Université de Namur

Faculté d'Informatique

INFO B125
Mathématiques pour l'informatique (première partie)

Marie-Ange Remiche

Cours donné par Martine De Vleeschouwer

Année académique 2017-2018

Table des matières

Avant-propos

Ce syllabus constitue une référence pour le cours INFO-B-125 mais ne contient pas nécessairement tous les exemples, toutes les démonstrations, toutes les explications, qu'elles soient intuitives ou plus formelles, que contient le cours oral.

Il se base sur divers ouvrages de base en mathématiques, comme

— K. Houston. *Comment penser comme un mathématicien.* de Boeck, 2011.

— R. Haggarty. *Mathématiques discrètes appliquées à l'informatique.* Pearson Education France, 2005.

— S.G. Krantz. *Discrete Mathematics Demystified.* McGraw Hill, 2009.

— M. Marchand. *Outils mathématiques pour l'informaticien. Deuxième édition.* de Boeck, 2005.

— J. Vélu, G. Avérous, I. Gil et F. Santi. *Mathématiques pour l'informatique.* Dunod, 2008.

— J. Vélu. *Méthodes mathématiques pour l'informatique. Quatrième édition.* Dunod, 2005.

N'hésitez pas à m'indiquer les erreurs que vous auriez pu rencontrer dans ce syllabus. Vos remarques me seront particulièrement utiles pour améliorer ce support de travail.

Pour rappel : Ce support de cours mis sur Internet a pour seule vocation d'être utilisée par les étudiants dans le cadre de leur cursus au sein de l'Université de Namur. Aucun autre usage ni diffusion ne sont autorisés.

Chapitre 1

Logique des propositions

Nous travaillons dans le cadre de ce cours, en logique du premier ordre. Ce chapitre présente des définitions fondamentales. Un élément clé dans la logique est la notion de formule. Celle-ci utilise les connecteurs définis dans la section 1.2, tandis que leurs propriétés sont présentées dans la section 1.4. La notion de tautologie (section 1.3) est présentée préalablement. Nous terminons ce chapitre en définissant la notion de consistance d'un ensemble de propositions.

1.1 Quelques définitions fondamentales

Nous travaillons avec des *propositions* ou des *prédicats*. Voici leur définition.

Définition 1.1

Une proposition *est un énoncé qui est soit vrai soit faux et cela sans ambiguïté.*

La *valeur de vérité* d'une proposition est vrai ou faux. Habituellement, on note la valeur *vrai* par 1 et la valeur *faux* par 0. Il s'agit d'une convention de notation. La logique formelle est basée sur deux hypothèses de départ qui sont

Le tiers-exclu : Toute proposition est soit vraie, soit fausse.

La non-contradiction : Aucune proposition vraie n'est fausse en même temps.

Exemple 1.1 Considérons les cas suivants.

- *Le lundi 13 mars 2012, il pleut.* est bien une proposition.
- *Quel jour sommes-nous ?* n'est pas une proposition.
- $2+3=5$ est une proposition qui est vrai lorsqu'on considère l'arithmétique telle qu'étudiée en primaire.
- *Il fera beau demain, mardi le 27 juillet 2052.* est également une proposition. Sa véracité ne pourra être établie que le 27 juillet 2052.

— *Boire ou conduire, Monsieur Bob doit choisir.* est également une proposition. On peut cependant remarquer, qu'elle est composée de plusieurs propositions.
— *Tout entier plus grand que 2 est la somme de deux nombres premiers.* Cette proposition porte sur tout élément d'un ensemble particulier d'éléments. Cet ensemble est bien précisé, il s'agit des entiers. Notons que personne n'a encore pu établir si cette proposition était vraie ou fausse.

Dans l'exemple *Tout entier plus grand que 2 est la somme de deux nombres premiers*, on remarque que la proposition est énoncée pour un ensemble d'éléments, à savoir les *entiers*. Appelons un de ses éléments x, on a alors pour cet élément particulier, la proposition $p(x)$, soit x *est la somme de deux nombres entiers*. Cette proposition dont la paramètre est x, porte le nom de *prédicat*.

Définition 1.2

Un prédicat *est une proposition exprimée à l'aide d'un paramètre.*

La valeur de vérité d'un prédicat dépend donc de la valeur du ou des paramètres du prédicat.

Exemple 1.2 Considérons les phrases suivantes :
— 2 *est plus petit que* 3
est une proposition vraie.
— n *est un nombre naturel.*
est un prédicat. Sa valeur de vérité dépend de la valeur du paramètre n.
— *Tous les chats sont gris.*
est une proposition fausse. En effet, il existe au moins un chat noir par exemple.
— *Un chat est gris ou noir*
est un prédicat. Le paramètre est le *chat* dont le nom devrait être précisé pour pouvoir connaître la véracité de la proposition qui en résulte.
— *En Belgique, lors des dernières élections, il existait au moins un électeur dont il s'agissait du premier scrutin*
est une proposition vraie si on peut nommer une jeune personne qui avait fêté ses 18 ans l'année du dit scrutin.

Exemple 1.3 Soit les prédicats suivants

1. $p(x)$: x a un manteau rouge,
2. $q(x)$: x ne porte pas de barbe.

Le prédicat $p(x)$ est vrai lorsque x est le *père Noël*. Par contre, dans le cas où x est Saint-Nicolas, $q(x)$ est faux.

Exercice 1.1 Supposons qu'on trouve sur la table trois pièces de monnaie exactement. Lesquelles de ces propositions sont-elles vraies ?

1. Il y a une pièce de monnaie sur la table. F
2. Il y a au moins une pièce de monnaie sur la table. V

3. Il y a quatre pièces de monnaie sur la table.
4. On compte moins de deux pièces sur la table.

La logique a pour objectif de combiner des propositions anonymes, notées usuellement $\mathtt{p}, \mathtt{q}, \mathtt{r}, \ldots$ qu'on appelle *variables propositionnelles*.

Définition 1.3

Une variable propositionnelle *est une proposition anonyme, dont la valeur de vérité n'est pas fixée. Par hypothèse, cette valeur est 0 ou 1.*

On utilise habituellement une lettre minuscule, à partir de la lettre $\mathtt{p}$ pour représenter une variable propositionnelle.

Définition 1.4

Soit $\mathtt{p}$ *une variable propositionnelle. Sa* table de vérité *est la suivante.*

p
1
0

Ces variables propositionnelles élémentaires permettent de construire des propositions plus complexes et cela, grâce aux connecteurs. On obtient alors des *formules* qui sont elles-mêmes des propositions. Dans la section suivante, on définit plusieurs connecteurs. Il faut savoir qu'on construit la table de vérité d'une formule en considérant toutes les combinaisons possibles de valeurs pour les différentes variables propositionnelles intervenantes.

1.2 Les connecteurs

Dans cette section, nous parlerons des connecteurs suivants :
— la négation, notée $\neg$,
— la disjonction, notée $\vee$,
— la conjonction, notée $\wedge$,
— l'implication, notée $\Rightarrow$,
— l'équivalence, notée $\Leftrightarrow$,
— la disjonction exclusive, notée $\oplus$.

La négation d'une proposition

Définition 1.5

La négation *d'une proposition* $\mathtt{p}$*, notée* $\neg\mathtt{p}$ *se définit par la table de vérité suivante*

Not

p	¬p
1	*0*
0	*1*

Le connecteur ¬ est unaire, car il n'a qu'un seul opérande. Il signifie *il est faux que* p.

Exemple 1.4 Reprenons la proposition vraie *2 est plus petit que 3*. Sa négation est *2 n'est pas plus petit que 3*. Clairement, cette nouvelle proposition est fausse.

Si la proposition p est *Je suis étudiant à la faculté d'informatique*. Sa négation est *Je ne suis pas étudiant à la faculté d'informatique*. Remarquez que nous ne mettons pas évidence le caractère vrai ou faux de cette proposition. Nous énonçons simplement un fait.

Les connecteurs suivants sont des connecteurs binaires ; ils mettent en relation deux variables propositionnelles.

La disjonction de deux propositions

Définition 1.6

La disjonction *de deux variables propositionnelles* p *et* q, *notée* $p \vee q$ *est définie par la table de vérité suivante*

p	q	$p \vee q$
1	*1*	*1*
1	*0*	*1*
0	*1*	*1*
0	*0*	*0*

Or

La proposition composée $p \vee q$ se lit p *ou* q. On parle parfois de *ou inclusif*. D'après sa table de vérité, on remarque que cette proposition est vraie si l'une des propositions élémentaires est vraie. L'exemple suivant donne une illustration intuitive pour cette table de vérité.

Exemple 1.5 Considérons l'énoncé suivant. *L'électeur choisit un candidat ou l'électeur choisit une liste.*

Bob a choisi de voter pour Alice de la liste CPP et pour la liste Fortran. D'après la définition du ou inclusif, vérifie-t-il les conditions de la proposition énoncée précédemment ?

Peut-on dés lors en conclure que spécifier les méthodes de vote en Belgique par cette proposition est adéquat ?

Remarque 1.1 Considérons la table de vérité de la proposition $p \vee \neg q \vee p$

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

On remarque que l'opérateur $\neg$ ne porte que sur la proposition q. Il est prioritaire sur tout autre opérateur. Si on désire qu'il porte sur une formule, il faut placer des parenthèses.

Remarque 1.2 Pour calculer la table de vérité de $p \vee \neg q \vee p$, nous avons utilisé la propriété suivante. La proposition $p \vee q \vee r$ est équivalente à $(p \vee q) \vee r$. On parle encore de propriété d'associativité pour l'opérateur $\vee$. Nous y reviendrons dans la proposition (1.17).

Exercice 1.2 Calculez les tables de vérité de $\neg p \vee q$ et de $\neg(p \vee q)$.

Conjonction de deux propositions

Définition 1.7

La conjonction *de deux propositions* p *et* q*, notée* $p \wedge q$ *se définit par la table de vérité suivante*

p	q	$p \wedge q$
1	*1*	*1*
1	*0*	*0*
0	*1*	*0*
0	*0*	*0*

AND

La formule $p \wedge q$ se lit p *et* q. Elle est vraie lorsque les deux propositions élémentaires sont vraies.

Exemple 1.6 *Nous sommes un vendredi et il fait beau.* Pour que cette proposition soit vrai, il faut qu'à la fois, ce jour est un vendredi et qu'on observe beau temps.

Les opérateurs $\vee$ et $\wedge$ ont la même priorité. Dans des formules combinant plus de deux opérandes de même priorité, nous avons la convention suivante. Nous utiliserons des parenthèses pour préciser sur quelle proposition s'applique un opérateur. Dès lors, les formules suivantes

$$(p \wedge q) \vee r \tag{1.1}$$
$$p \wedge (q \vee r) \tag{1.2}$$

sont différentes. Pour s'en convaincre, il suffit d'écrire les tables de vérité de chaque formule. Pour cela, il faut donc bien considérer toutes les combinaisons possibles pour les différentes variables propositionnelles intervenantes. Si il existe une combinaison de valeurs pour p, q et r telles que la valeur de vérité de la formule est différente, alors les formules (1.1) et (1.2) sont bien différentes.

Exercice 1.3 Calculez la table de vérité de $p \wedge ((q \wedge \neg p) \vee \neg p)$.

Exercice 1.4 Complétez la proposition suivante

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

pour obtenir la table de vérité suivante

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

Simplifiez la formule finale, en observant la table de vérité.

Rappelons encore une fois, que dans le cas de l'opérateur unaire $\neg$, celui-ci prend la priorité sur tout autre opérateur. Par exemple, dans la formule

$$\neg p \vee q, \tag{1.3}$$

la négation ne porte que sur la variable p et non sur la formule $p \vee q$. Si tel était le cas, on aurait écrit $\neg(p \vee q)$.

Ces trois premiers connecteurs sont appelés *connecteurs fondamentaux*. Tout autre connecteur peut être défini à partir de ces trois connecteurs fondamentaux. Il en va ainsi du connecteur $\Rightarrow$. En voici sa définition.

L'implication

Définition 1.8

*L'*implication *entre deux propositions* p *et* q*, notée* $p \Rightarrow q$ *se définit par la table de vérité suivante*

Si Alors

p	q	$p \Rightarrow q$
1	*1*	*1*
1	*0*	*0*
0	*1*	*1*
0	*0*	*1*

L'implication $p \Rightarrow q$ se lit *Si* p, *alors* q. D'après la table de vérité qui définit l'opérateur $\Rightarrow$, une implication n'est pas vérifiée lorsque p est vraie et que q est pourtant fausse. On dit encore que l'implication $p \Rightarrow q$ est vraie dès que p est fausse ou que q est vraie.

Exemple 1.7 Considérons les quelques exemples suivants.

1. Soit la proposition suivante,
$$(x \in \mathbb{R}) \wedge (x > 0) \Rightarrow (x^2 < 0 \vee 2x < 0). \tag{1.4}$$
Cette proposition est vrai pour $x = -3$ par exemple mais est fausse lorsque $x = 2$.
2. Soit la proposition *Si les poules ont des dents, alors il pleut demain.* Cette proposition est toujours vraie, en logique mathématique. En français, elle peut apparaître comme innocente. Nous donnons ici une implication qui est vraie puisque l'hypothèse est fausse. En effet, les poules n'ont pas de dents. Dès lors, nous l'avons vu, d'après la table de vérité, lorsque l'hypothèse p (= *les poules ont des dents*) est fausse, alors l'implication $p \Rightarrow q$ est vraie. Attention, c'est l'*implication* qui est vraie, nous ne donnons aucune information quant à la véracité de q.
3. Soit la proposition *Ne peuvent voter que les citoyens belges de plus de 18 ans.* Soit p *être citoyen belge* et q *avoir plus de 18 ans.* Notons enfin la proposition *pouvoir voter* par r. On lit donc la proposition de départ comme $p \wedge q \Rightarrow r$. Dans la législation belge, cet énoncé est vrai.
4. *Si* $1 = 2$, *alors* $5 = 6$ est une implication qui est vraie.
5. *Si* $-1 = 1$, *alors* $1 = 1$ est également une implication qui est vraie.

Nous avons le résultat suivant.

Propriété 1.1

Soit p et q deux variables propositionnelles. La formule $p \Rightarrow q$ est équivalente à $\neg(p \wedge \neg q)$.

Démonstration : Il faut savoir que deux formules sont équivalentes si elles conduisent à la même table de vérité. Construisons les deux tables de vérité.

Nous avons par définition de $\Rightarrow$ (voir la définition 1.8)

p	q	$p \Rightarrow q$
1	1	1
1	0	0
0	1	1
0	0	1

et

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

en utilisant les définitions de la négation (définition 1.5) et de la conjonction (définition 1.7). La dernière colonne de chaque tableau étant identique à l'autre, les deux formules sont bien équivalentes. ■

Remarque 1.3 Pour calculer la table de vérité de $\neg(\mathtt{p}\wedge\neg\mathtt{q})$, on a introduit deux colonnes pour obtenir le résultat final, soit celle correspondant à la négation de la variable propositionnelle $\mathtt{q}$ et celle correspondant à la formule $\mathtt{p}\wedge\neg\mathtt{q}$.

On peut donc dire que le connecteur $\Rightarrow$ est *définissable* à partir des connecteurs $\neg$ et $\wedge$.

Remarque 1.4 Remarquons la présence des parenthèses dans la proposition composée $\neg(\mathtt{p}\wedge\neg\mathtt{q})$. Le connecteur $\Rightarrow$ est moins prioritaire que les opérateurs $\vee$ et $\wedge$. Ainsi la proposition $\mathtt{p}\Rightarrow\mathtt{q}\vee\mathtt{r}$ est la même que $\mathtt{p}\Rightarrow(\mathtt{q}\vee\mathtt{r})$. Dans la suite, nous utilisons souvent la deuxième écriture pour éviter les malentendus.

Exercice 1.5 Calculez les tables de vérité de $\mathtt{p}\Rightarrow\mathtt{q}\vee\mathtt{r}$ et $(\mathtt{p}\Rightarrow\mathtt{q})\vee\mathtt{r}$.

Les rôles joués par $\mathtt{p}$ et $\mathtt{q}$ dans la formule $\mathtt{p}\Rightarrow\mathtt{q}$, sont différents. On ne peut les commuter sans changer la valeur de vérité de la formule.

Définition 1.9

Soit la proposition $\mathtt{p}\Rightarrow\mathtt{q}$. *La proposition* $\mathtt{p}$ *située à gauche du connecteur* $\Rightarrow$ *est appelée* antécédent, *la proposition* $\mathtt{q}$ *est appelée* conséquent.
On parle encore pour $\mathtt{p}$ *d'*hypothèse *et pour* $\mathtt{q}$ *de* conclusion.

Remarque 1.5 Un énoncé d'implication en français s'exprime souvent sous la forme *Si ..., alors ...*. Ce n'est pas toujours le cas, nous pouvons également avoir des énoncés tels que par exemple,

- *La somme de deux nombres pairs est paire.* En effet, pour *conclure* qu'une somme de deux entiers est paire, nous travaillons sous *l'hypothèse* que les deux entiers étaient pairs.
- *J'emporte un parapluie si le temps est couvert.* L'hypothèse est que *le temps est couvert* et la conclusion est que *j'emporte un parapluie.*

Pour observer une implication dans le langage courant, il convient donc de bien identifier les hypothèses et les conclusions.

Voici à présent une définition importante, qui sera utile dans l'énoncé et la preuve de théorème.

Définition 1.10

Lorsque l'implication $\mathtt{p}\Rightarrow\mathtt{q}$ *est vraie, on dit que* $\mathtt{p}$ *est une* condition

suffisante *pour avoir* q *et que* q *est une* condition nécessaire *pour avoir* p.

En effet, lorsque $p \Rightarrow q$ est vraie, établir que p est vraie suffit pour savoir que q est vraie. Par contre, établir que q est vraie, ne suffit pas pour affirmer que p est vraie. De plus si q est fausse, alors nécessairement p sera fausse, toujours dans le contexte où l'implication $p \Rightarrow q$ est vraie.

Exemple 1.8 Prenons cette proposition "ABCD est un losange $\Rightarrow$ ABCD est un parallélogramme". Cet énoncé est vrai. La condition "ABCD est un losange" est une condition suffisante pour observer qu'il s'agit d'un parallélogramme. En effet, tous les losanges sont des parallélogrammes. La condition "ABCD est un parallélogramme" est une condition nécessaire pour observer un losange mais non suffisante. En effet, il existe des parallélogrammes qui ne sont pas des losanges.

Exemple 1.9 Remarquons l'énoncé suivant *Etre un gant est suffisant pour être un vêtement*. On y lit également qu'être un gant n'est pas nécessaire pour être un vêtement.

L'énoncé $x = -2$ est suffisant pour que $x \leq 0$. Nous avons également que $x \leq 0$ est nécessaire pour que $x = -2$. Le fait que $x \leq 0$ n'est par contre pas suffisant pour affirmer que $x = -2$.

Définition 1.11

La formule $q \Rightarrow p$ *est la* réciproque *de* $p \Rightarrow q$.
La formule $\neg q \Rightarrow \neg p$ *est la* contraposée *de* $p \Rightarrow q$.
La formule $\neg p \Rightarrow \neg q$ *est l'*inverse *de* $p \Rightarrow q$.

Exemple 1.10 Soit la proposition *Si tu es sérieux dans tes études, tu étudieras.* Sa contraposée est *Si tu n'étudies pas, tu n'es pas sérieux dans tes études.*

Remarque 1.6 Cet exemple illustre la propriété suivante.Une implication a toujours la même valeur de vérité que sa contraposée. Elles sont dites équivalentes. Pour établir cela, nous pouvons construire les tables de vérité, ou réaliser une preuve en utilisant diverses résultats d'équivalence. Nous démontrons cette propriété via cette approche dans l'exemple 1.15.

Remarque 1.7 Notons que la négation d'une implication exprimée en français, n'est pas p *n'implique pas* q. Prenons l'énoncé suivant *Si il fait beau demain, je vais à la piscine.* La négation de celui-ci est *Il fait beau et je ne vais pas à la piscine.*

Exercice 1.6 Enoncez la négation, la réciproque, la contraposée et l'inverse des énoncés suivants.

1. *Si mon chat est noir, alors tous les chats ne sont pas gris.*
2. *Si* x *est pair, alors* x^2 *est pair.*

3. *Si je suis corse, alors je suis Napoléon.*

Exercice 1.7 Calculez la négation, la réciproque, la contraposée et l'inverse de $(p \vee q) \Rightarrow \neg r$. Quelle est la valeur de vérité de chacune des propositions résultantes dans le cas où $p = 0$, $q = 1$ et $r = 0$?

