le ou-exclusif par opposition au ou-inclusif (le ou classique)

2. « Jean n'est fort ni en maths ni en chimie » ;

3. « Jean est soit fort en maths, soit fort en chimie et faible en maths » ;

4. « Jean est fort en maths s'il est fort en chimie »

Exercice 23.7.19 (∗)
Soit **p**, et **q** deux variables propositionnelles, construire la table de vérité correspondant à l'expression : « ni p, ni q ». En déduire une formulation reposant sur les cinq connecteurs logiques.

Exercice 23.7.20 (∗)
Construire la table de vérité des formules suivantes :

1. $(\neg p \wedge q)$;

2. $(\neg p \supset (p \vee q))$;

3. $(p \supset (\neg p \supset p))$;

4. $\neg(\neg p \wedge \neg q)$

Exercice 23.7.21 (∗)
On considère la formule $F = (p \supset q)$, que l'on suppose **vraie**. On s'intéresse aux conséquences que l'on peut tirer de F lorsque

- On affirme l'antécédent : p ;
- On affirme le conséquent : q ;
- On nie l'antécédent : $\neg p$;
- On nie le conséquent : $\neg q$;

Pour vous aidez, vous pouvez retranscrire les informations en français :

- p : « je suis malade » ;
- q : « je reste au lit » ;

Exercice 23.7.22 (∗)
Soit p la proposition « il pleut », q la proposition « il y a des nuages ». Écrire en français $p \supset q$, sa contraposée, sa réciproque, la contraposée de la réciproque. Quelles implications sont vraies ?

Exercice 23.7.23 (∗)
Montrez que, F et G étant des formules logiques.

Rappel : $\mathbf{I[}\neg F\mathbf{]} = 1 - \mathbf{I[}F\mathbf{]}, \mathbf{I[}(F \wedge G)\mathbf{]} = \min(\mathbf{I[}F\mathbf{]}, \mathbf{I[}G\mathbf{]}), \mathbf{I[}(F \vee G)\mathbf{]} = \max(\mathbf{I[}F\mathbf{]}, \mathbf{I[}G\mathbf{]})$

1. $\mathbf{I[}\neg\neg F\mathbf{]} = \mathbf{I[}F\mathbf{]}$

2. $\mathbf{I[}(F \wedge \neg F)\mathbf{]} = 0$

3. $\mathbf{I[}(F \vee \neg F)\mathbf{]} = 1$

4. $\mathbf{I[}\neg(F \vee G)\mathbf{]} = \mathbf{I[}(\neg F \wedge \neg G)\mathbf{]}$

5. $\mathbf{I[}\neg(F \wedge G)\mathbf{]} = \mathbf{I[}(\neg F \vee \neg G)\mathbf{]}$

Exercice 23.7.24 (∗)
On note F, G des formules logiques. Montrez, en utilisant les tables de vérités que

1. $\forall F, \forall G, \mathbf{I}[\![(F \supset G)]\!] = \mathbf{I}[\![(\neg F \vee G)]\!]$
2. $\forall F, \forall G, \mathbf{I}[\![(F \equiv G)]\!] = \mathbf{I}[\![((F \supset G) \wedge (G \supset F))]\!]$
3. $\forall F, \forall G, \mathbf{I}[\![\neg(F \wedge \neg G)]\!] = \mathbf{I}[\![(F \supset G)]\!]$

Exercice 23.7.25 (∗)
1. Établir que $\{\neg, \supset\}$ est adéquat, mais que ce n'est pas une base
2. Montrer que $\{\mathtt{i}te\}$ est une base, on rappelle $\mathbf{I}[\![\mathtt{i}te(0, x, y)]\!] = \mathbf{I}[\![y]\!], \mathbf{I}[\![\mathtt{i}te(1, x, y)]\!] = \mathbf{I}[\![x]\!]$
3. Montrer que $\{\mathtt{n}or\}$ est une base, $\mathbf{I}[\![\mathtt{n}or(x, y)]\!] = \mathbf{I}[\![\neg(x \vee y)]\!]$.

Exercice 23.7.26 (∗)
SoientF, G des formules

- Montrez que tout modèle de $(F \wedge G)$ est un modèle de F et un modèle de G
- Montrez que tout modèle de F est un modèle de $(F \vee G)$

Exercice 23.7.27 (∗∗)
1. Montrer que $\supset$ n'est pas associative, illustrer par un exemple en français.
2. Montrer que $\equiv$ est associative.

Exercice 23.7.28 (∗∗)
Pour chacune des formules suivantes, établissez de deux manières différentes s'il s'agit d'une **tautologie**, c'est-à-dire une formule toujours vraie.

$$\begin{array}{lcl}
((p \vee q) & \supset & p) \\
((p \wedge q) & \supset & p) \\
((p \wedge q) & \supset & (p \vee q)) \\
((p \vee q) & \supset & (p \wedge q)) \\
((p \supset \neg q) & \equiv & \neg((p \vee p) \wedge (q \wedge q))) \\
((p \supset (q \supset p)) & \supset & (p \equiv (\neg q \wedge p))) \\
((p \equiv (\neg q \wedge p)) & \supset & (p \supset (q \supset p)))
\end{array}$$

Exercice 23.7.29 (∗∗)
Soient A, B, C, D, E des formules, établir si les formules suivantes sont des tautologies :

1. $((A \equiv (B \supset C)) \equiv ((A \wedge C) \vee (\neg(A \equiv B) \wedge \neg C)))$
2. $((((\neg A \supset B) \supset (\neg A \supset C)) \wedge (B \supset \neg C)) \supset A)$
3. $(((A \wedge B) \supset (C \vee D)) \equiv ((A \supset C) \vee (B \supset D)))$