Exercice 1.8 Quel est l'autre nom de l'inverse de la réciproque ?

Dans la remarque 1.6, nous avons présenté la notion d'équivalence de propositions composées. Elle se note par $\Leftrightarrow$.

L'équivalence

Définition 1.12

Le connecteur équivalence *entre deux propositions* p *et* q, *noté* $p \Leftrightarrow q$ *est la proposition composée suivante*

$$(p \Rightarrow q) \wedge (q \Rightarrow p). \tag{1.5}$$

On lit $p \Leftrightarrow q$ *comme* p *si et seulement si* q. *On dit alors que* p *est une* condition nécessaire et suffisante *de* q, *lorsque la proposition* $p \Leftrightarrow q$ *est vraie.*

Remarque 1.8 Lorsque la proposition $p \Leftrightarrow q$ est vraie, on dit que q est un *synonyme* de p.

Exemple 1.11 *L'entier* n *est pair si et seulement si* $n + 1$ *est impair.* Pour établir un tel résultat, nous devons établir les deux implications suivantes

1. *Si l'entier* n *est pair alors* $n + 1$ *est impair.*
2. *L'entier* n *est pair seulement si* $n + 1$ *est impair.*

Supposons n pair. Il existe $k \in \mathbb{N}$ tel que $n = 2k$. Ainsi $n + 1 = 2k + 1$, ce qui en fait un nombre impair.

Supposons à présent $n + 1$ impair. Il existe $k \in \mathbb{N}$ tel que $n + 1 = 2k + 1$. Dès lors $n = 2k$, ce qui en fait un nombre pair.

Exercice 1.9 Les propriétés suivantes sont elles vraies ?

1. Soit $x \in \mathbb{R} : x < 4 \Leftrightarrow x < 5$.
2. Soit la famille Tantmieux composée de Alice, 42 ans, Bob, 45 ans, Ada, 18 ans, Steve 15 ans et Pascal 12 ans.
   - L'âge de Ada plus l'âge de Steve et de Pascal est égal à l'âge de Bob, si et seulement si Alice a 42 ans ?
   - Bob est moins âgé qu'Alice si et seulement si Ada est la plus jeune.

Disjonction exclusive

Enfin, un dernier connecteur est présenté dans la définition suivante.

Définition 1.13

Le connecteur appelé disjonction exclusive *entre* p *et* q, *noté* $p \oplus q$ *est équivalent à la formule suivante*

$$(p \wedge \neg q) \vee (\neg p \wedge q). \tag{1.6}$$

Remarque 1.9 Le symbole $\oplus$ se lit *ou exclusif.*

La formule $p \oplus q$ est vraie lorsque p *ou* q *exclusivement* est vrai. Si les deux propositions sont vérifiées, alors $p \oplus q$ est fausse.

Exemple 1.12 *Soit* x *un nombre naturel, alors* x *est pair ou impair.* est un énoncé exprimé en français où le ou utilisé est un ou exclusif.

1.3 Tautologie

On peut démontrer que la formule $(p \Leftrightarrow q) \Leftrightarrow \neg(p \oplus q)$ est toujours vraie et ce quelles que soient les valeurs de vérité des variables propositionnelles p et q. Il s'agit d'une tautologie.

Définition 1.14

Une tautologie *est une formule* F *dont la valeur de vérité est vrai quelle que soit la valeur attribuée aux variables propositionnelles qui la composent.*

On note que F est une tautologie par $\vdash F$. Cette notation indique donc que la formule F est toujours vraie, quelque soit la valeur des différentes variables propositionnelles qui la composent. Cette notation est importante. Remarquons les résultats suivants concernant l'opérateur $\wedge$.

Propriété 1.2

Soit la variable propositionnelle p. *Dès lors la proposition*

$$p \wedge 0 \Leftrightarrow 0$$

est toujours vraie, tout comme

$$p \wedge 1 \Leftrightarrow p$$
$$p \wedge \neg p \Leftrightarrow 0$$

le sont également.

Nous avons donc les tautologies suivantes

$$\vdash p \wedge 0 \Leftrightarrow 0 \tag{1.7}$$
$$\vdash p \wedge 1 \Leftrightarrow p \tag{1.8}$$
$$\vdash p \wedge \neg p \Leftrightarrow 0. \tag{1.9}$$

Concernant l'opération $\vee$, nous avons le résultat suivant.

Propriété 1.3

L'opérateur $\vee$ respecte les tautologies suivantes.

$$\vdash p \vee 0 \Leftrightarrow p \tag{1.10}$$
$$\vdash p \vee 1 \Leftrightarrow 1 \tag{1.11}$$
$$\vdash p \vee \neg p \Leftrightarrow 1. \tag{1.12}$$

Les propositions énoncées dans les propriétés 1.2 et 1.3 peuvent être établies en considérant les tables de vérité.

Exemple 1.13 Considérons la proposition suivante *Ma peau est de couleur verte et j'ai les cheveux rouges ou il n'est pas vrai de dire que tous les chats sont gris.* Posons

- p *Ma peau est de couleur verte*,
- q *j'ai les cheveux rouges*,
- r *tous les chats sont gris*.

Dès lors, la proposition à considérer se résume à

$$p \wedge q \vee \neg r \tag{1.13}$$

Comme p est faux, nous avons $p \wedge q$ qui vaut 0 quelle que soit la valeur de vérité de q. Ainsi la proposition de départ se résume simplement à $\neg r$. Or r a une valeur de vérité de 0. Dès lors la proposition de départ est vrai.

Remarque 1.10 Les propositions énoncées dans les propriétés 1.2 et 1.3 sont des propositions exprimées grâce à la logique du premier ordre. Le symbole $\vdash$ est utilisé pour indiquer que ces propositions sont toujours vraies.

Dans la section suivante, nous présentons les propriétés des connecteurs fondamentaux.

1.4 Propriétés des connecteurs fondamentaux

Les équivalences suivantes peuvent se démontrer en développant les tables de vérité.

Propriété 1.4

Les connecteurs $\wedge$ et $\vee$ vérifient la propriété de commutativité :

$$\vdash p \wedge q \Leftrightarrow q \wedge p \tag{1.14}$$
$$\vdash p \vee q \Leftrightarrow q \vee p \tag{1.15}$$

Les connecteurs $\wedge$ et $\vee$ vérifient la propriété d'associativité :

$$\vdash p \wedge (q \wedge r) \Leftrightarrow (p \wedge q) \wedge r \tag{1.16}$$
$$\vdash p \vee (q \vee r) \Leftrightarrow (p \vee q) \vee r \tag{1.17}$$

Les connecteurs $\wedge$ et $\vee$ vérifient chacun la propriété de distributivité sur l'autre :

$$\vdash p \wedge (q \vee r) \Leftrightarrow (p \wedge q) \vee (p \wedge r) \tag{1.18}$$
$$\vdash p \vee (q \wedge r) \Leftrightarrow (p \vee q) \wedge (p \vee r) \tag{1.19}$$

Les connecteurs $\wedge$ et $\vee$ vérifient la propriété d'idempotence :

$$\vdash p \wedge p \Leftrightarrow p \tag{1.20}$$
$$\vdash p \vee p \Leftrightarrow p \tag{1.21}$$

Démonstration : Nous établissons la tautologie (1.20). Pour cela, nous comparons les tables de vérité. Nous avons

p	p	$p \wedge p$
1	1	1
0	0	0

dont la dernière colonne, comparée à p est bien équivalente. ■

Remarque 1.11 On remarque que seule p intervient comme variable propositionnelle dans $p \wedge p$. Dès lors, le nombre de combinaisons possibles pour les valeurs de vérité n'est que de deux dans la table de vérité précédente.

Propriété 1.5

Le connecteur $\neg$ respecte le principe de double négation, soit

$$\vdash \neg\neg p \Leftrightarrow p \tag{1.22}$$

Propriété 1.6 (*Loi de De Morgan*)

La loi de De Morgan reprend deux tautologies

$$\vdash \neg(p \wedge q) \Leftrightarrow \neg p \vee \neg q \tag{1.23}$$
$$\vdash \neg(p \vee q) \Leftrightarrow \neg p \wedge \neg q \tag{1.24}$$

Démonstration : Nous établissons (1.23) en construisant les tables de vérité.

Nous avons

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

et

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

Ces deux tables de vérité sont bien les mêmes (première, deuxième et dernières colonnes identiques). ■

Exemple 1.14 En logique du premier ordre, lorsque vous niez une proposition telle que *Je pars à la mer ou à la montagne*, cela revient à dire *Je ne pars ni à la mer, ni à la montagne*. De la même manière, si vous niez *Je pars à la mer et je pars en voiture*, cela revient à dire *Je ne pars pas à la mer ou je ne pars pas en voiture*. Dès lors, vous pouvez partir à la mer mais alors ce ne sera pas en voiture.

Exemple 1.15 Dans la remarque 1.6, nous avons indiqué qu'une implication a toujours la même valeur de vérité que sa contraposée. Etablissons ce résultat sans utiliser les tables de vérité mais en utilisant à bon escient les propriétés que nous venons d'énoncer.

Nous devons établir que

$$\vdash (p \Rightarrow q) \Leftrightarrow (\neg q \Rightarrow \neg p). \tag{1.25}$$

Nous avons

$$\vdash (p \Rightarrow q) \Leftrightarrow \neg(p \wedge \neg q) \quad \text{par la proposition 1.1} \tag{1.26}$$
$$\Leftrightarrow \neg(\neg q \wedge p) \quad \text{par (1.14)} \tag{1.27}$$
$$\Leftrightarrow \neg q \Rightarrow \neg p \quad \text{par la proposition 1.1} \tag{1.28}$$

Pour terminer ce chapitre, nous énonçons quelques propriétés d'un connecteur qui n'est pas fondamental, le connecteur $\oplus$. Nous avons la propriété suivante.

Propriété 1.7

Soit p, q *et* r *trois variables propositionnelles, alors*

$$\vdash p \oplus p \Leftrightarrow 0 \tag{1.29}$$
$$\vdash p \oplus \neg p \Leftrightarrow 1 \tag{1.30}$$
$$\vdash p \oplus 0 \Leftrightarrow p \tag{1.31}$$
$$\vdash p \oplus 1 \Leftrightarrow \neg p \tag{1.32}$$
$$\vdash p \oplus (q \oplus r) \Leftrightarrow ((p \oplus q) \oplus r) \tag{1.33}$$
$$\vdash p \oplus q \Leftrightarrow (q \oplus p) \tag{1.34}$$
$$\vdash (p \oplus q) \oplus q \Leftrightarrow p \tag{1.35}$$

La démonstration de cette proposition est laissée à titre d'exercice. Remarquons la tautologie (1.35), celle-ci pourrait être généralisée. Nous avons alors

$$\vdash ((p \oplus q) \oplus q) \oplus q \Leftrightarrow (p \oplus q) \qquad \text{par (1.35)} \tag{1.36}$$
$$\vdash (((p \oplus q) \oplus q) \oplus q) \oplus q \Leftrightarrow p \tag{1.37}$$

en appliquant deux fois (1.35) cette fois, et ainsi de suite. Dès lors, seule la parité du nombre d'occurrences d'une variable propositionnelle dans une formule où n'intervient que l'opérateur $\oplus$, détermine le résultat.

Exemple 1.16 La formule

$$p \oplus p \oplus q \oplus r \oplus r \tag{1.38}$$

est équivalente à q.

Notons enfin la remarque suivante.

Remarque 1.12 Nous venons de voir comment connecter des variables propositionnelles à l'aide des connecteurs (fondamentaux ou non). De la même manière, nous pouvons connecter les prédicats. Considérons l'exemple suivant, il se base sur les prédicats $p(x)$ et $q(x)$ tels que définis dans l'exemple 1.3.

Le prédicat $p(x) \vee q(x)$ est vrai pour x défini comme le *Père Noël*. Le prédicat $p(x) \wedge \neg q(x)$ est également vrai. Par contre,

$$\neg(\neg(\neg p(x)) \vee \neg q(x)) \tag{1.39}$$

est faux pour x défini comme le *Père Noël* mais est vrai si x est *Batman*.

1.5 Consistance d'un ensemble de propositions

On définit la *consistance* d'un ensemble de propositions de la manière suivante.

Définition 1.15

Un ensemble de propositions est dit consistant lorsqu'il est possible que toutes les propositions soient vraies simultanément. On précise alors les conditions sous lesquelles ces propositions sont vraies.

Exemple 1.17 Soit les propositions suivantes
- *Marcel aime le chocolat.*
- *Marcel aime le sucre ou le chocolat.*

Posons $\mathtt{p}$ *Marcel aime le chocolat* et $\mathtt{q}$ la proposition *Marcel aime le sucre*. Cet ensemble de propositions se note alors de manière symbolique par

$$\begin{array}{c} \mathtt{p} \\ \mathtt{q} \vee \mathtt{p} \end{array}$$

Pour que ces deux propositions soient vraies simultanément, il suffit que $\mathtt{p}$ soit vraie.

Etablir la consistance d'un ensemble de propositions composées se réalise en déterminant les valeurs de vérité des variables propositionnelles les composant de telle manière que les propositions composées soient toutes simultanément vraies.

Exemple 1.18 Nous considérons la piscine communale qui doit respecter les règles de fonctionnement suivantes.

1. La piscine accueille plusieurs nageurs si et seulement si elle est ouverte.
2. Si la piscine est ouverte, on peut acheter un ticket d'entrée.
3. On achète un ticket d'entrée ou la piscine est en travaux.
4. Si la piscine n'est pas en travaux, la piscine accueille plusieurs nageurs et on ne peut pas acheter de ticket d'entrée.

Afin de vérifier la consistance de cet ensemble de propositions composées, posons les variables propositionnelles suivantes

$\mathtt{p}$ *La piscine accueille plusieurs nageurs* ,
$\mathtt{q}$ *la piscine est ouverte*,
$\mathtt{r}$ *on peut acheter un ticket d'entrée*,
$\mathtt{s}$ *la piscine est en travaux.*

L'ensemble des règles se représente alors comme

1. $\mathtt{p} \Leftrightarrow \mathtt{q}$,
2. $\mathtt{q} \Rightarrow \mathtt{r}$,
3. $\mathtt{r} \vee \mathtt{s}$,
4. $\neg\mathtt{s} \Rightarrow (\mathtt{p} \wedge \neg\mathtt{r})$.

Cet ensemble est consistant uniquement dans les cas suivants

p	q	r	s
0	0	0	1
0	0	1	1
1	1	1	1

Qu'en penser ?

Chapitre 2

Logique des prédicats

Nous avons présenté dans la définition 1.2, la notion de prédicat. Dans ce chapitre, nous présentons la logique associée aux prédicats. Un prédicat utilise la notion de paramètre dont la valeur peut être précisée grâce aux quantificateurs. Nous présentons la notion de quantificateur dans la section 2.1. Leurs propriétés sont explicitées dans la section 2.2.

2.1 Les quantificateurs

Nous définissons le quantificateur universel et le quantificateur existentiel. Un quantificateur permet de préciser l'ensemble des valeurs possibles pour les différents paramètres.

Définition 2.1

Soit $p(x)$ *un prédicat en la variable* x*. La proposition*

$$\forall x : p(x) \tag{2.1}$$

est vraie si et seulement si toute valeur de x *rend vrai le prédicat* $p(x)$*. Le symbole* $\forall$ *s'appelle le* quantificateur universel.

L'équation (2.1) se lit *Pour tout* x *tel que* $p(x)$.

Remarque 2.1 Une proposition telle que formulée par (2.1) est fausse dès qu'il existe un seul x pour lequel la proposition $p(x)$ ne peut être vérifiée.

Exemple 2.1 Nous avons la proposition suivante

$$\forall n \in \mathbb{Z} : (n \geq 2) \Rightarrow (n^2 \geq 4). \tag{2.2}$$

Notons d'abord la structure de cette proposition. On trouve d'abord la présence du quantificateur $\forall$ qui porte sur le paramètre n. Ensuite, nous avons le prédicat

$$(n \geq 2) \Rightarrow (n^2 \geq 4). \tag{2.3}$$

Il s'agit d'un prédicat *composé* puisqu'il fait intervenir deux prédicats, soit

— $p(n)$ défini comme $n \geq 2$,
— $q(n)$ défini comme $n^2 \geq 4$.

Ces deux prédicats sont reliés par une implication.

Exercice 2.1 Traduisez en langage mathématique les propositions suivantes

1. *Le carré de tout nombre réel est positif.*
2. *Pour tout nombre impair* x*, le nombre* $x^2 + 1$ *est également impair.*
3. *Le produit de deux nombres rationnels et la somme de deux nombres rationnels sont rationnels.*

Définition 2.2

Soit $p(x)$ *un prédicat en la variable* x*. La proposition*

$$\exists x : p(x) \tag{2.4}$$

est vraie si et seulement si il existe au moins une valeur de x *qui rend vraie la proposition* $p(x)$*. Le symbole* $\exists$ *s'appelle le* quantificateur existentiel.

La proposition (2.4) se lit *Il existe* x *tel que* $p(x)$.

Remarque 2.2 Une proposition formulée comme en (2.4) est fausse dès qu'on a établi que la proposition $p(x)$ était fausse pour toute valeur x possible.

Ces quantificateurs ont une priorité maximale. Ils portent donc sur l'ensemble du prédicat.

Exercice 2.2 Traduisez en langage mathématique les propositions suivantes.

1. *Il existe un entier tel que son carré vaut* 4.
2. *Il existe un réel plus petit que* -1 *dont le carré est* $1/4$.

Les objets x sur lesquels portent les quantificateurs ne sont pas quelconques mais appartiennent à des ensembles bien particuliers. On notera par exemple

$$\forall x \in A : p(x), \tag{2.5}$$

qui se lit *pour tout élément* x *de* A *tel que* $p(x)$. De la même manière, on note

$$\exists x \in B : p(x) \tag{2.6}$$

qui signifie *il existe un élément* x *de* B *tel que* $p(x)$. Il s'agit d'une écriture abrégée de

$$\forall x : ((x \in A) \Rightarrow p(x)) \tag{2.7}$$
$$\exists x : ((x \in B) \wedge p(x)), \tag{2.8}$$

respectivement. En effet, reprenons (2.5). Ce prédicat est équivalent à (2.7) puisque (2.5) indique que si $x \in A$, alors, $p(x)$ est observée. Le fait d'appartenir à A est une condition suffisante pour observer $p(x)$.

Exemple 2.2 L'assertion

Les naturels positifs plus petits que 4, sont plus petits que 7.

peut s'écrire

$$\forall x \in \mathbb{N} : (x \leq 4) \Rightarrow (x \leq 7), \tag{2.9}$$

ou encore

$$\forall x \in \{0, 1, 2, 3, 4\} : (x \leq 7). \tag{2.10}$$

En plus des quantificateurs $\forall$ et $\exists$, on peut également parler du quantificateur $\exists!$ qui signifie *il existe un et un seul*. Ecrire la formule

$$\exists! x : p(x) \tag{2.11}$$

est équivalent à écrire

$$\exists x : p(x) \wedge (\forall y \forall z : (p(y) \wedge p(z) \Rightarrow y = z)). \tag{2.12}$$

La première partie $\exists x : p(x)$ affirme l'existence d'un tel élément qui vérifie $p(x)$. La seconde affirme l'unicité de l'élément. Leur conjonction affirme qu'il existe un et un seul élément.

2.2 Propriétés des quantificateurs

Dans cette section, nous envisageons les prédicats où sont liés quantificateurs et connecteurs. Deux remarques préalables s'imposent. Premièrement, dans la formule

$$\forall x : p(x) \tag{2.13}$$

x est une variable. Dès lors, la formule

$$\forall y : p(y) \tag{2.14}$$

est rigoureusement équivalente à la formule (2.13). Nous avons donc

$$\vdash \forall x : p(x) \Leftrightarrow \forall y : p(y). \tag{2.15}$$

Ensuite, remarquons qu'on peut tout à fait écrire des formules où les quantificateurs se présentent en cascade, soit

$$\forall x \, \exists y \, \exists z : p(x, y, z). \tag{2.16}$$

Dans ce prédicat, la valeur de y peut dépendre de x et celle de z de x et de y.

Exemple 2.3 Considérons les différents cas.

1. $\forall x \in \mathbb{R}\, \exists y \in \mathbb{R} : y > x$.
2. Par définition, la limite

$$\lim_{x \to 2} \frac{x^2 - 4}{x^2 - 3x + 2} = 4 \tag{2.17}$$

peut s'écrire

$$\forall \epsilon > 0 : \exists \delta > 0 : \|x - 2\| < \delta \Rightarrow \left\| \frac{x^2 - 4}{x^2 - 3x + 2} - 4 \right\| \leq \epsilon. \tag{2.18}$$

3. La formule $\forall x : (x + y = x)$ reste un prédicat. En effet, x est lié par le quantificateur universel, mais y reste un paramètre de la proposition.
4. La quantité l est la limite de la suite $a_1, a_2, \ldots$ si

$$\forall \epsilon > 0 : \exists N \geq 1 : \forall n \geq N : -\epsilon < a_n - l < \epsilon, \tag{2.19}$$

qui se lit *Quel que soit le* ϵ *que l'on me donne, je peux trouver un nombre naturel* N *tel que à partir de l'indice* n *(pour autant qu'il soit plus grand que* N*), la différence entre le terme* a_n *de la suite et la quantité* l *est au maximum de* ϵ*.*

Dans cet énoncé, il faut bien comprendre que la quantité N dépend de la valeur de ϵ.

L'exemple précédent insiste à nouveau sur le fait que l'ordre dans lequel se présentent les quantificateurs, est important. Les deux formules suivantes ne sont pas équivalentes :

$$\forall x\, \exists y : p(x, y), \tag{2.20}$$

$$\exists y\, \forall x : p(x, y). \tag{2.21}$$

Dans la première proposition, y peut dépendre de la valeur de x, alors que dans la seconde, ce n'est pas le cas.

Exemple 2.4 Considérons les deux propositions suivantes :

$$\forall x \in \mathbb{N} : \exists y \in \mathbb{N} : x - y = 0, \tag{2.22}$$

$$\exists y \in \mathbb{N} : \forall x \in \mathbb{N} : x - y = 0. \tag{2.23}$$

La seconde proposition est fausse. Elle affirme qu'il existe un naturel tel que retranché à n'importe quel autre naturel, on obtient 0.

Exercice 2.3 Traduisez en langage mathématique : *Pour toute paire de nombres naturels, il y a un nombre naturel plus grand que chacun d'eux.*

Dans la proposition suivante, on décrit la négation des formules où des quantificateurs sont présents.

Propriété 2.1

Les règles fondamentales de négation des formules quantifiées sont

$$\vdash \neg(\forall x : p(x)) \Leftrightarrow \exists x : \neg p(x) \tag{2.24}$$
$$\vdash \neg(\exists x : p(x)) \Leftrightarrow \forall x : \neg p(x) \tag{2.25}$$

Exemple 2.5 Dès lors, on obtient pour la formule suivante

$$\neg(\forall x \in A : \exists y \in B : \exists z \in C\, \forall t \in B : p(x, y, z, t)) \tag{2.26}$$
$$\Leftrightarrow$$
$$\exists x \in A : \forall y \in B : \forall z \in C : \exists t \in B : \neg p(x, y, z, t) \tag{2.27}$$

Exercice 2.4 Nier les propositions suivantes

1. Pour tout $x \in \mathbb{Z} : x^2 \neq 4$.
2. *Il existe un mathématicien qui n'est pas intelligent.*
3. *Il n'existe pas de nombre réel positif plus petit que tous les autres.*

Examinons à présent les propriétés des quantificateurs en relation avec les connecteurs de conjonction et de disjonction. Nous avons la propriété suivante.

Propriété 2.2

Soit $p(x), q(x)$ deux prédicats, on a les tautologies suivantes

$$\vdash (\forall x : p(x) \wedge q(x)) \Leftrightarrow (\forall x\, p(x)) \wedge (\forall x : q(x)) \tag{2.28}$$
$$\vdash (\exists x : p(x) \vee q(x)) \Leftrightarrow (\exists x : p(x)) \vee (\exists x : q(x)). \tag{2.29}$$

Attention, les formules suivantes ne sont pas des tautologies

$$\forall x : (p(x) \vee q(x)) \Leftrightarrow (\forall x : p(x)) \vee (\forall x : q(x)) \tag{2.30}$$
$$\exists x : (p(x) \wedge q(x)) \Leftrightarrow (\exists x : p(x)) \wedge (\exists x : q(x)). \tag{2.31}$$

Toute la difficulté réside dans la portée du quantificateur par rapport à la proposition. Considérons en effet les exemples suivants.

Exemple 2.6 1. Soit la proposition suivante

$$\forall x \in \mathbb{N} : (x < 3) \vee (x \geq 3). \tag{2.32}$$

Celle-ci n'est clairement pas équivalente à

$$(\forall x \in \mathbb{N} : x < 3) \vee (\forall x \in \mathbb{N} : x \geq 3). \tag{2.33}$$

En effet, cette dernière proposition est fausse. Tous les naturels ne sont pas strictement plus petits que 3, donc $(\forall x \in \mathbb{N} : x < 3)$ est faux. De la même manière, tous les naturels ne sont pas plus grand que 3, donc $(\forall x \in \mathbb{N} : x \geq 3)$ est faux. Ainsi, la disjonction de ces deux propositions est fausse.

2. De la même manière, nous avons

$$\exists x \in \mathbb{N} : (x < 1) \wedge (x > 2) \tag{2.34}$$

qui n'est pas équivalente à

$$(\exists x \in \mathbb{N} : x < 1) \wedge (\exists x \in \mathbb{N} : x > 2). \tag{2.35}$$

En effet, la proposition (2.34) est fausse, alors que la proposition (2.35) est vraie.

Chapitre 3

Calcul booléen

Le calcul booléen est un modèle mathématique permettant de représenter des systèmes logiques, binaires et de représenter des opérations dans ces systèmes.

Dans un premier temps, on développe les axiomes du calcul booléen. Ceux-ci peuvent être aisément mis en relation avec les propriétés mises en évidence dans la logique des propositions. A partir de ces axiomes, on peut démontrer des propriétés fondamentales du calcul booléen. Celles-ci sont présentées dans la section 3.2. Les formes normales conjonctive et disjonctive sont définies dans la section 3.3. Toute expression booléenne peut donc se présenter sous une de ces deux formes. Grâce au diagramme de Karnaugh présenté dans la section 3.4, toute forme normale disjonctive pourra être aisément simplifiée, pour autant qu'on travaille avec moins de quatre variables. Enfin, dans la section 3.5, on présente un cas pratique d'utilisation du calcul booléen et la représentation graphique utilisée en électronique.

3.1 Axiomes du calcul booléen

L'algèbre de Boole de base est composée de l'ensemble d'éléments $\mathcal{B} = \{0, 1\}$, sur lesquels opèrent les opérations $\times$, $+$ et $\bar{\cdot}$ de la manière suivante.

Définition 3.1

Soit a, b *et* c *trois éléments issus de* $\mathcal{B}$*. Les axiomes qui permettent de définir l'algèbre de Boole sont les suivants*

$$a \times 1 = a \tag{3.1}$$
$$a + 0 = a \tag{3.2}$$
$$a \times b = b \times a \tag{3.3}$$
$$a + b = b + a \tag{3.4}$$
$$a \times (b + c) = (a \times b) + (a \times c) \tag{3.5}$$

$$a + (b \times c) = (a + b) \times (a + c) \tag{3.6}$$
$$a \times \overline{a} = 0 \tag{3.7}$$
$$a + \overline{a} = 1 \tag{3.8}$$

Le calcul booléen est une autre méthode pour représenter la logique des propositions. Un élément a est alors interprété comme une variable propositionnel. On remarque alors rapidement la correspondance qui existe entre l'opération $\times$ et l'opérateur logique $\wedge$; entre l'opération $+$ et l'opérateur logique $\vee$ et entre l'opérateur $\overline{\cdot}$ et l'opérateur logique $\neg$. Tous ces axiomes peuvent donc être traduits en propriété des opérateurs logiques $\wedge$, $\vee$ et $\neg$.

3.2 Propriétés du calcul booléen

Nous avons alors les résultats suivants.

Propriété 3.1

Soit a, b *et* c *trois éléments issus de* $\mathcal{B}$*. Nous avons alors*

$$a \times a = a \tag{3.9}$$
$$a + a = a \tag{3.10}$$
$$a \times 0 = 0 \tag{3.11}$$
$$a + 1 = 1 \tag{3.12}$$
$$\overline{\overline{a}} = a \tag{3.13}$$
$$a + (b + c) = (a + b) + c \tag{3.14}$$
$$a \times (b \times c) = (a \times b) \times c \tag{3.15}$$
$$\overline{a \times b} = \overline{a} + \overline{b} \tag{3.16}$$
$$\overline{a + b} = \overline{a} \times \overline{b} \tag{3.17}$$
$$\overline{0} = 1 \tag{3.18}$$
$$\overline{1} = 0 \tag{3.19}$$
$$a \times (a + b) = a \tag{3.20}$$
$$a + (a \times b) = a \tag{3.21}$$
$$\overline{a} \times (a \times b) = 0 \tag{3.22}$$
$$\overline{a} + (a + b) = 1 \tag{3.23}$$
$$a \times (\overline{a} \times \overline{b}) = 0 \tag{3.24}$$
$$a + (\overline{a} + \overline{b}) = 1 \tag{3.25}$$

Démonstration : Démontrons l'équation (3.9). Nous avons par l'axiome (3.1),

$$\begin{aligned}
& a \times 1 = a && && (3.26)\\
\Leftrightarrow\ & a \times (a + \overline{a}) = a && \text{par (3.8)} && (3.27)\\
\Leftrightarrow\ & a \times a + a \times \overline{a} = a && \text{par (3.5)} && (3.28)\\
\Leftrightarrow\ & a \times a + 0 = a && \text{par (3.7)} && (3.29)\\
\Leftrightarrow\ & a \times a = a && \text{par (3.2)} && (3.30)
\end{aligned}$$

ce qu'il fallait démontrer. Démontrons à présent l'équation (3.20). Nous avons

$$\begin{aligned}
a \times (a + b) &= a \times a + a \times b && \text{par (3.5)} && (3.31)\\
&= a + a \times b && \text{par (3.9)} && (3.32)\\
&= a \times (1 + b) && \text{par (3.5)} && (3.33)\\
&= a \times 1 && \text{par (3.12)} && (3.34)\\
&= a && \text{par (3.1)} && (3.35)
\end{aligned}$$

■

3.3 Forme normale conjonctive et disjonctive

Nous travaillons avec des fonctions d'un ou plusieurs éléments de $\mathcal{B}$, soit une application définie comme

$$f : \mathcal{B}^n \to \mathcal{B} : (a_1, a_2, \ldots, a_n) \to f(a_1, a_2, \ldots, a_n). \tag{3.36}$$

Cette fonction peut être définie par sa table de vérité, par une formule, par un circuit de portes (voir la section 3.5).

Toute fonction booléenne peut être exprimée sous une forme équivalente appelée *forme normale disjonctive*. Ceci est possible parce que les trois opérateurs $\overline{\cdot}$, $+$ et $\times$ forment un ensemble d'opérateurs dit *fonctionnellement complet*. On pourrait d'ailleurs démontrer que les ensembles $\overline{\cdot}$, $+$ et $\overline{\cdot}$, $\times$ sont eux-mêmes suffisants pour exprimer toute fonction booléenne. En effet, nous avons les propriétés (3.16) et (3.17) respectivement qui permettent de transformer toute opération $\times$ en opération $+$ et toute opération $+$ en opération $\times$ respectivement.

Remarque 3.1 Dans le cadre de ce cours, on ne démontre pas formellement que l'ensemble d'opérateurs $\{\overline{\cdot}, +, \times\}$ est fonctionnellement complet. Cependant, il est intéressant d'établir que l'ensemble $\{\overline{\cdot}, \oplus\}$ avec l'opération $\oplus$ définie comme

$$a \oplus b \stackrel{\text{def}}{=} (a \times \overline{b}) + (\overline{a} \times b) \tag{3.37}$$

n'est pas fonctionnellement complet.

A présent, pour alléger nos écritures, l'opération $\times$ ne sera plus représentée dans nos équations. Ainsi l'écriture $a \times b$ sera simplement notée par ab. De plus, nous faisons également l'hypothèse que cette opération est prioritaire sur l'opération $+$. Ainsi $ab + cd$ signifie $(ab) + (cd)$.

La définition d'une *forme normale disjonctive* est construite à partir des définitions suivantes.

Définition 3.2

Un atome *est une variable propositionnelle en logique, un élément en calcul booléen.*

Un littéral *est un atome ou la négation d'un atome.*

Une clause conjonctive *est une conjonction de littéraux, soit*

$$l_1 l_2 \dots l_n \tag{3.38}$$

où l_i *est un littéral, pour tout* i.

Une clause disjonctive *est une disjonction de littéraux, soit*

$$l_1 + l_2 + \dots + l_n \tag{3.39}$$

où l_i *est un littéral, pour tout* i.

Remarque 3.2 Notons donc la forme simplifiée de (3.38) où l'opérateur $\times$ n'est plus explicitement utilisé.

Définition 3.3

Une formule f *est sous* forme normale disjonctive *lorsqu'elle est exprimée sous la forme d'une disjonction de clauses conjonctives*

$$f = m_1 + m_2 + \dots + m_k, \tag{3.40}$$

avec m_i *clause conjonctive pour tout* i.

Nous avons également que toute fonction booléenne peut être exprimée sous *forme normale conjonctive*. En voici la définition.

Définition 3.4

Une formule f *est sous* forme normale conjonctive *lorsqu'elle est exprimée sous la forme d'une conjonction de clauses disjonctives*

$$f = m_1 m_2 \dots m_k, \tag{3.41}$$

avec m_i clause disjonctive *pour tout* i*, soit une disjonction de littéraux, donc*

$$l_1 + l_2 + \dots + l_n \tag{3.42}$$

où l_i *est un littéral, pour tout* i.

Comme toute fonction booléenne peut s'écrire sous une forme normale, on peut dire que l'ensemble des opérateurs $\{+, ., \bar{\cdot}\}$ est un *ensemble complet d'opérateurs*.

On parle de *normaliser* une formule f lorsqu'il s'agit de faire apparaître une telle disjonction ou conjonction de clauses. Pour ce faire, une approche systématique existe. En logique propositionnelle, il s'agit

1. d'éliminer tous les connecteurs non-fondamentaux, soit $\Leftrightarrow, \oplus, \Rightarrow$.
2. De faire porter le connecteur unaire $\neg$ directement sur les variables. Utiliser pour cela les lois de De Morgan (proposition 1.6) et le principe de double négation (1.22).
3. Utiliser la distributivité de de $\wedge$ sur $\vee$ (soit (1.18)) pour une forme disjonctive et la distributivité $\vee$ sur $\wedge$ (1.19) pour obtenir une forme normale conjonctive

Ainsi, une telle démarche peut être également appliquée dans le formalisme propre au calcul booléen. Soit une fonction booléenne f, il faut

1. faire porter le connecteur unaire $\bar{\cdot}$ directement sur les variables. Utiliser pour cela les propriétés (3.16) et (3.17) et le principe de double négation (3.13).
2. Utiliser la distributivité de $\times$ sur $+$ (soit (3.5)) pour obtenir une forme normale disjonctive et la distributivité de $+$ sur $\times$ (3.6) pour obtenir une forme normale conjonctive.

Exemple 3.1 Soit la fonction

$$\overline{\overline{\mathsf{p}}\mathsf{q}}(\mathsf{p}+\mathsf{q}) \tag{3.43}$$

une forme normale disjonctive s'obtient en réalisant les étapes précédemment citées, ce qui donne

$$\begin{aligned}
\overline{\overline{\mathsf{p}}\mathsf{q}}(\mathsf{p}+\mathsf{q}) &= (\overline{\overline{\mathsf{p}}}+\overline{\mathsf{q}})(\mathsf{p}+\mathsf{q}) && (3.44)\\
&= (\mathsf{p}+\overline{\mathsf{q}})(\mathsf{p}+\mathsf{q}) && (3.45)\\
&= (\mathsf{p}+\overline{\mathsf{q}})\mathsf{p}+(\mathsf{p}+\overline{\mathsf{q}})\mathsf{q} && (3.46)\\
&= \mathsf{p}\mathsf{p}+\overline{\mathsf{q}}\mathsf{p}+\mathsf{p}\mathsf{q}+\overline{\mathsf{q}}\mathsf{q} && (3.47)\\
&= \mathsf{p}+\overline{\mathsf{q}}\mathsf{p}+\mathsf{p}\mathsf{q}+0 && (3.48)\\
&= \mathsf{p}+\overline{\mathsf{q}}\mathsf{p}+\mathsf{p}\mathsf{q} && (3.49)
\end{aligned}$$

en notant que (3.45) est une forme normale conjonctive.

Lorsqu'on travaille directement avec la table de vérité de la fonction booléenne f, il suffit de repérer les valeurs des variables pour lesquelles la fonction f vaut 1 et de les reprendre comme termes de la forme normale disjonctive. Considérons l'exemple suivant.

Exemple 3.2 Soit la fonction $\mathsf{f}(\mathsf{p},\mathsf{q},\mathsf{r})$ dont la table de vérité est

p	q	r	f
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	0

Cette table donne la fonction

$$\overline{p}\,\overline{q}r + \overline{p}qr + p\overline{q}\,\overline{r}, \tag{3.50}$$

sous forme normale disjonctive.

3.4 Diagramme de Karnaugh

Le *diagramme de Karnaugh* est une méthode graphique qui permet de simplifier une forme normale disjonctive afin de limiter l'utilisation de symbole. Cette méthode est facile à manipuler tant que la fonction ne comprend que trois ou quatre variables.

Soit une formule comprenant trois variables p, q et r. Le diagramme de Karnaugh est comme une matrice où les colonnes correspondent à toutes les conjonctions possibles de p et q et leurs négations, et les lignes à r et à sa négation. Cela donne le diagramme suivant

	pq	$\overline{p}q$	$\overline{p}\,\overline{q}$	$p\overline{q}$
r				
$\overline{r}$				

Si dans la forme normale disjonctive, la conjonction xyz est présente, on place 1 dans la cellule correspondante dans le diagramme de Karnaugh. Par exemple, soit la formule

$$pqr + \overline{p}q\overline{r}, \tag{3.51}$$

on obtient alors le diagramme de Karnaugh suivant

	pq	$\overline{p}q$	$\overline{p}\,\overline{q}$	$p\overline{q}$
r	1			
$\overline{r}$		1		

Par la suite, on groupe les cellules contiguës remplies de 1. Par exemple, la formule suivante

$$\overline{p}qr + \overline{p}\,\overline{q}r, \tag{3.52}$$

qui donne le diagramme de Karnaugh

	pq	$\overline{p}q$	$\overline{p}\,\overline{q}$	$p\overline{q}$
r		1	1	
$\overline{r}$				

est telle qu'on y repère deux cellules contiguës remplies de 1 ; celles-ci sont grisées, soit

	pq	$\overline{p}q$	$\overline{p}\,\overline{q}$	$p\overline{q}$
r		1	1	
$\overline{r}$				

Dès lors, on peut simplifier la formule de la manière suivante

$$\overline{p}qr + \overline{p}\,\overline{q}r = \overline{p}r \tag{3.53}$$

puisque

$$\begin{aligned} \overline{p}qr + \overline{p}\,\overline{q}r &= \overline{p}(q + \overline{q})r && \text{par (3.5)} && (3.54)\\ &= \overline{p}1r && \text{par (3.8)} && (3.55)\\ &= \overline{p}r && \text{par (3.1)} && (3.56) \end{aligned}$$

L'idée est donc de repérer dans le diagramme, tous les couples de cellules contiguës et de les simplifier. Remarquons les exemples suivants.

Exemple 3.3 Soit le diagramme suivant

	pq	$\overline{p}q$	$\overline{p}\,\overline{q}$	$p\overline{q}$
r	1	1		
$\overline{r}$	1	1	1	

correspondant à la formule

$$pqr + \overline{p}\,\overline{q}\,\overline{r} + \overline{p}qr + pq\overline{r} + \overline{p}q\overline{r} \tag{3.57}$$

On remarque la présence de cinq cellules contiguës. L'équation correspondant aux quatre cellules grisées dans

	pq	$\overline{p}q$	$\overline{p}\,\overline{q}$	$p\overline{q}$
r	1	1		
$\overline{r}$	1	1	1	

se simplifie en

$$pqr + \overline{p}qr + pq\overline{r} + \overline{p}q\overline{r} = q \tag{3.58}$$

On peut ensuite, prendre les deux cellules grisées représentées ci-dessous

	pq	$\overline{p}q$	$\overline{p}\,\overline{q}$	$p\overline{q}$
r	1	1		
$\overline{r}$	1	1	1	

En effet, nous avons l'égalité suivante

$$p + p = p. \tag{3.59}$$

Toute cellule grisée peut donc être considérée deux fois.

L'équation correspondant à ces deux cellules se simplifie comme

$$\overline{p}q\overline{r} + \overline{p}\,\overline{q}\,\overline{r} = \overline{p}\,\overline{r}. \tag{3.60}$$

La formule (3.57) peut donc se simplifier ainsi

$$pqr + \overline{p}\,\overline{q}\,\overline{r} + \overline{p}qr + pq\overline{r} + \overline{p}q\overline{r} = q + \overline{p}\,\overline{r}. \tag{3.61}$$

Remarque 3.3 Considérons le cas représenté comme ci-dessous

	pq	$\overline{p}q$	$\overline{p}\,\overline{q}$	$p\overline{q}$
r	1			1
$\overline{r}$				

Celui-ci correspond à la formule

$$pqr + p\overline{q}r \tag{3.62}$$

Les deux cellules marquées à 1 sont bien contiguës. En effet, si nous étiquetons les colonnes en démarrant par $\overline{p}q$ plutôt que pq, nous obtenons la figure

	$\overline{p}q$	$\overline{p}\,\overline{q}$	$p\overline{q}$	pq
r			1	1
$\overline{r}$				

Il ne faut donc pas oublier ce que les colonnes de bord peuvent induire comme simplification.

Nous pouvons également travailler avec des diagrammes à quatre variables. Soit p, q, r, s quatre variables booléennes. Nous avons alors la représentation en diagramme de Karnaugh suivante

	pq	$\overline{p}q$	$\overline{p}\,\overline{q}$	$p\overline{q}$
rs				
$\overline{r}s$				
$\overline{r}\,\overline{s}$				
$r\overline{s}$				

Ainsi considérons l'exemple suivant.

Exemple 3.4 Soit la proposition

$$\overline{p}\,\overline{q}r\overline{s} + \overline{p}\,\overline{q}rs + \overline{p}qr\overline{s} + \overline{p}qrs + p\overline{q}rs + pq\overline{r}\,\overline{s} + pq\overline{r}s + pqr\overline{s}, \tag{3.63}$$

qui donne le diagramme de Karnaugh

	pq	$\overline{p}q$	$\overline{p}\,\overline{q}$	$p\overline{q}$
rs		1	1	1
$\overline{r}s$	1			
$\overline{r}\,\overline{s}$	1			
$r\overline{s}$	1	1	1	

On y recherche les groupes de cellules contiguës et dont la taille est un multiple de 2. Nous avons un groupe de quatre cellules, représenté comme

	pq	$\overline{p}q$	$\overline{p}\,\overline{q}$	$p\overline{q}$
rs		1	1	1
$\overline{r}s$	1			
$\overline{r}\,\overline{s}$	1			
$r\overline{s}$	1	1	1	

La simplification en est la suivante

$$\overline{p}r. \tag{3.64}$$

En considérant successivement les couples de cellules dans les trois figures suivantes

	pq	$\overline{p}q$	$\overline{p}\,\overline{q}$	$p\overline{q}$
rs		1	1	1
$\overline{r}s$	1			
$\overline{r}\,\overline{s}$	1			
$r\overline{s}$	1	1	1	

	pq	$\overline{p}q$	$\overline{p}\,\overline{q}$	$p\overline{q}$
rs		1	1	1
$\overline{r}s$	1			
$\overline{r}\overline{s}$	1			
$r\overline{s}$	1	1	1	

	pq	$\overline{p}q$	$\overline{p}\,\overline{q}$	$p\overline{q}$
rs		1	1	1
$\overline{r}s$	1			
$\overline{r}\overline{s}$	1			
$r\overline{s}$	1	1	1	

nous obtenons alors la formule simplifiée suivante

$$\overline{p}r + pq\overline{r} + \overline{q}rs + qr\overline{s}. \tag{3.65}$$

Par cet exemple, on prend conscience que plusieurs formules simplifiées auraient pu être trouvées. Toutes sont bien sûr équivalentes.

Pour un diagramme pour cinq variables, nous avons la représentation

	pqr	$pq\overline{r}$	$p\overline{q}\,\overline{r}$	$p\overline{q}r$	$\overline{p}\,\overline{q}r$	$\overline{p}q\,\overline{r}$	$\overline{p}q\overline{r}$	$\overline{p}qr$
tu								
$\overline{t}u$								
$\overline{t}\,\overline{u}$								
$t\overline{u}$								

La visualisation devient de plus en plus difficile.

Illustrons à présent l'intérêt des diagrammes de Karnaugh pour concevoir des fonctions booléennes dont la sortie était connue. Supposons qu'on considère une fonction à quatre variables dont la sortie est décrite par le diagramme de Karnaugh suivant.

	pq	$\overline{p}q$	$\overline{p}\,\overline{q}$	$p\overline{q}$
rs			1	1
$\overline{r}s$	1	1		
$\overline{r}\,\overline{s}$			1	
$r\overline{s}$				

On y repère aisément deux groupes de deux cellules, soit

	pq	$\overline{p}q$	$\overline{p}\,\overline{q}$	$p\overline{q}$
rs			1	1
$\overline{r}s$	1	1		
$\overline{r}\,\overline{s}$			1	
$r\overline{s}$				

L'expression est donc la suivante

$$\overline{q}rs + q\overline{r}s + \overline{p}\,\overline{q}\,\overline{r}\,\overline{s}. \tag{3.66}$$

Exercice 3.1 Considérons la fonction f dont la table de vérité est la suivante

p	q	r	f
0	0	0	1
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	0

Donnez son expression et simplifiez la à l'aide d'un diagramme de Karnaugh.

3.5 Applications

Dans cette section, nous analysons deux cas pratiques. Dans le premier, on simplifie l'expression logique qui fixe les conditions de réussite ou d'échec d'un étudiant dans une institution particulière. La seconde application met en évidence une méthode graphique permettant de concevoir des dispositifs binaires, tels que les systèmes électroniques.

Un cas pratique

Considérons les conditions d'échec d'une année d'étude dans une institution particulière. Trois conditions sont importantes, les voici

$a =$ avoir plus de 3 points d'échec,
$b =$ avoir une moyenne générale inférieure à 12,
$c =$ être en règle d'inscription.

Un étudiant ne réussit pas son année, lorsque

— il a plus de 3 points d'échec, une moyenne générale inférieure à 12, et est en règle d'inscription,
— il a plus de 3 points d'échec, une moyenne générale supérieure à 12 et est en règle d'inscription,
— il a moins de 3 points d'échec, une moyenne générale inférieure à 12, et est en règle d'inscription,
— il a plus de 3 points d'échec, une moyenne générale inférieure à 12, et n'est pas en règle d'inscription.

Dans cette institution, on admet effectivement qu'un étudiant qui n'est pas en règle d'inscription mais dont la moyenne générale est supérieure à 12 et qui n'a pas 3 points d'échec, doit pouvoir réussir son année moyennant un délai supplémentaire pour régler sa situation.

En langage mathématique, nous avons donc qu'un étudiant n'a pas réussi son année d'étude lorsque

$$a \times b \times c + a \times \overline{b} \times c + \overline{a} \times b \times c + a \times b \times \overline{c}. \tag{3.67}$$

C'est la condition qui doit être codée pour rendre la procédure de délibération des étudiants automatique. Cette condition peut cependant être simplifiée en utilisant les propriétés de l'algèbre de Boole. Pour continuer, nous reprenons notre écriture simplifiée en notant donc

$$a \times b \overset{\text{noté}}{=} ab. \tag{3.68}$$

Nous avons

$$\begin{aligned}
&abc + a\overline{b}c + \overline{a}bc + ab\overline{c} && && (3.69)\\
&\quad = abc + abc + abc + a\overline{b}c + \overline{a}bc + ab\overline{c} && \text{par (3.10)} && (3.70)\\
&\quad = (abc + a\overline{b}c) + (abc + \overline{a}bc) + (abc + ab\overline{c}) && \text{par (3.4) et (3.14)} && (3.71)\\
&\quad = ac(b + \overline{b}) + (a + \overline{a})bc + ab(c + \overline{c}) && \text{par (3.5) et (3.3)} && (3.72)\\
&\quad = ac\,1 + 1\,bc + ab\,1 && \text{par (3.8)} && (3.73)\\
&\quad = ac + bc + ab && \text{par (3.1)} && (3.74)
\end{aligned}$$

soit une formule simplifiée.

Electronique et circuits logiques

Cette représentation graphique permet des interprétations intuitives. Nous présentons dans les figures 3.1 et 3.2 la représentation graphique des deux connecteurs fondamentaux conjonction et disjonction. On parle de porte `AND` pour la conjonction et de porte `OR` pour la disjonction.

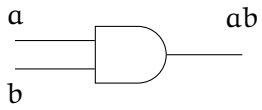


FIGURE 3.1 – Porte `AND`

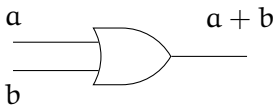


FIGURE 3.2 – Porte `OR`

On peut combiner ces portes logiques `AND` et `OR` avec la négation. La négation est représentée graphiquement par $\circ$, comme à la figure 3.3. On peut également, grâce à la négation, construire les portes `NAND` (voir la figure 3.4) et `NOR` (voir la figure 3.5). Notons également qu'une porte `NAND` permet

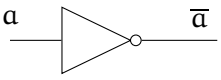


FIGURE 3.3 – Porte `NOT`

de simuler un inverseur. Pour cela, considérons les deux circuits présentés dans la figure 3.6. Aussi, la porte `NAND` permet de représenter la porte `AND` (voir la figure 3.7) ou encore `OR` (voir la figure 3.8) . L'opérateur `NAND` est donc un opérateur qui à lui seul constitue un ensemble fonctionnellement complet.

Ces représentations graphiques consistent en une autre manière de réaliser les calculs. Considérons le circuit représenté à la figure 3.9. Il est équivalent à $a \oplus b$ en logique propositionnelle. Ce circuit se lit en notation usuelle en logique propositionnelle de la manière suivante

$$\neg[\neg(\neg(a \wedge b) \wedge a) \wedge \neg(\neg(a \wedge b) \wedge b)].$$

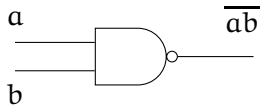


FIGURE 3.4 – Porte NAND

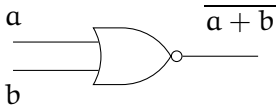


FIGURE 3.5 – Porte NOR

et en notation propre au calcul booléen

$$\overline{\overline{\overline{a}ab}\,\overline{\overline{b}ab}}. \tag{3.75}$$

Nous devons donc établir que

$$\overline{\overline{\overline{a}ab}\,\overline{\overline{b}ab}} = a\overline{b} + \overline{a}b, \tag{3.76}$$

en utilisant (1.6) et la notation propre au calcul booléen. Nous avons

$$\begin{aligned}
\overline{\overline{\overline{a}ab}\,\overline{\overline{b}ab}} &= \overline{\overline{\overline{a}ab}} + \overline{\overline{\overline{b}ab}} && \text{par (3.16)} && (3.77)\\
&= a(\overline{a}+\overline{b}) + b(\overline{a}+\overline{b}) && \text{par (3.13) et (3.16)} && (3.78)\\
&= a\overline{b} + \overline{a}b && \text{par (3.5), (3.7) et (3.2)} && (3.79)
\end{aligned}$$

L'opérateur XOR est également représenté par le graphe de la figure 3.10.

Exemple 3.5 Considérons à présent le circuit de la figure 3.11. Dans un premier temps, on détermine l'expression de la fonction de transmission (la fonction booléenne) modélisée par ce circuit. Celle-ci est

$$pqr + p\overline{q}r + pq\overline{r}. \tag{3.80}$$

En effet, nous devons calculer la sortie de chaque porte dans un ordre bien particulier. Dans le cas présent, la sortie de la porte 1 est

$$pq, \tag{3.81}$$

nécessaire pour calculer la sortie de la porte 3, soit

$$pqr. \tag{3.82}$$

Résumons les sorties des différentes portes dans le tableau suivant.

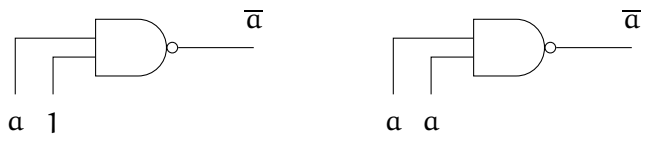


FIGURE 3.6 – Deux circuits équivalent à une porte `NOT`

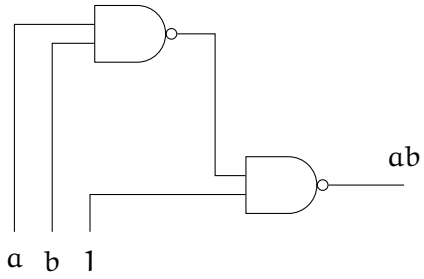


FIGURE 3.7 – Un circuit équivalent à une porte `AND`

Porte	Sortie
1	pq
2	$p\overline{q}$
3	pqr
4	$p\overline{q}r$
5	$pq\overline{r}$
6	$pqr + p\overline{q}r$
7	$pqr + p\overline{q}r + pq\overline{r}$

Représentons dans un diagramme de Karnaugh, la sortie finale. Nous avons le diagramme suivant.

	pq	$\overline{p}q$	$\overline{p}\,\overline{q}$	$p\overline{q}$
r	1			1
$\overline{r}$	1			

On y distingue donc deux couples de cellules grisées. Le premier couple est le suivant

	pq	$\overline{p}q$	$\overline{p}\,\overline{q}$	$p\overline{q}$
r	1			1
$\overline{r}$	1			

Il permet la simplification

$$pqr + p\overline{q}r = pr. \tag{3.83}$$

Le second est

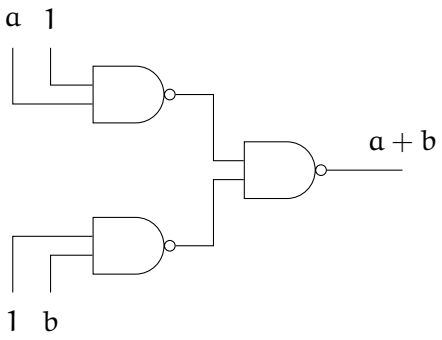


FIGURE 3.8 – Un circuit équivalent à une porte `OR`

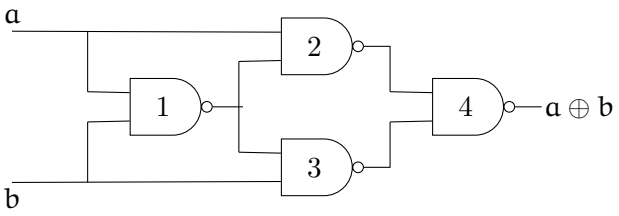


FIGURE 3.9 – Circuit `XOR avec des portes NAND uniquement`

	pq	$\overline{p}q$	$\overline{p}\,\overline{q}$	$p\overline{q}$
r	1			1
$\overline{r}$	1			

qui donne

$$pqr + pq\overline{r} = pq. \tag{3.84}$$

Ainsi

$$pqr + p\overline{q}r + pq\overline{r} = pr + pq. \tag{3.85}$$

Nous avons alors une représentation en circuit plus simple (voir la figure 3.12).

On peut également travailler avec des diagrammes de circuit en permettant aux portes `AND` et `OR` de prendre plus de deux entrées. Nous obtenons alors les circuits 3.13 et 3.14 pour trois entrées.

L'exemple suivant nous donne un cas pratique d'utilisation des diagrammes de Karnaugh. Il est issu de M.Marchand *Outils mathématiques pour l'informaticien, Mathématiques discrètes, 2ème édition.* de Boeck, 2005, pages 239 à 241.

Exemple 3.6 En `Décimal codé binaire` (DCB), chaque chiffre de 0 à 9 est décrit par son code binaire à quatre bits. Les représentations binaires de 1010 (soit

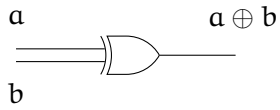


FIGURE 3.10 – Porte `XOR`

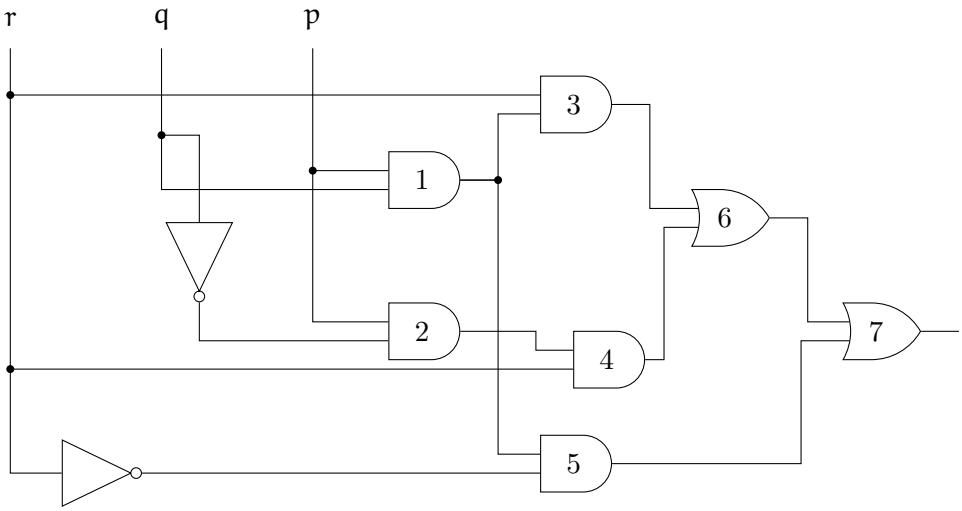


FIGURE 3.11 – Un circuit particulier

10) à `1111` (soit 15) ne sont pas utilisées. Ainsi, tout nombre peut être aisément codé. Par exemple, `721` sera codé comme `011100100001`.

En `Excess-three` (XS3), chaque chiffre de 0 à 9 est décrit par son code binaire à quatre bits auquel on ajoute 3. Dès lors, `721` sera codé comme `101001010100`.

Nous souhaitons construire un circuit logique qui permet de traduire tout nombre exprimé en DCB en un nombre exprimé en XS3. Ce circuit possède donc 4 entrées (un par bit de codage DCB) et 4 sorties (une par bit de codage XS3). Nous avons la table de traduction suivante

	INPUT DCB				OUTPUT XS3			
Décim.	x_1	x_2	x_3	x_4	y_1	y_2	y_3	y_4
0	0	0	0	0	0	0	1	1
1	0	0	0	1	0	1	0	0
2	0	0	1	0	0	1	0	1
3	0	0	1	1	0	1	1	0
4	0	1	0	0	0	1	1	1
5	0	1	0	1	1	0	0	0
6	0	1	1	0	1	0	0	1
7	0	1	1	1	1	0	1	0
8	1	0	0	0	1	0	1	1
9	1	0	0	1	1	1	0	0

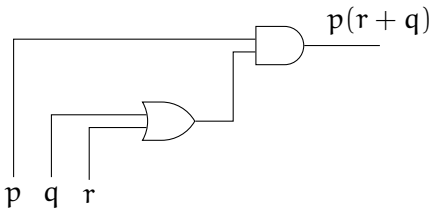


FIGURE 3.12 – Représentation simplifiée du circuit présenté dans la figure 3.11

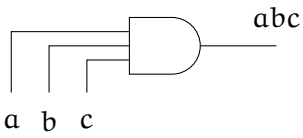


FIGURE 3.13 – Porte `AND avec trois entrées`

Pour découvrir le circuit logique qui réalise cette traduction, il suffit de considérer que chacun des outputs y_1, y_2, y_3 et y_4 est le résultat d'une fonction booléenne à quatre variables (soit x_1, x_2, x_3 et x_4). Etudions la fonction dont l'output est y_1, nous avons le diagramme de Karnaugh suivant

	x_1x_2	$x_1\overline{x}_2$	$\overline{x}_1\,\overline{x}_2$	$\overline{x}_1x_2$
x_3x_4			0	1
$x_3\overline{x}_4$			0	1
$\overline{x}_3\,\overline{x}_4$		1	0	0
$\overline{x}_3x_4$		1	0	1

Notre objectif est de déterminer la fonction qui correspond à ce diagramme. Les cases vides n'étant pas spécifiées par le traducteur ; nous pouvons y placer un 1 ou un 0. La fonction résultante sera plus simple à exprimer si on peut faire des regroupements de cellules. On propose donc de remplir avec des 1 les cases vides. On retrouve alors un premier groupe de huit cellules contiguës, à savoir

	x_1x_2	$x_1\overline{x}_2$	$\overline{x}_1\,\overline{x}_2$	$\overline{x}_1x_2$
x_3x_4	1	1	0	1
$x_3\overline{x}_4$	1	1	0	1
$\overline{x}_3\,\overline{x}_4$	1	1	0	0
$\overline{x}_3x_4$	1	1	0	1

Celles-ci se simplifient en une fonction booléenne simple, soit

$$x_1 \tag{3.86}$$

Nous avons ensuite le regroupement de six cellules

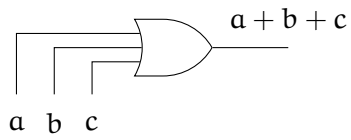


FIGURE 3.14 – Porte `OR avec 3 entrées`

	x_1x_2	$x_1\overline{x}_2$	$\overline{x}_1\,\overline{x}_2$	$\overline{x}_1x_2$
x_3x_4	1	1	0	1
$x_3\overline{x}_4$	1	1	0	1
$\overline{x}_3\,\overline{x}_4$	1	1	0	0
$\overline{x}_3x_4$	1	1	0	1

qui donne

$$x_2(x_3 + x_4) \tag{3.87}$$

La fonction est donc

$$y_1 = x_1 + (x_2(x_3 + x_4)) \tag{3.88}$$

Si nous réalisons la même analyse pour y_2, y_3 et y_4 respectivement, nous obtenons les fonctions

$$\begin{aligned} y_2 &= (\overline{x}_2(x_3 + x_4)) + (x_2\overline{x}_3\overline{x}_4) && (3.89)\\ &= \overline{x}_2(x_3 + x_4) + x_2(\overline{x_3 + x_4}) && (3.90)\\ y_3 &= (\overline{x}_3\overline{x}_4) + (x_3x_4) && (3.91)\\ &= \overline{x_3 + x_4} + (x_3x_4) && (3.92)\\ y_4 &= \overline{x}_4 && (3.93) \end{aligned}$$

où nous voyons apparaître le terme $x_3 + x_4$ plusieurs fois. Ce point est important pour réaliser le circuit tel que présenté dans la figure 3.15.

Exemple 3.7 Il s'agit d'un exemple tiré de J. Vélu, G. Avérous, I. Gil et F. Santi. *Mathématiques pour l'informatique.* Dunod, 2008.

Une entreprise veut embaucher sur un certain poste des candidats ni trop faibles ni trop brillants. Pour cela, elle va noter les candidats et ne retenir pour une audition que ceux dont la note est comprise entre 7 et 12. Voici comment la note N d'un candidat est obtenue : quatre critères sont d'abord notés 0 ou 1, puis on ajoute ces quatre notes en les multipliant chacune par un coefficient, soit

critère	âge	compétence	expérience	sociabilité
Nom de la note	a	b	c	d
Coefficient	7	7	4	2

Soit une fonction booléenne $f(a, b, c, d)$ qui vaut 1 si le candidat est retenu pour l'audition, 0 sinon. La table de vérité de celle-ci est

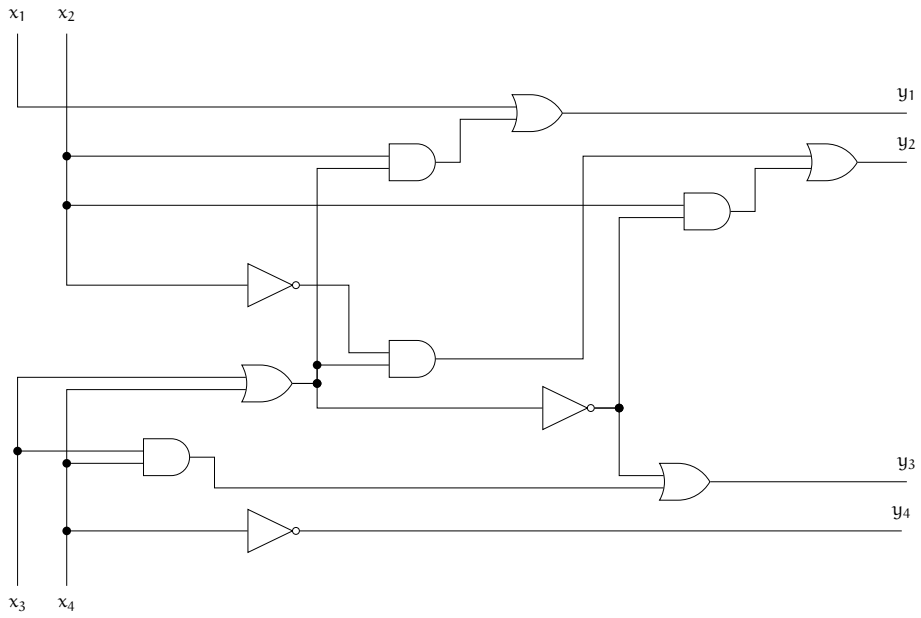


FIGURE 3.15 – Diagramme de circuit du traducteur

a	b	c	d	$7a + 7b + 4c + 2d$	$f(a, b, c, d)$
0	0	0	0	0	0
0	0	0	1	2	0
0	0	1	0	4	0
0	0	1	1	6	0
0	1	0	0	7	1
0	1	0	1	9	1
0	1	1	0	11	1
0	1	1	1	13	0
1	0	0	0	7	1
1	0	0	1	9	1
1	0	1	0	11	1
1	0	1	1	13	0
1	1	0	0	14	0
1	1	0	1	16	0
1	1	1	0	18	0
1	1	1	1	20	0

La fonction booléenne s'exprime donc en forme normale disjonctive comme

$$\overline{a}b\overline{c}\overline{d} + \overline{a}b\overline{c}d + \overline{a}bc\overline{d} + a\overline{b}\overline{c}\overline{d} + a\overline{b}\overline{c}d + a\overline{b}c\overline{d} \tag{3.94}$$

Nous pouvons simplifier cette expression à l'aide du diagramme de Karnaugh. Celui-ci est

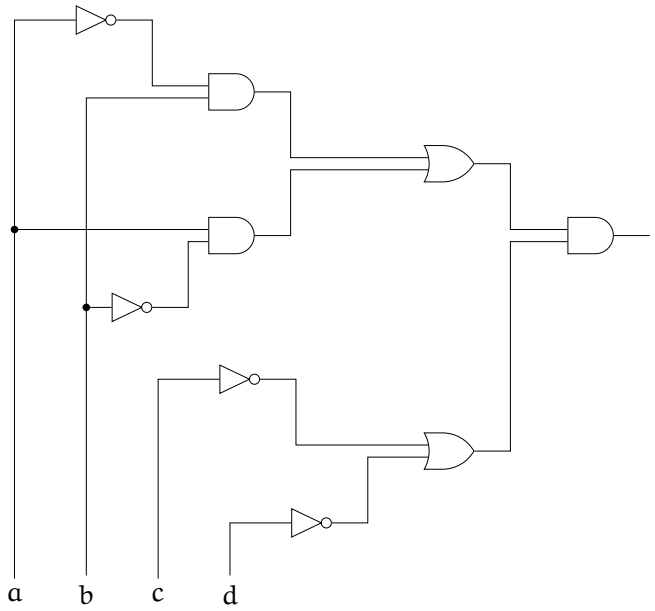


FIGURE 3.16 – Un circuit pour décider si un candidat est retenu

	ab	$\overline{a}b$	$\overline{a}\,\overline{b}$	$a\overline{b}$
cd				
$\overline{c}d$		1		1
$\overline{c}\,\overline{d}$		1		1
$c\overline{d}$		1		1

ce qui donne

$$f(a,b,c,d) = \overline{a}b\overline{c} + \overline{a}b\overline{d} + a\overline{b}\overline{c} + a\overline{b}\,\overline{d}. \tag{3.95}$$

Le circuit correspondant est représenté dans la figure 3.16, en notant que

$$\begin{aligned} f(a,b,c,d) &= \overline{a}b\overline{c} + \overline{a}b\overline{d} + a\overline{b}\overline{c} + a\overline{b}\,\overline{d} && (3.96)\\ &= \overline{a}b(\overline{c}+\overline{d}) + a\overline{b}(\overline{c}+\overline{d}) && (3.97)\\ &= (\overline{a}b + a\overline{b})(\overline{c}+\overline{d}) && (3.98) \end{aligned}$$

Chapitre 4

Codage - Décodage

On distingue essentiellement deux moments clé pour le codage d'un message. Le premier, appelé *codage-source* a pour objectif de coder selon un code source particulier un message exprimé dans un langage particulier en un message binaire. Le code ASCII est un exemple de codage-source. Il utilise 8 bits par caractère. Le second, appelé *codage-transmission* a pour objectif de préparer le message binaire en vue d'une transmission efficace, sans erreur et rapide. En effet, toute transmission de message est sujette à des problèmes de transmission et donc toute transmission est susceptible de comporter des erreurs de transmission. Dans ce chapitre, nous nous intéressons aux fondements du codage-transmission. Les probabilités d'erreur de transmission seront discutées dans le cours INFO B223.

Dans la section 4.1, quelques définitions usuelles sont présentées. Ensuite, on explique comment détecter sur base du code choisi, les erreurs de transmission et éventuellement les moyens pour les corriger. Dans la section 4.3, nous nous concentrons sur les méthodes de codage linéaire systématiques.

4.1 Définitions préalables

L'objectif est la transmission de messages exprimés en format binaire. On parle de *mot* pour désigner l'unité d'information à transmettre.

Définition 4.1

Un mot $\mathbf{b}$ *est un élément de* $\mathbb{B}^n$ *avec* $\mathbb{B} = \{0, 1\}$. *On le désigne par*

$$\mathbf{b} = b_1 b_2 \dots b_n, \tag{4.1}$$

où $b_i \in \mathbb{B}$.

Remarque 4.1 La notation (4.1) représente un élément de $\mathbb{B}^n$. Ainsi, dans $\mathbb{B}^3$ si $\mathbf{b} = 011$, alors $b_1 = 0, b_2 = 1, b_3 = 1$

L'opérateur logique $\oplus$ trouve tout naturellement son correspondant en calcul binaire, l'*addition modulo 2*.

Définition 4.2

*L'*addition modulo 2 *définie sur* $\mathbb{B}$ *est*

$$1 \oplus 1 \stackrel{def}{=} 0 \tag{4.2}$$
$$1 \oplus 0 \stackrel{def}{=} 1 \tag{4.3}$$
$$0 \oplus 1 \stackrel{def}{=} 1 \tag{4.4}$$
$$0 \oplus 0 \stackrel{def}{=} 0 \tag{4.5}$$

Sur $\mathbb{B}^n$, *l'*addition modulo 2 *de deux mots binaires de longueur* n *est obtenue en sommant modulo 2 chaque bit de même position.*

Nous avons alors la propriété suivante.

Propriété 4.1

L'addition modulo 2 définie sur $\mathbb{B}$ *est telle que pour tout* a, b *et* $c \in \mathbb{B}$ *:*

$$(a \oplus b = c) \Rightarrow (b \oplus c = a) \wedge (c \oplus a = b). \tag{4.6}$$

L'addition modulo 2 nous permet de définir la notion de *distance de Hamming* particulièrement utile dans la suite. Celle-ci utilise également la notion de *poids* d'un mot.

Exemple 4.1 Cette dernière propriété est également utilisée dans des techniques de stockage dites redondantes. Ces techniques de stockage permettent de conserver une même information plusieurs fois sans pour autant multiplier le nombre de support d'enregistrement. Considérons le cas présenté dans l'exemple suivant.

Imaginons que nous devons enregistrer la séquence suivante de seize bits

$$1100011010111001 \tag{4.7}$$

sur trois disques. Pour créer de la redondance, on peut placer tous les bits en position impaire sur le premier disque et les autres sur le second disque. Le troisième disque contient ensuite le résultat de l'opération $\oplus$ appliquée au bit situé en même position sur les deux disques. Ainsi, dans notre exemple, le premier disque contient la séquence

$$10011110 \tag{4.8}$$

le second

$$10100101 \tag{4.9}$$

tandis que le troisième

$$00111011. \tag{4.10}$$

Si un disque fait défaut, on pourra aisément reconstruire son contenu en consultant les deux autres disques.

Définition 4.3

Le poids d'un mot $\mathbf{b}$*, noté* $w(\mathbf{b})$ *est le nombre de bits égaux à 1 dans le mot* $\mathbf{b}$*.*

La distance de Hamming entre $\mathbf{a}$ *et* $\mathbf{b}$*, tous deux issus de* $\mathbb{B}^n$*, distance notée* $\delta(\mathbf{a}, \mathbf{b})$ *est le nombre de fois qu'un bit de même position est distinct dans* $\mathbf{a}$ *et* $\mathbf{b}$*.*

Dès lors, nous pouvons démontrer que

$$\delta(\mathbf{a}, \mathbf{b}) = w(\mathbf{a} \oplus \mathbf{b}), \tag{4.11}$$

et que la distance de Hamming est une distance à proprement parler. Elle est donc symétrique, soit

$$\delta(\mathbf{a}, \mathbf{b}) = \delta(\mathbf{b}, \mathbf{a}), \tag{4.12}$$

elle est positive,

$$\delta(\mathbf{a}, \mathbf{b}) \geq 0 \tag{4.13}$$

elle respecte l'inégalité triangulaire

$$\delta(\mathbf{a}, \mathbf{b}) + \delta(\mathbf{b}, \mathbf{c}) \geq \delta(\mathbf{a}, \mathbf{c}), \tag{4.14}$$

et

$$(\delta(\mathbf{a}, \mathbf{b}) = 0) \Leftrightarrow \mathbf{a} = \mathbf{b}. \tag{4.15}$$

Enfin, elle est invariante par translation, soit $\mathbf{a}, \mathbf{b} \in \mathbb{B}^n$, alors

$$\delta(\mathbf{a} \oplus \mathbf{x}, \mathbf{b} \oplus \mathbf{x}) = \delta(\mathbf{a}, \mathbf{b}), \tag{4.16}$$

pour tout mot $\mathbf{x} \in \mathbb{B}^n$.

Le *codage par blocs* est une méthode de présentation des mots à transmettre en vue de permettre au récepteur du mot de détecter la présence éventuelle d'erreurs de transmission, et le cas échéant de les corriger. Mathématiquement parlant, le codage est la transformation ϕ d'un mot de $\mathbb{B}^n$ dans un mot de $\mathbb{B}^p$ (avec $p > n$). L'adjonction de ces bits supplémentaires doit permettre au récepteur de détecter et de corriger les erreurs éventuelles. On parle de *bits de contrôle* ou *bits de parité* pour désigner ces bits supplémentaires. Lorsque ces bits de contrôle sont placés à la fin du mot, on dit que le codage est *systématique*.

Définition 4.4

Dans le cas d'une transformation ϕ *de mot de longueur* n *vers des mots de longueur* p*, on parle alors de* (n, p)-codage.

L'image $\phi(\mathbb{B}^n)$ *d'un codage* ϕ *est l'ensemble des* mots-codes *obtenus*

par le codage ϕ. *L'ensemble* $\phi(\mathbb{B}^n)$ *est le* code *obtenu par* ϕ.

Le rendement *ou le* taux *d'un codage est le rapport* τ *défini comme*

$$\tau \stackrel{def}{=} \frac{n}{p}. \tag{4.17}$$

Notons enfin que le mot transmis est appelé *message*.

Le décodage est mathématiquement également une fonction.

Définition 4.5

Le décodage Φ *associé à un codage* $\phi : \mathbb{B}^n \to \mathbb{B}^p$ *est une fonction telle que*

$$\Phi \circ \phi = \mathbf{1}_{\mathbb{B}^n}, \tag{4.18}$$

où $\circ$ *est la loi de composition des fonctions et* $\mathbf{1}_{\mathbb{B}^n}$ *est la relation identique sur* $\mathbb{B}^n$.

Remarque 4.2 L'opération de composition des fonctions $\Phi \circ \phi$ est définie pour tout $\mathbf{b} \in \mathbb{B}^n$ comme

$$\Phi \circ \phi(\mathbf{b}) \stackrel{\text{def}}{=} \Phi(\phi(\mathbf{b})). \tag{4.19}$$

Pour terminer cette section, nous considérons deux codages particuliers, le *codage avec contrôle de parité* et le *codage par répétition*.

Codage avec contrôle de parité Le principe est le suivant. Chaque mot $\mathbf{b} = b_1 b_2 \dots b_n$ est codé par $\phi(\mathbf{b}) = b_1 b_2 \dots b_n b_{n+1}$ où le bit supplémentaire b_{n+1} est tel que

$$b_{n+1} = \begin{cases} 1 & \text{si } w(\mathbf{b}) \text{ est impair} \\ 0 & \text{dans le cas contraire.} \end{cases} \tag{4.20}$$

Le rendement de ce $(n, n+1)$-codage est

$$\tau = \frac{n}{n+1}. \tag{4.21}$$

Clairement un mot transmis $\phi(\mathbf{b})$ n'est pas un mot-code dès lors que $w(\phi(\mathbf{b}))$ est impair.

Un décodage possible est de simplement supprimer le dernier bit à tout message reçu. Avec une telle technique de décodage, il n'y aucun intérêt à la présence du bit de contrôle pour une éventuelle correction.

Notons cependant que ce bit de contrôle permet juste de détecter la présence éventuelle d'erreur.

Codage par répétition Le principe est le suivant. Chaque bit du mot $\mathbf{b}$ est transmis séparément et est répété un nombre k de fois. Le mot $\mathbf{b} = b_1 b_2 \dots b_n$ sera donc transmis sous la forme

$$\phi(\mathbf{b}) = \underbrace{b_1 b_1 \dots b_1}_{k} b_2 b_2 \dots b_2 \dots b_n b_n \dots b_n. \tag{4.22}$$

Le rendement de ce (n, kn)-codage est

$$\tau = \frac{1}{k} \tag{4.23}$$

Clairement un mot transmis $\phi(\mathbf{b})$ n'est pas un mot-code dès lors que le mot ne peut être divisé en blocs de k bits tous égaux.

De la même manière, on peut coder un mot en le répétant k fois plutôt que de répéter chaque bit.

On détecte *certainement* une erreur dans la transmission dès lors qu'on n'obtient pas un mot-code. Se pose alors la question de la correction de cette erreur.

4.2 La détection et la correction d'erreur

Une fois l'erreur détectée, la correction se base alors sur le principe suivant : *l'erreur commise est vraisemblablement l'erreur la plus probable*. Nous proposons dans cette section de déterminer le mot-code le plus vraisemblable sans passer par un calcul de probabilité. Avant de ce faire, nous expliquons les conditions dans lesquelles une erreur peut être détectée de manière *certaine*. Une définition préalable s'impose.

Définition 4.6

La distance minimale du code, notée δ est la plus petite distance de Hamming séparant deux mots-code distincts.

Nous avons alors la propriété suivante.

Propriété 4.2

Tous les messages faux comportant un nombre d'erreur strictement plus petit que δ sont nécessairement détectés.

On dit alors que le codage ϕ est $(\delta - 1)$-*détecteur*.

Exemple 4.2 Le codage avec contrôle de parité est 1-détecteur. On ne peut détecter la présence d'erreur dans un message où deux bits auraient été modifiés.

Le codage par répétition de taux $\tau = 1/k$ est un $(k-1)$-détecteur. Il est certain que le message doit contenir la répétition de k fois le même bit. Si moins de $k-1$ d'entre eux sont modifiés, le message reçu n'est pas un mot-code.

Pour la correction, nous avons cette fois la définition suivante.

Définition 4.7

On dit d'un décodage $\Phi : \mathbb{B}^p \to \mathbb{B}^n$ *associé à un codage* ϕ *qu'il est* k-correcteur *si chaque fois que le nombre d'erreurs de transmission ne dépasse pas* k*, le mot reçu est corrigé.*

Supposons que nous travaillons avec un codage à répétition où $k = 3$. Lorsqu'on reçoit le message 001, doit-on déduire qu'il y a eu une seule erreur de transmission ou deux ? Le principe sur lequel se base tout décodeur est de poser que l'erreur la plus probable a été commise, celle comprenant donc le moins de bits erronés. Ensuite, pour corriger le message, le décodeur choisit le mot-code le plus proche du message, proche dans le sens de la distance de Hamming. Il applique ici le principe du maximum de vraisemblance.

Nous avons alors la propriété suivante.

Propriété 4.3

Lorsque le nombre N *d'erreurs d'un message est tel que* $N < \delta/2$*, le message peut être corrigé.*

DIST MIN du Code

Exemple 4.3 Pour le codage avec répétition, tout message comprenant strictement moins de $k/2$ erreurs pourra être corrigé (selon le principe du maximum de vraisemblance).

Considérons à présent l'étude de cas complète.

Exemple 4.4 Soit le $(3, 8)$-codage suivant

m	$\phi(m)$
000	00000000
001	01110010
010	10011100
011	10100101
100	01101101
101	10110000
110	11110111
111	00001111

Les mots-code définis par le codage $\phi(.)$ se trouvent donc dans la seconde colonne. Par exemple, le mot 1010101010 n'est pas un mot-code.

Observons à présent la distance de Hamming qui sépare chaque couple de mot-code.

	00000000	01110010	10011100	10100101	01101101	10110000	11110111
01110010	4						
10011100	4	6					
10100101	4	6	4				
01101101	5	5	5	3			
10110000	3	3	3	3	6		
11110111	7	3	5	3	4	4	
00001111	4	6	4	4	3	7	5

La distance δ minimale du code est donc $\delta = 3$. Le code ϕ est donc 2-détecteur.

Soit le mot 10100101 transmis et mal-réceptionné comme 10110000, soit avec 3 erreurs. Pour le récepteur, il s'agit pourtant d'un mot-code. Il ne peut donc pas détecter l'erreur.

Rappelons le encore une fois ; toute la démarche de détection et de correction d'erreur part du principe que *l'erreur commise est vraisemblablement l'erreur la plus probable*.

4.3 Codage linéaire

On distingue la notion de *code linéaire* de *codage linéaire*. Voici leur définition.

Définition 4.8

Un code est un code linéaire *lorsque la somme modulo 2 de deux mots-code quelconques est un mot-code.*

Un codage est un codage linéaire *lorsque la transformation ϕ respecte l'égalité suivante*

$$\phi(m_1 \oplus m_2 \ldots \oplus m_k) = \phi(m_1) \oplus \phi(m_2) \ldots \oplus \phi(m_k) \tag{4.24}$$

et ce pour tout k et tous mots $m_1, m_2, \ldots, m_k$.

Quand un codage est linéaire, son code est linéaire et 0 dans $\mathbb{B}^p$ est le mot-code obtenu par le codage du mot $0 \in \mathbb{B}^n$.

Un codage linéaire est tel qu'il est complètement défini lorsqu'on connaît les mots-code obtenus par le codage des n mots suivants

$$b_1 = 100\ldots00 \tag{4.25}$$
$$b_2 = 010\ldots00 \tag{4.26}$$
$$\ldots \tag{4.27}$$
$$b_{n-1} = 000\ldots10 \tag{4.28}$$
$$b_n = 000\ldots01. \tag{4.29}$$

Ceux-ci constituent ce qu'on appelle la *base canonique* de $\mathbb{B}^n$. Leur mot-code correspondant est

$$\phi(b_1), \phi(b_2), \ldots, \phi(b_n). \tag{4.30}$$

Ainsi, prenons b qui s'écrit comme une combinaison linéaire modulo 2 de $b_1, b_2, \ldots, b_n$, soit

$$b = \sum_{i=1}^{n} a_i b_i \tag{4.31}$$

avec $a_i = 0$ ou 1, pour tout $i \in \{1, \dots, n\}$. Alors,

$$\phi(\mathbf{b}) = \sum_{i=1}^{n} a_i \phi(\mathbf{b}_i) \tag{4.32}$$

par les propriétés d'un codage-linéaire.

Remarque 4.3 Le symbole $\sum$ est utilisé dans ce contexte pour désigner la somme modulo 2.

Exemple 4.5 Reprenons le codage avec contrôle de parité où $n = 3$ et $p = 4$. La base canonique est

$$\mathbf{b}_1 = 100 \tag{4.33}$$
$$\mathbf{b}_2 = 010 \tag{4.34}$$
$$\mathbf{b}_3 = 001 \tag{4.35}$$

dont les mots-code correspondant sont

$$\phi(\mathbf{b}_1) = 1001 \tag{4.36}$$
$$\phi(\mathbf{b}_2) = 0101 \tag{4.37}$$
$$\phi(\mathbf{b}_3) = 0011. \tag{4.38}$$

Ainsi le mot $\mathbf{b} = 011$ tel que

$$\mathbf{b} = 0 \cdot \mathbf{b}_1 \oplus 1 \cdot \mathbf{b}_2 \oplus 1 \cdot \mathbf{b}_3 \tag{4.39}$$

est tel que son mot-code correspondant est

$$\phi(\mathbf{b}) = 0 \cdot \phi(\mathbf{b}_1) \oplus 1 \cdot \phi(\mathbf{b}_2) \oplus 1 \cdot \phi(\mathbf{b}_3) \tag{4.40}$$
$$= 0101 \oplus 0011 \tag{4.41}$$
$$= 0110. \tag{4.42}$$

Grâce à cette base canonique, on obtient une représentation matricielle pratique pour coder un mot par un codage ϕ. Dans le cas d'un codage systématique, le codage se fait grâce à une matrice appelé *matrice génératrice du codage*, notée $\mathbf{G}$.

Définition 4.9

La matrice génératrice du codage, *notée* $\mathbf{G}$, *est obtenue par la juxtaposition de deux matrices, à savoir*

- *une matrice identité de taille* n, *soit les mots* $\mathbf{b}_i$ *écrits sous-forme de vecteur ligne,*
- *une matrice dite* de parité, *contenant les bits de contrôle des mots-code* $\phi(\mathbf{b}_i)$.

On obtient donc une matrice $n \times p$. Grâce à elle, on obtient directement le codage de n'importe quel mot de $\mathbb{B}^n$ en procédant comme suit

$$\mathbf{bG}. \tag{4.43}$$

Remarque 4.4 Le produit matriciel tel que manipulé en (4.43) se définit de la manière suivante. Soit les matrices $\mathbf{A} = (a_{ij})$ (avec $1 \leq i \leq n$ et $1 \leq j \leq p$) et $\mathbf{B} = (b_{ij})$ (avec $1 \leq i \leq p$ et $1 \leq j \leq m$), la matrice $\mathbf{C} = \mathbf{AB}$ est une matrice $\mathbf{C} = (c_{ij})$ telle que

$$c_{ij} = \sum_{k=1}^{p} (a_{ik}\, b_{kj}) \tag{4.44}$$

$$= a_{i1}\, b_{1j} \oplus a_{i2}\, b_{2j} \oplus \ldots \oplus a_{ip}\, b_{pj}, \tag{4.45}$$

pour $1 \leq i \leq n$ et $1 \leq j \leq m$.

Exemple 4.6 Le codage avec contrôle de parité est un codage linéaire. Sa matrice génératrice dans le cas où $n = 3$, est

$$\begin{pmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \end{pmatrix} \tag{4.46}$$

Ainsi le mot 110 est codé sous la forme

$$\begin{pmatrix} 1 & 1 & 0 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \end{pmatrix} = \begin{pmatrix} 1 & 1 & 0 & 0 \end{pmatrix} \tag{4.47}$$

Le codage par répétition est également un codage linéaire. Dans le cas où $n = 2$ (on considère des mots de longueur 2), et pour une répétition de trois fois chaque mot, on obtient comme matrice génératrice

$$\begin{pmatrix} 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 \end{pmatrix} \tag{4.48}$$

soit la matrice identité de taille 2×2 à laquelle on juxtapose une matrice de parité où les vecteurs de la base canonique sont répétés trois fois.

Exemple 4.7 Attention, pour que la matrice $\mathbf{G}$ ait la forme telle que présentée dans la définition 4.9, le codage doit être linéaire et systématique. Considérons le codage suivant

$\mathbf{m}$	$\phi(\mathbf{m})$
000	0000
001	1110
010	0111
011	1001
100	1011
101	0101
110	1100
111	0010

Il s'agit bien d'un codage linéaire. Il n'est cependant pas systématique. Remarquons par exemple la transformation de 001 en 1110.

Correction par le tableau standard

Dans le cas d'un codage linéaire systématique, on peut également construire un tableau qui permet de repérer rapidement le mot-code le plus proche d'un message afin de le corriger en respectant la méthode du maximum de vraisemblance. Ce tableau se construit de la manière suivante. Nous prenons comme illustration le codage avec contrôle de parité où $n = 3$ et $p = 4$.

1. On classe les mots de $\mathbb{B}^p$ selon le poids de ceux-ci. Si deux mots ont le même poids, on les classe par ordre lexicographique avec 0 avant 1. On obtient une liste $\mathcal{B}$.
   Pour $\mathbb{B}^4$, nous avons les mots suivants et leur poids correspondant.

$\mathbb{B}^4$	Poids
0000	0
0001	1
0010	1
0011	2
0100	1
0101	2
0110	2
0111	3
1000	1
1001	2
1010	2
1011	3
1100	2
1101	3
1110	3
1111	4

   ce qui donne le classement suivant

$$\begin{array}{cccccccc} 0000 & 0001 & 0010 & 0100 & 1000 & 0011 & 0101 & 0110 \\ 1001 & 1010 & 1100 & 0111 & 1011 & 1101 & 1110 & 1111 \end{array} \quad (4.49)$$

2. Par définition, le tableau comportera 2^n colonnes et $2^{(p-n)}$ lignes.
   Pour le codage avec contrôle de parité, nous aurons donc un tableau de $2^3 = 8$ colonnes et $2^{(4-3)} = 2$ lignes.
3. On remplit le tableau en commençant par la première ligne. On y place les mots-code en ordre de classement.
   Pour le codage avec contrôle de parité, les mots-code sont dans l'ordre suivant l'ordre observé dans (4.49), soit

$$0000 \quad 0011 \quad 0101 \quad 0110 \quad 1001 \quad 1010 \quad 1100 \quad 1111 \quad (4.50)$$

   ce qui donne la première ligne du tableau standard.

4. Dans les lignes suivantes, on place le premier mot de la liste $\mathcal{B}$ qui n'est pas encore utilisé dans le remplissage du tableau.
   Pour notre exemple, on obtient

   0000 0011 0101 0110 1001 1010 1100 1111
   0001

5. Cet élément est sommé (modulo 2) à l'élément qui se trouve dans la première ligne, pour obtenir l'élément dans la colonne en question.
   On obtient dans le cas du codage avec contrôle de parité,

   0000 0011 0101 0110 1001 1010 1100 1111
   0001 0010

   et ainsi de suite jusqu'à obtenir

   0000 0011 0101 0110 1001 1010 1100 1111
   0001 0010 0100 0111 1000 1011 1101 1110

Pour le cas du codage à répétition avec $n = 1$ et $p = 3$, soit une répétition de 3 fois le même bit. On obtient les étapes suivantes

1. On détermine le poids de chaque mot,

$\mathbb{B}^3$	Poids
000	0
001	1
010	1
011	2
100	1
101	2
110	2
111	3

   ce qui donne une liste $\mathcal{B}$ définie comme

   000 001 010 100 011 101 110 111 (4.51)

2. Le tableau comporte $2^1 = 2$ colonnes et $2^{3-1} = 4$ lignes.
3. La première ligne est

   000 111

4. Le tableau se remplit ensuite ligne par ligne, soit d'abord

   000 111
   001

puis

000	111
001	110

et finalement

000	111
001	110
010	101
100	011

Grâce au tableau, tout message erroné est corrigé en utilisant le mot-code (situé sur la première ligne) de la même colonne où se trouve le mot erroné. On prend conscience que plus l'erreur est peu probable, plus le mot se trouve dans une ligne basse.

Correction par la liste des syndromes

Plutôt que de construire le tableau complètement, on peut calculer directement ce qu'on appelle le *syndrome* du message obtenu par le récepteur. Si ce syndrome est nul, alors le message reçu est bien un mot-code. Nous expliquons comment obtenir le syndrome d'un message et ensuite comment corriger un message erroné en utilisant son syndrome.

Définition 4.10

Soit un codage ϕ. *La* matrice de contrôle, *notée* $\mathbf{H}$ *est une matrice à* $p - n$ *lignes et à* p *colonnes obtenue en juxtaposant la transposée de la matrice de parité avec une matrice identité de dimension* $p - n$.

Le syndrome *d'un message* $\mathbf{m}$, *noté* $\sigma(\mathbf{m})$ *est*

$$\sigma(\mathbf{m}) \stackrel{def}{=} \mathbf{m}\mathbf{H}', \tag{4.52}$$

où $\mathbf{H}'$ *est la transposée de la matrice* $\mathbf{H}$.

Attention, il s'agit du produit matriciel où la somme est la somme modulo 2.

Nous avons la propriété suivante.

Propriété 4.4

Le syndrome de la somme modulo 2 de deux messages est la somme modulo 2 des syndromes de chaque message, soit

$$\sigma(\mathbf{m} \oplus \mathbf{n}) = \sigma(\mathbf{m}) \oplus \sigma(\mathbf{n}). \tag{4.53}$$

Exemple 4.8 Considérons le codage avec contrôle de parité où $n = 3$ et $p = 4$. La matrice de contrôle est une matrice à 1 ligne et 4 colonnes, soit

$$\mathbf{H} = \begin{pmatrix} 1 & 1 & 1 & 1 \end{pmatrix}. \tag{4.54}$$

Ainsi le message 1101 qui est érroné, donne le syndrome

$$\sigma(\begin{pmatrix} 1 & 1 & 0 & 1 \end{pmatrix}) = \begin{pmatrix} 1 \end{pmatrix} \tag{4.55}$$

soit un mot de $p - n$ bits. Pour le message 1100 qui est un mot-code, nous avons

$$\sigma(\begin{pmatrix} 1 & 1 & 0 & 0 \end{pmatrix}) = \begin{pmatrix} 0 \end{pmatrix} \tag{4.56}$$

Le codage par répétition tel que manipulé dans l'exemple 4.6 correspond lui à une matrice de contrôle de taille 4×6, soit

$$\mathbf{H} = \begin{pmatrix} 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 \end{pmatrix} \tag{4.57}$$

Ainsi, le message 110111 qui est erroné a pour syndrome

$$\sigma(\begin{pmatrix} 1 & 1 & 0 & 1 & 1 & 1 \end{pmatrix}) = \begin{pmatrix} 1 & 0 & 0 & 0 \end{pmatrix} \tag{4.58}$$

qui n'étant pas le vecteur nul nous permet de conclure que ce message n'est pas un mot-code. Pour le message 010101, nous avons

$$\sigma(\begin{pmatrix} 0 & 1 & 0 & 1 & 0 & 1 \end{pmatrix}) = \begin{pmatrix} 0 & 0 & 0 & 0 \end{pmatrix}. \tag{4.59}$$

Celui-ci est donc bien un mot-code.

Le syndrome d'un message, si il est différent du vecteur nul nous indique la présence d'erreur.

A présent, nous mettons en évidence une méthode pratique pour corriger, le cas échéant, un message. Cette méthode se base sur une liste de syndromes. Cette liste remplace le tableau standard. Pour chaque ligne du tableau standard, on détermine le syndrome correspondant. En effet, tous les mots situés sur une même ligne appartiennent à une même classe d'équivalence définie sur base de la relation $\mathcal{R}$.

Définition 4.11

On dit de deux mots $\mathbf{m}, \mathbf{n}$ de $\mathbb{B}^p$ qu'ils sont en relation $\mathcal{R}$, noté $\mathbf{m}\,\mathcal{R}\,\mathbf{n}$ lorsque $\mathbf{m} \oplus \mathbf{n}$ est un mot-code.

Représentons cette relation pour le codage avec contrôle de parité lorsque $n = 2$ et $p = 3$.

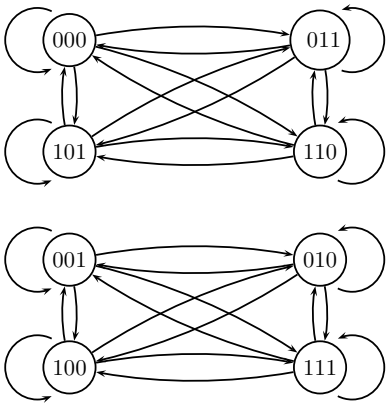
Exemple 4.9 Dans un premier temps, on calcule la somme de tous les mots appartenant à $\mathbb{B}^3$.

	000	001	010	011	100	101	110	111
000	000							
001	001	000						
010	010	011	000					
011	011	010	001	000				
100	100	101	110	111	000			
101	101	100	111	110	001	000		
110	110	111	100	101	010	011	000	
111	111	110	101	100	011	010	001	000

Les mots-code du codage sont

$$000 \qquad 011 \qquad 101 \qquad 110.$$

On les souligne dans le tableau. Ainsi, lorsque une somme $\mathbf{a} \oplus \mathbf{b}$ est soulignée, cela signifie que $\mathbf{a} \mathcal{R} \mathbf{b}$. Le graphe correspondant est



On y remarque facilement une partition en deux sous-ensembles. Ceux-ci sont ce qu'on appelle des *classes d'équivalence*. Celles-ci sont

$$\{000, 011, 101, 110\}$$
$$\{001, 010, 100, 111\}$$

Cette relation $\mathcal{R}$ est une relation d'équivalence. Elle définit donc des classes d'équivalence. Si on observe bien le cas d'étude considéré, on prend conscience que chaque classe correspond en fait à une ligne du tableau standard. Nous avons également le résultat suivant.

Propriété 4.5

Deux messages sont équivalents pour la relation $\mathcal{R}$ si et seulement si ils ont les mêmes syndromes.

Démonstration : On établit dans cette preuve uniquement que la condition est suffisante.

Soit deux mots $\mathfrak{m}$ et $\mathfrak{n}$ tels que leur somme modulo 2 est un mot-code, soit

$$\sigma(\mathfrak{m} \oplus \mathfrak{n}) = \mathfrak{0}. \tag{4.60}$$

Or, par (4.53), nous avons

$$\sigma(\mathfrak{m} \oplus \mathfrak{n}) = \sigma(\mathfrak{m}) \oplus \sigma(\mathfrak{n}) \tag{4.61}$$

ce qui implique que

$$\sigma(\mathfrak{m}) = \sigma(\mathfrak{n}). \tag{4.62}$$

Dès lors, deux mots dans la même classe d'équivalence ont le même syndrome. ■

Ainsi, à chaque ligne du tableau standard correspond un syndrome. Ces syndromes permettent de construire ce qu'on appelle la *liste des syndromes*. A la réception d'un message, il suffit au récepteur de calculer son syndrome. En fonction, il peut corriger le message en utilisant le mot correspondant à la colonne où se situe le message. Plutôt que d'utiliser le tableau standard, il utilise un tableau de deux colonnes où dans la première se placent les syndromes (classés par ordre lexicographique) et dans la seconde, se trouve un message de $\mathbb{B}^p$. Grâce à celui-ci, on procède à la correction des mots erronés. Pour construire ce tableau, appelé *liste des syndromes*, on procède comme suit. Nous illustrons notre propos en considérant le cas du codage avec contrôle de parité où $n = 2$ et $p = 3$.

1. Dans la première colonne, on place l'ensemble des syndromes possibles (tels que définis dans $\mathbb{B}^{p-n}$).

   Dans le cas du codage avec contrôle de parité où $n = 2$ et $p = 3$, nous avons $\mathbb{B}^1$ et donc les syndromes suivants

Syndrome	Message
0	
1	

2. On classe les mots de $\mathbb{B}^p$ par poids croissant. A poids égal, on les classe par ordre lexicographique.

   Dans notre étude de cas, voici les mots classés, soit $\mathcal{B}$

   $$000 \quad 001 \quad 010 \quad 100 \quad 011 \quad 101 \quad 110 \quad 111$$

3. Dans l'ordre ainsi déterminé, on calcule le syndrome de chaque mot jusqu'à obtenir tous les syndromes possibles. On retient dans le tableau, le premier mot de $\mathcal{B}$ qui donne chaque syndrome.

   Dans notre cas, le tableau se complète de la manière suivante

Syndrome	Message
0	000
1	001

puisque la matrice $\mathbf{H}'$ est

$$\mathbf{H}' = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} \tag{4.63}$$

Grâce à ce tableau, on peut procéder à la correction de message faux. On procède comme décrit ci-dessous

1. Calculer le syndrome du message reçu.

   Dans le cas d'un codage avec contrôle de parité où $n = 2$ et $p = 3$, le récepteur reçoit le message suivant 010. Son syndrome est $\sigma(010) = (1)$.

2. Corriger le message en lui additionnant le message répertorié dans la liste des syndromes pour l'entrée correspondant au syndrome obtenu.

   Dans notre cas, le message correspondant au syndrome est 001. Le message corrigé est donc 011.

Considérons à présent l'étude de cas complète suivante.

Exemple 4.10 Nous travaillons avec un codage avec répétition. On répète une fois les mots de deux bits. Ainsi,

$$n = 2 \tag{4.64}$$
$$p = 4. \tag{4.65}$$

1. La matrice génératrice est la suivante

$$\begin{pmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{pmatrix} \tag{4.66}$$

   puisque $\mathbf{b}_1 = 10$ et $\mathbf{b}_2 = 01$. Dès lors, les bits de contrôle seront respectivement 10 et 01.

2. Les mots de $\mathbb{B}^4$ sont classés selon leur poids, ce qui donne la liste $\mathcal{B}$,

$\mathbb{B}^4$	Poids	$\mathcal{B}$
0000	0	0000
0001	1	0001
0010	1	0010
0011	2	0100
0100	1	1000
0101	2	0011
0110	2	0101
0111	3	0110
1000	1	1001
1001	2	1010
1010	2	1100
1011	3	0111
1100	2	1011
1101	3	1101
1110	3	1110
1111	4	1111

Les mots-code pour ce codage, sont

$$0000 \qquad 0101 \qquad 1010 \qquad 1111$$

ce qui donne le tableau standard suivant

0000	0101	1010	1111
0001	0100	1011	1110
0010	0111	1000	1101
0011	0110	1001	1100

(4.67)

3. On calcule à présent les sommes modulo 2 des mots de $\mathbb{B}^4$, ce qui donne le tableau 4.1. Dans cette table, nous avons également souligné tous les mots-code de notre codage. On obtient bien quatre classes d'équivalence telle que le déterminaient les quatre lignes du tableau standard (donné en (4.67)).
4. La matrice de contrôle est la suivante

$$\mathbf{H} = \begin{pmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{pmatrix} \tag{4.68}$$

5. Il y a 2^2 syndromes, à savoir

$$00 \qquad 01 \qquad 10 \qquad 11 \tag{4.69}$$

6. La liste des syndromes est

Syndrome	Message
00	0000
01	0001
10	0010
11	0011

	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
0000	0000															
0001	0001	0000														
0010	0010	0011	0000													
0011	0011	0010	0001	0000												
0100	0100	0101	0110	0111	0000											
0101	0101	0100	0111	0110	0001	0000										
0110	0110	0111	0100	0101	0010	0011	0000									
0111	0111	0110	0101	0100	0011	0010	0001	0000								
1000	1000	1001	1010	1011	1100	1101	1110	1111	0000							
1001	1001	1000	1011	1010	1101	1100	1111	1110	0001	0000						
1010	1010	1011	1000	1001	1110	1111	1100	1101	0010	0011	0000					
1011	1011	1010	1001	1000	1111	1110	1101	1100	0011	0010	0001	0000				
1100	1100	1101	1110	1111	1000	1001	1010	1011	0100	0101	0110	0111	0000			
1101	1101	1100	1111	1110	1001	1000	1011	1010	0101	0100	0111	0110	0001	0000		
1110	1110	1111	1100	1101	1010	1011	1000	1001	0110	0111	0100	0101	0010	0011	0000	
1111	1111	1110	1101	1100	1011	1010	1001	1000	0111	0110	0101	0100	0011	0010	0001	0000

TABLE 4.1 – Somme modulo 2 des différents mots de $\mathbb{B}^4$

Prenons à présent le message 0110. Il ne s'agit pas d'un mot-code. Son syndrome est (11). On doit donc additionner 0011 au message réceptionné. Ainsi

$$0110 \oplus 0011 = 0101. \tag{4.70}$$

En utilisant le tableau standard, on obtenait la même correction.

Prenons le message envoyé 1010, réceptionné, on obtient 1110. Par la technique de correction, on obtient 1111 qui n'était pas le message envoyé. En effet, notre code est 1-détecteur mais 0-correcteur. Dès lors, une erreur ne sera pas nécessairement corrigée correctement. La technique de maximum de vraisemblance est utilisée et elle a proposé cette correction, sans garantie que cela soit la bonne correction. Sur base de ce codage, le récepteur n'a pas suffisamment d'information pour corriger le message erroné de façon certaine.

En conclusion, mettons en évidence le résultat suivant.

Propriété 4.6

La condition nécessaire et suffisante pour qu'un codage linéaire permette de corriger de façon certaine toutes les erreurs de poids 1 est que la matrice $\mathbf{H}'$ ait toutes ses lignes distinctes et non-nulles

Exemple 4.11 Il est clair que la matrice $\mathbf{H}'$ telle que calculée en (4.68) ne possède pas cette propriété.

Supposons qu'on travaille avec r bits de contrôle. Le codage est entièrement déterminé par la matrice de parité $\mathbf{P}$ (de taille $n \times r$). En effet, on construit la matrice génératrice sur base d'une juxtaposition de la matrice identité et de la matrice de parité. Il faut donc la choisir pour que la matrice $\mathbf{H}$ respecte la condition énoncée dans la propriété 4.6. Nous devons donc choisir n mots de longueur r pour constituer $\mathbf{P}$, n mots distincts et différents de la base canonique de $\mathbb{B}^r$, et distincts du vecteur nul. Dans $\mathbb{B}^r$ reste le choix entre $2^r - r - 1$ mots. Ces n mots distincts existent lorsque $n \leq 2^r - r - 1$. Lorsque $n = 2^r - r - 1$, on parle de *code de Hamming*.

Définition 4.12

Un code de Hamming *est un code linéaire avec $n = 2^r - r - 1$ et $p = (n+r)$ dont la matrice $\mathbf{H}'$ est alors constituée de tous les mots binaires de longueur r non-nul.*

On peut démontrer que pour tout code de Hamming, nous avons $\delta = 3$.

Chapitre 5

Systèmes de numération

Soit un naturel n. Dans le langage courant, cet entier est exprimé dans la base décimale. Dans le cadre de ce chapitre, nous allons expliciter comment représenter ce nombre naturel dans d'autres systèmes de numération (section 5.1 à 5.3). Nous nous attardons sur le système binaire en présentant quelques opérations particulières dans la section 5.4. Enfin, dans les sections 5.5 à 5.7, nous terminons ce chapitre en présentant les notations en virgule fixe et flottante utilisées pour la représentation des nombres réels sur les machines.

5.1 Représentation des entiers

Nous définissons un système de numération pour ensuite affirmer l'unicité d'une représentation.

Définition 5.1

Un système de numération *d'un nombre naturel est constitué*
- *d'une* base $B \in \mathbb{N} : B > 1$,
- *d'un ensemble de* B symboles, *appelés également* chiffres. *Chacun représente un entier compris entre* 0 *et* $B - 1$.

Le théorème suivant est donné sans démonstration.

Théorème 5.1

Soit $x \in \mathbb{N}$ *et un système de numération de base* B. *Alors il existe des entiers* $n, a_0, a_1, \ldots, a_n$ *compris entre 0 et* $B - 1$ *tels que*

$$x = a_n B^n + a_{n-1} B^{n-1} + \ldots a_1 B^1 + a_0 B^0. \tag{5.1}$$

De plus, ces entiers $n, a_0, a_1, \ldots, a_n$ *sont uniques si on prend la convention que* a_n *n'est pas nul.*

Grâce à un système de numération, tout entier peut maintenant être représenté de façon unique avec un nombre fini de symboles. Depuis toujours,

nous travaillons en base 10. Les opérations d'addition et de multiplication en base 10 ont été intégrées jeune et sont devenues naturelles.

5.2 Conversion d'une base B à une base B'

Le passage d'une écriture dans une base à une écriture dans une autre base s'appelle *la conversion*. Voici les étapes de traitement à réaliser systématiquement pour faire ce travail. Soit un nombre naturel n exprimé dans la base B par les chiffres suivants

$$x \to a_n a_{n-1} \dots a_0. \tag{5.2}$$

On désire convertir ce nombre dans la base B'. Nous travaillons dans la notation propre à la base B. Les étapes à réaliser sont

1. On divise x par B', puis le quotient obtenu par B' jusqu'à obtenir 0 comme quotient.
2. Chaque reste est converti, en les écrivant dans la base B'. On utilise pour cela une table de conversion où chaque symbole de la base B trouve son symbole correspondant dans la base B' lorsque $B' > B$.
3. Les restes sont présentés de gauche à droite en démarrant par le dernier reste obtenu.

Attention, la division de deux nombres se fait dans une même base.

Exemple 5.1 1. Soit en base 10 : 1324. On propose de convertir ce nombre en base 2. On réalise l'étape 1 dans le tableau suivant. Le nombre de la colonne de gauche 1324 est divisé par 2, son quotient et son reste sont indiqués. On recommence ce travail avec le quotient obtenu jusqu'à ce que celui-ci vaut 0.

Nombre	Quotient	Reste
1324	662	0
662	331	0
331	165	1
165	82	1
82	41	0
41	20	1
20	10	0
10	5	0
5	2	1
2	1	0
1	0	1

Chaque reste obtenu (colonne de droite) est exprimé dans la base 2, ce qui est direct ici puisque $2 < 10$. On obtient alors le nombre en base 2, en juxtaposant chaque reste en commençant par le dernier. On obtient alors

$$1324_{10} = 10100101100_2 \tag{5.3}$$

où la notation a_b exprime le nombre a dans la base b.

2. 1324 en base 10 à convertir en base 5. Nous obtenons le tableau des divisions successives

Nombre	Quotient	Reste
1324	264	4
264	52	4
52	10	2
10	2	0
2	0	2

Ainsi, nous obtenons

$$1324_{10} = 20244_5. \tag{5.4}$$

3. 1324 en base 10 à convertir en base 13. Nous obtenons le tableau des divisions successives

Nombre	Quotient	Reste
1324	101	11
101	7	10
7	0	7

Ainsi, nous obtenons

$$1324_{10} = 7AB_{13}, \tag{5.5}$$

en utilisant la table de conversion suivante

B = 10	0	1	2	3	4	5	6	7	8	9	10	11	12
B = 13	0	1	2	3	4	5	6	7	8	9	A	B	C

On remarque ici la notation utilisée. En effet, en base 13, les chiffres utilisés s'étendent de 0 à 12. Pour bien comprendre qu'il s'agit d'un chiffre et non du nombre 12 exprimé en base 10, on prend la convention de noter 10 avec la lettre A, 11 avec la lettre B et ainsi de suite. D'où la notation 7AB pour exprimer 1324 en base 13.

4. Soit 123002_4, écrivons le en base 10. Comme mentionné plus haut, nous devons réaliser la division de 123002_4 par 22_4 (soit 10_{10}). Pour ce faire considérons la table de multiplication de 22_4, nous avons

Itération	Produit
0	0_4
1	22_4
2	110_4
3	132_4

Remarque 5.1 En effet, 22_4, par exemple, est

$$22_4 = 2 \cdot 4^1 + 2 \cdot 4^0 \tag{5.6}$$
$$= 10_{10} \tag{5.7}$$

Cette table est nécessaire pour réaliser l'étape 1 de la conversion, celle-ci donne

Nombre	Quotient	Reste
123002	2231	0
2231	101	3
101	1	13
1	0	1

La difficulté pour obtenir ce tableau réside dans le fait qu'il faut être capable de diviser 123002_4 par 22_4. Le calcul écrit de cette division est le suivant

$$\begin{array}{cccccccc|cccc} 1 & 2 & 3 & 0 & 0 & 2 & 2 & 2 & & & \\ \hline - & 1 & 1 & 0 & & & & 2 & 2 & 3 & 1 \\ \hline & & 1 & 3 & 0 & & & & & & \\ - & & 1 & 1 & 0 & & & & & & \\ \hline & & & 2 & 0 & 0 & & & & & \\ - & & & 1 & 3 & 2 & & & & & \\ \hline & & & & & 2 & 2 & & & & \\ - & & & & & 2 & 2 & & & & \\ \hline & & & & & & 0 & & & & \end{array} \tag{5.8}$$

Nous utilisons pour cela la table de multiplication de 22_4. Ainsi en prenant les 3 premiers chiffres de 123002_4, soit 123, comme $22_4 \times 2$ donne 110_4, nous avons pour le premier nombre un quotient 2 et un reste de 13. On abaisse ensuite 0 et on reprend la division. A la suite de la troisième étape, il s'agit de soustraire 132_4 à 200_4. Or nous avons que

$$132_4 + 2_4 = 200_4. \tag{5.9}$$

Remarquons ensuite que le troisième reste, à savoir 13, est écrit en base 4, en base 10, ce reste vaut 7. Nous avons donc

$$123002_4 = 1730_{10}. \tag{5.10}$$

5. Soit 123002_4, écrivons le en base 7, soit 13_4. Considérons dès lors, la table de multiplication de 13_4, nous avons

Itération	Produit
0	0_4
1	13_4
2	32_4
3	111_4

Divisons à présent 123002_4 par 13_4,

$$\begin{array}{cccccccc|cccc} 1 & 2 & 3 & 0 & 0 & 2 & 1 & 3 & & & \\ \hline - & 1 & 1 & 1 & & & & 3 & 3 & 1 & 3 \\ \hline & & 1 & 2 & 0 & & & & & & \\ - & & 1 & 1 & 1 & & & & & & \\ \hline & & & & 3 & 0 & & & & & \\ - & & & & 1 & 3 & & & & & \\ \hline & & & & 1 & 1 & 2 & & & & \\ - & & & & 1 & 1 & 1 & & & & \\ \hline & & & & & & 1 & & & & \end{array} \tag{5.11}$$

Nous avons bien que $30_4 - 13_4$ donne 11_4 car

$+$	1	2	3	10	11
13_4	20	21	22	23	30

L'étape 1 de la conversion donne ainsi

Nombre	Quotient	Reste
123002	3313	1
3313	203	2
203	11	0
11	0	11

Attention, le quatrième reste, à savoir 11, est écrit en base 4. En base 7, ce reste vaut 5, comme l'indique la table de conversion suivante.

Base 4	Base 7
0	0
1	1
2	2
3	3
10	4
11	5
12	6

Nous avons donc

$$123002_4 = 5021_7. \tag{5.12}$$

6. Ecrivons 1150_{13} en base 4. Les symboles manipulés en base 13 sont

   0 1 2 3 4 5 6 7 8 9 A B C

   4 en base 13 s'écrit donc via le même symbole. Sa table de multiplication exprimée en base 13 donne

Itérations	Produits
0	0
1	4
2	8
3	C
4	13
5	17
6	1B
7	22
8	26
9	2A
A	31
B	35
C	39

Ainsi divisant 1150 par 4 donne

$$\begin{array}{rcccc|l} & 1 & 1 & 5 & 0 & 4 \\ - & & \mathsf{C} & & & \hline & & 2 & 5 & \\ - & & 2 & 2 & & \\ \hline & & & 3 & 0 & \\ - & & & 2 & \mathsf{A} & \\ \hline & & & & 3 & \end{array} \qquad \begin{array}{l} 3\ 7\ 9 \\ \\ \\ \\ \\ \\ \\ \end{array} \tag{5.13}$$

On obtient finalement

Nombre	Quotient	Reste
1150	379	3
379	B8	3
B8	2B	3
2B	9	1
9	2	1
2	0	2

soit le nombre 211333_4.

7. Supposons que 1150 soit écrit en base 10. On propose de le convertir en base 14. Nous avons la table de conversion suivante

$B = 10$	0	1	2	3	4	5	6	7	8	9	10	11	12	13
$B' = 14$	0	1	2	3	4	5	6	7	8	9	A	B	C	D

On réalise alors l'opération de division

$$\begin{array}{rcccc|l} & 1 & 1 & 5 & 0 & 14 \\ - & 1 & 1 & 2 & & \hline & & & 3 & 0 & \\ - & & & 2 & 8 & \\ \hline & & & & 2 & \end{array} \qquad \begin{array}{l} 8\ 2 \\ \\ \\ \\ \\ \\ \end{array} \tag{5.14}$$

Ainsi, on obtient finalement la table

Nombre	Quotient	Reste
1150	82	2
82	5	12
5	0	5

soit le nombre $5C2_{14}$ en notant que 12 est représenté par le symbole C en base 14.

Lorsque B' (la base vers laquelle on opère la conversion) est une puissance entière de la base de départ B, soit

$$B' = B^k, \tag{5.15}$$

pour $k \in \mathbb{N}$, on a une règle de conversion plus simple. Celle-ci devient

1. découper le nombre n à convertir, en tranches de k chiffres, à partir de la droite.

2. Convertir le nombre obtenu dans chaque tranche par son correspondant dans la nouvelle base.

Exemple 5.2 1. Parlons de deux bases particulières, la base 2 ou *binaire* et la base 16 ou *hexadécimale*. Soit le nombre $n = 1111001010001_2$. Comme $16 = 2^4$, on découpe le nombre en tranches de 4 chiffres, on obtient

$$n = 1 \quad 1110 \quad 0101 \quad 0001 \tag{5.16}$$

soit en notation hexadécimale

$$n = 1 \quad 14 \quad 5 \quad 1 \tag{5.17}$$
$$= 1E51_{16}. \tag{5.18}$$

2. Voici une adresse MAC exprimée en binaire

$$00000000 : 00000011 : 10111010 : 00100110 : 00000001 : 10110000 \tag{5.19}$$

soit en format hexadécimal

$$00 : 03 : BA : 26 : 01 : B0 \tag{5.20}$$

3. Soit en base 3 le nombre 1221120_3. Pour le convertir en base 9, nous réalisons la découpe suivante

$$1\,|\,22\,|\,11\,|\,20 \tag{5.21}$$

chaque bloc est converti en base 9 ce qui donne

$$1\,|\,8\,|\,4\,|\,6 \tag{5.22}$$

puisque

$B = 3$	0	1	2	10	11	12	20	21	22
$B' = 9$	0	1	2	3	4	5	6	7	8

4. Soit 32123_4 converti en base 16 on obtient 39B puisque

$B = 4$	10	11	12	13	20	21	22	23	30	31	32	33
$B' = 16$	4	5	6	7	8	9	A	B	C	D	E	F

5. Soit $AA2B1C_{16}$. En base 4, nous avons

A	A	2	B	1	C
22	22	02	23	01	30

soit 222022301304.

Avant de présenter quelques représentations possibles pour les nombres négatifs, voici un résultat qui explique une borne au nombre de symboles nécessaires pour représenter n'importe quel entier x dans une base B.

Théorème 5.2

Le nombre de symboles nécessaires pour représenter tout naturel x *dans la base* B *est le plus petit entier strictement supérieur à* $\mathrm{Log}_B(x)$.

Démonstration : Soit $n+1$ le nombre de symboles nécessaires pour représenter x dans la base B. La démonstration est alors immédiate lorsqu'on note que

$$B^n \leq x \leq B^{n+1}. \tag{5.23}$$

Dès lors

$$n \leq \mathrm{Log}_B(x) \leq n+1, \tag{5.24}$$

ce qui donne le résultat attendu. ■

On peut donc dire que ce nombre de symboles est à peu près proportionnel au logarithme du nombre dans la base en question.

5.3 Représentation des nombres négatifs

La représentation des nombres entiers dépend du langage de programmation utilisé. Habituellement, on travaille en codage binaire, avec un certain nombre de bits pour représenter un nombre. Comme ce nombre de bits est fini, il existe un maximum et un minimum pour représenter un entier.

Au cours du temps, plusieurs méthodes ont été proposées pour représenter les nombres négatifs. Nous présentons ces méthodes.

Utilisation du bit de signe

Il s'agit simplement ici de réserver un bit dans la représentation du nombre pour indiquer si il est positif ou négatif.

Exemple 5.3 Soit un système où cinq bits sont utilisés pour représenter un entier, le premier bit (à gauche) marqué de 1 indique que le nombre est positif, 0 que le nombre est négatif. Ainsi

Représentation binaire	décimale
10010	2
10101	5
01111	-15
11111	15

Cette représentation a un gros désavantage : le calcul binaire (et en particulier l'addition) devient beaucoup plus lourd.

La méthode du complément à 1

Le principe est ici d'inverser tous les bits, soit de prendre le complément à 1 pour chaque bit.

Exemple 5.4 Soit un système où quatre bits sont utilisés pour représenter un entier. On pourra avec cette méthode représenter huit chiffres seulement. Ainsi, avec la méthode du complément à 1

décimale positive	Représentation binaire positive	négative
0	0000	1111
1	0001	1110
2	0010	1101
3	0011	1100
4	0100	1011
5	0101	1010
6	0110	1001
7	0111	1000

Cette méthode a été rapidement abandonnée. Considérons les sommes suivantes

$$\text{décimal :} \quad (-1) + 1 = 0 \tag{5.25}$$
$$\text{binaire :} \quad 1110 + 0001 = 1111 \tag{5.26}$$
$$\text{décimal :} \quad 1 - (1) = 0 \tag{5.27}$$
$$\text{binaire :} \quad 1110 - 1110 = 0000 \tag{5.28}$$

ce qui donne deux représentations binaires pour le nombre 0.

Méthode du complément à 2

Le principe est de
— prendre le complément à 1,
— ajouter 1.

Exemple 5.5 Soit un système où quatre bits sont utilisés pour représenter un entier. A nouveau, seul 8 chiffres peuvent être représentés. Nous avons

décimale	Représentation binaire	son opposé
0	0000	0000
1	0001	1111
2	0010	1110
3	0011	1101
4	0100	1100
5	0101	1011
6	0110	1010
7	0111	1001

et ainsi de suite.

5.4 Cas particulier : le calcul binaire

Dans cette section, nous nous intéressons à la représentation binaire et aux règles qui s'appliquent pour réaliser aisément l'addition, la soustraction, la multiplication et la division de deux nombres binaires. Nous travaillons avec une représentation des nombres négatifs qui correspond à la méthode du complément à 2.

Addition/Soustraction

Nous avons

$$0+0=0 \tag{5.29}$$
$$1+0=1 \tag{5.30}$$
$$1+1=10 \tag{5.31}$$

Dès lors, nous avons dans un système où 4 bits sont utilisés (tout en sachant qu'on ne pourra représenter que 7 chiffres),

$$\begin{array}{rrcccc} 3 & & 0 & 0 & 1 & 1 \\ 4 & + & 0 & 1 & 0 & 0 \\ \hline 7 & & 0 & 1 & 1 & 1 \end{array} \qquad \begin{array}{rrcccc} -1 & & 1 & 1 & 1 & 1 \\ -2 & + & 1 & 1 & 1 & 0 \\ \hline -3 & & 1 & 1 & 0 & 1 \end{array}$$

$$\begin{array}{rrcccc} -2 & & 1 & 1 & 1 & 0 \\ -5 & + & 1 & 0 & 1 & 1 \\ \hline -7 & & 1 & 0 & 0 & 1 \end{array} \qquad \begin{array}{rrcccc} -2 & & 1 & 1 & 1 & 0 \\ 3 & + & 0 & 0 & 1 & 1 \\ \hline 1 & & 0 & 0 & 0 & 1 \end{array} \tag{5.32}$$

On remarque l'utilisation du report, excepté sur le bit gauche.

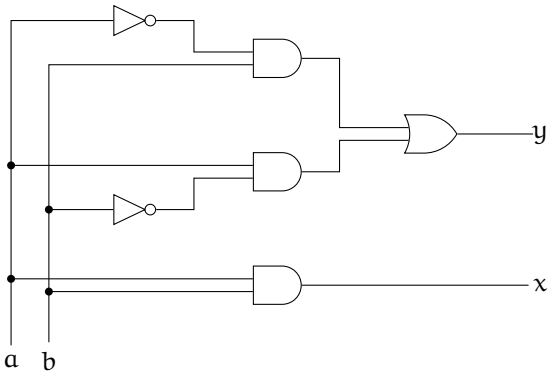
Lorsque la somme des deux membres, donne un nombre qui ne peut être représenté dans le système de numération, la convention est d'utiliser alors le plus grand nombre que l'on peut représenter. Imaginez alors les erreurs de calculs et la propagation de celles-ci.

Exemple 5.6 On peut faire le lien entre le calcul binaire, et l'électronique via la représentation des circuits logiques. En effet, l'addition de 2 bits correspond à un dispositif électronique, l'*additionneur 2 bits*. Celui-ci correspond aux additions suivantes

$$0 + 0 = 00 \tag{5.33}$$
$$1 + 0 = 01 \tag{5.34}$$
$$0 + 1 = 01 \tag{5.35}$$
$$1 + 1 = 10 \tag{5.36}$$

On le représente comme le circuit logique suivant

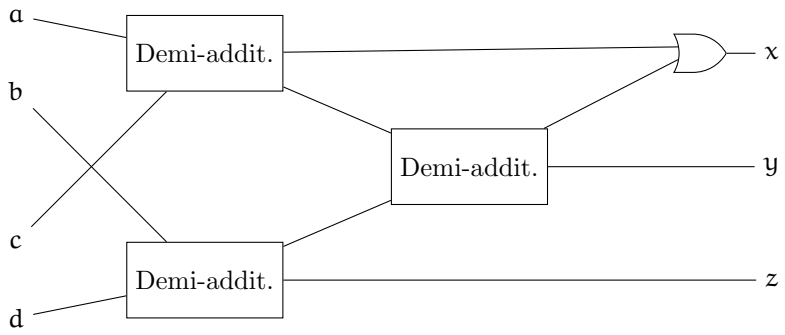
cet élément de circuit peut être représenté logiquement par le circuit suivant



avec

$$a + b = xy \tag{5.37}$$

Pour représenter les additions de deux bits, nous avons le circuit suivant



en notant que

$$ab + cd = xyz. \tag{5.38}$$

Multiplication

La multiplication par une puissance de 2 est simple. Il suffit de décaler vers la gauche la représentation du nombre d'autant de bit que la puissance de 2.

$$\begin{aligned} 5_{10} \times 32_{10} &= 101_2 \times 100000_2 && (5.39)\\ &= 10100000_2 && (5.40) \end{aligned}$$

Lorsqu'il ne s'agit pas d'un nombre puissance de 2, il convient alors de décomposer le produit.

$$\begin{aligned} 5_{10} \times 9_{10} &= (4_{10} + 1_{10}) \times 9_{10} && (5.41)\\ &= 4_{10} \times 9_{10} + 1_{10} \times 9_{10} && (5.42)\\ &= 100_2 \times 1001_2 + 1_2 \times 1001_2 && (5.43)\\ &= 100100_2 + 1001_2 && (5.44)\\ &= 101101_2 && (5.45) \end{aligned}$$

Division euclidienne

Lorsqu'on désire diviser un nombre par une puissance de 2, il suffit de déplacer la virgule d'autant de bits que la valeur de l'exposant ($4 = 2^2$). Ainsi,

$$\begin{aligned} 15_{10} \,/\, 4_{10} &= 1111_2 \,/\, 100_2 && (5.46)\\ &= 11,11_2 && (5.47) \end{aligned}$$

soit $11_2 = 3_{10}$ pour le quotient et derrière la virgule $11_2 = 3_{10}$ comme reste.

Lorsqu'il s'agit de nombre négatif, en considérant la méthode du complément à 2, nous avons la démarche suivante. Considérons pour cet exemple une représentation avec cinq bits, ce qui donne

décimale	Représentation binaire	son opposé
0	00000	00000
1	00001	11111
2	00010	11110
3	00011	11101
4	00100	11100
5	00101	11011
6	00110	11010
7	00111	11001
8	01000	11000
9	01001	10111
10	01010	10110
11	01011	10101
12	01100	10100
13	01101	10011
14	01110	10010
15	01111	10001

Ainsi,

$$-13_{10} / 4_{10} = 10011_2 / 100_2 \tag{5.48}$$

$$= 11100,11_2 \tag{5.49}$$

on déplace la virgule vers la gauche d'autant de position que la valeur de l'exposant ($4 = 2^2$) et on ajoute autant de 1 à gauche. Ainsi, on obtient 11100 pour la représentation sur 5 bits, avec le principe de complément à 2, ce qui donne -4 et 11_2 pour le reste, soit 3. Ceci est correct puisque

$$-13_{10} / 4_{10} = (-4) \times 4 + 3, \tag{5.50}$$

avec un reste positif.

Pour une division d'un entier positif avec un nombre quelconque toujours positif, on peut procéder comme pour une division classique, en notant que

$$1 \div 1 = 1 \tag{5.51}$$

$$0 \div 1 = 0 \tag{5.52}$$

Ainsi lorsqu'on réalise la division $47_{10} \div 13_{10}$, on obtient en base 2

$$\begin{array}{cccccccc|cccc} & 1 & 0 & 1 & 1 & 1 & 1 & & 1 & 1 & 0 & 1 \\ - & & 1 & 1 & 0 & 1 & & & 1 & 1 & & \\ \hline & & 1 & 0 & 1 & 0 & 1 & & & & & \\ - & & & 1 & 1 & 0 & 1 & & & & & \\ \hline & & & 1 & 0 & 0 & 0 & & & & & \end{array} \tag{5.53}$$

soit $47_{10} \div 13_{10}$ est égal à 11_2 = (3 fois 13) plus 1000_2(= 8, comme reste).

Calcul de pgcd

L'objectif est à présent de calculer le pgcd de deux nombres binaires a et b. L'algorithme proposé est un algorithme adapté de l'algorithme d'Euclide, utilisant les propriétés suivantes.

Propriété 5.1

Soit a et b deux nombres binaires.

1. *On observe que*

$$a\underbrace{000\ldots0}_{k} = a \cdot 10^k, \tag{5.54}$$

   avec a impair.

2. *Soit a et b impairs, alors*

$$\text{pgcd}(a \cdot 10^p, b \cdot 10^q) = \text{pgcd}(a, b) \cdot 10^r, \tag{5.55}$$

   où $r = \inf(p, q)$.

3. *On observe*

$$\text{pgcd}(a, b) = \text{pgcd}(a - b, b). \tag{5.56}$$

Ainsi, l'algorithme pour calculer $\text{pgcd}(a, b)$ est le suivant

1. On supprime les zéros communs qui terminent a et b, nous avons alors

$$\text{pgcd}(a, b) = H \cdot 10^r, \tag{5.57}$$

   où r est le nombre de zéros communs.

2. On supprime les zéros qui terminent l'un des deux nombres.
3. Les nombres restant se terminent tous deux par 1. Deux cas se présentent
   — $(a = b) \Rightarrow \text{pgcd}(a, b) = a$,
   — $a \neq b$, on passe à l'étape suivante.
4. On reprend l'étape 1 avec $\text{pgcd}(\max(a, b) - \min(a, b), \min(a, b))$.

Exemple 5.7 Calculons $\text{pgcd}(110011001000, 1010100100)$. Nous avons les étapes suivantes

— Le nombre de zéros communs qui terminent a et b est 2. Dès lors, nous cherchons

$$\text{pgcd}(a, b) = \text{pgcd}(1100110010, 10101001) \cdot 10^2. \tag{5.58}$$

— Ensuite,

$$\begin{aligned}
&\text{pgcd}(1100110010, 10101001)\\
&\quad = \text{pgcd}(110011001, 10101001) && \text{par l'étape 2.} && (5.59)\\
&\quad = \text{pgcd}(11110000, 10101001) && \text{par l'étape 4.} && (5.60)\\
&\quad = \text{pgcd}(1111, 10101001) && \text{par l'étape 2.} && (5.61)\\
&\quad = \text{pgcd}(10011010, 1111) && \text{par l'étape 4.} && (5.62)\\
&\quad = \text{pgcd}(1001101, 1111) && \text{par l'étape 2.} && (5.63)\\
&\quad = \text{pgcd}(111110, 1111) && \text{par l'étape 4.} && (5.64)\\
&\quad = \text{pgcd}(11111, 1111) && \text{par l'étape 2.} && (5.65)\\
&\quad = \text{pgcd}(10000, 1111) && \text{par l'étape 4.} && (5.66)\\
&\quad = \text{pgcd}(1111, 1) && \text{par l'étape 2.} && (5.67)\\
&\quad = 1 && && (5.68)
\end{aligned}$$

— Ainsi

$$\begin{aligned}
\text{pgcd}(a, b) &= \text{pgcd}(1100110010, 10101001) \cdot 10^2 && (5.69)\\
&= 100. && (5.70)
\end{aligned}$$

5.5 Représentation des réels

A présent, nous nous intéressons à la représentation des nombres réels. Certains réels nécessitent pour leur représentation une infinité de chiffres selon la base B choisie.

Exemple 5.8 Le réel $1/3$ est représenté par $0,333\ldots$ en base 10 et par $0,1$ en base 3.

Nous avons le théorème suivant.

Théorème 5.3

Soit x un nombre réel strictement positif. Alors il existe une suite d'entiers $A_n, A_{n-1}, \ldots, A_1, A_0, a_1, a_2, \ldots$ compris entre 0 et $B-1$ tels que la différence

$$x - (A_n B^n + A_{n-1} B^{n-1} + \ldots + A_0 + a_1 B^{-1} + \ldots + a_k B^{-k}) \qquad (5.71)$$

est positive et tend vers 0 lorsque k tend vers l'infini.

Le nombre x dans la base B a donc la représentation suivante

$$x = A_n A_{n-1} \ldots A_0, a_1 a_2 \ldots \qquad (5.72)$$

La convention est de choisir $A_n \neq 0$, ce qui implique que n est bien défini. Pour le nombre de chiffres après la virgule, nous avons le théorème suivant.

Théorème 5.4

Lorsque x *est de la forme*

$$\frac{a_p}{B^p}, \tag{5.73}$$

avec a_p *et* p *entiers, alors il existe deux écritures en base* B *pour* x. *Celles-ci sont*

1. *pour la première : pour tout* $k > p : a_k = 0$.
2. *Pour la seconde :* $a'_p = a_p - 1$ *et* $k > p : a_k = B - 1$.

Lorsque x n'est pas de la forme (5.73), il n'existe alors qu'une unique écriture de la forme (5.72).

Exemple 5.9 Soit $x = 2/100$ (ou encore $2/10^2$), alors

$$x = 0,0200000\ldots \tag{5.74}$$
$$= 0,0199999\ldots \tag{5.75}$$

Dans la suite, nous proposons deux notations pour la représentation des nombres réels. La première la *notation en virgule fixe* utilise directement le résultat du théorème 5.3. La seconde, la *notation en virgule flottante* permet de représenter un nombre plus large de réels.

5.6 Notation en virgule fixe des nombres réels

Définition 5.2

Soit x *un nombre réel non nul. Sa* représentation en virgule fixe *est*

$$\{[x_n x_{n-1}\ldots x_1 x_0, x_{-1}x_{-2}\ldots x_{-m}], b, s\}, \tag{5.76}$$

où

- $b \in \mathbb{N}, b \geq 2$, *est appelée la* base,
- $s \in \{0,1\}$ *est appelé le* signe,
- $x_i \in \mathbb{N}, 0 \leq x_i < b, i = -m,\ldots,n$ *sont les* symboles,
- m *désigne le nombre de chiffres après la virgule,*
- $n+1$ *est le nombre de chiffres avant la virgule.*

On a donc pour $x \in \mathbb{R}$,

$$x = (-1)^s \left(\sum_{k=-m}^{n} x_k b^k \right). \tag{5.77}$$

Exemple 5.10 1. Soit en notation décimale le réel $227,375$. Sa notation en

virgule fixe est

$$\{[227,375],10,0\} \tag{5.78}$$
$$\{[11100011,011],2,0\} \tag{5.79}$$
$$\{[3203,12],4,0\} \tag{5.80}$$

2. Soit la notation $[111,011]$, elle correspond

$$\text{en base } 2 : 2^2+2^1+2^0+2^{-2}+2^{-3} = 7,375 \tag{5.81}$$
$$\text{en base } 10 : 10^2+10^1+10^0+10^{-2}+10^{-3} = 111,011 \tag{5.82}$$
$$\text{en base } 4 : 4^2+4^1+4^0+4^{-2}+4^{-3} = 21,078125 \tag{5.83}$$

Il convient de faire attention au nombre de chiffres significatifs nécessaires pour représenter un nombre x. Par exemple, le nombre réel $x = 0.1$ a une représentation finie en base $b = 10$ tandis qu'il demande un nombre infini de chiffres significatifs pour la base $b = 2$. Reprenez également l'exemple 5.8 pour l'observer.

Les ordinateurs emploient souvent trois bases : $b = 10$ la base décimale, $b = 2$ la base binaire, $b = 16$ la base hexadécimale.

5.7 Notation en virgule flottante des nombres réels

Définition 5.3

Etant donné un nombre réel non nul x, *sa* représentation en virgule flottante *est*

$$\{[a_1 a_2 \dots a_t], e, b, s\}, \tag{5.84}$$

où

- $b \in \mathbb{N}, b \geq 2$ *est appelé la* base,
- $e \in \mathbb{Z}, L \leq e \leq U$ *est appelé l'*exposant,
- $s \in \{0,1\}$ *est le* signe.
- t *est le nombre de chiffres significatifs,*
- $a_i \in \mathbb{N}, 0 < a_1 < b, 0 \leq a_i < b, i = 2,\dots,t$
- *la quantité*

$$m = m(x) = \sum_{i=1}^{t} a_i b^{t-i} \tag{5.85}$$

est appelée mantisse.
Cette notation est utilisée pour encoder le nombre réel

$$x = (-1)^s b^e \sum_{i=1}^{t} a_i b^{-i} = (-1)^s m b^{e-t} \tag{5.86}$$

Remarque 5.2 Les quantités L et U sont fixées par les contraintes de la machine à savoir le nombre de bits utilisés pour représenter e. Considérez le premier cas étudié dans l'exemple 5.12.

Exemple 5.11 Considérons l'exemple suivant $b = 4, e = 3$ et $t = 5$. Soit le nombre représenté par [21301]. Calculons la mantisse

$$m = 2 \cdot b^{5-1} + 1 \cdot b^{5-2} + 3 \cdot b^{5-3} + 0 \cdot b^{5-4} + 1 \cdot b^{5-5} \tag{5.87}$$
$$= 625. \tag{5.88}$$

Dès lors, le nombre [21301] en notation décimale est le nombre $39,0625$ en utilisant l'équation (5.86).

On parle de notation en virgule *flottante* car le nombre de chiffres après la virgule n'est pas fixé par la notation. Tout dépend de l'exposant e. Dans l'exemple précédent, si e était égal à 10, on obtenait 640000. La mantisse reste constante, comme la base. On "flotte" simplement dans l'ensemble des réels selon la valeur de l'exposant e.

Définition 5.4

On parle de notation à virgule flottante normalisée *lorsque* $a_1 > 0$ *dans l'écriture (5.84).*

Dès lors, la mantisse est telle que

$$b^{t-1} \leq m \leq b^t - 1. \tag{5.89}$$

En effet, la plus petite valeur s'obtient en choisissant la représentation minimale soit $[1000\ldots0]$, et la plus grande en prenant $[(b-1)(b-1)\ldots(b-1)]$. Par induction, on prouve aisément que

$$\sum_{i=1}^{t} (b-1)b^{t-i} = b^t - 1. \tag{5.90}$$

Obtenir la notation en virgule flottante

La notation en virgule flottante a en fait pour objectif de ramener tout nombre à un nombre avec 0 devant la virgule, multiplié par une puissance appropriée de la base. Pour clarifier notre propos, nous proposons d'expliquer la méthode pour obtenir une notation en virgule flottante à partir de deux exemples.

Exemple 5.12 1. Travaillons dans la base 2 ($b = 2$) avec $t = 23$ chiffres (soit une mantisse codée sur maximum 23 bits), 1 bit utilisé pour le signe (égal donc à 0 ou 1), et 8 bits utilisés pour l'exposant.

Pour représenter un exposant négatif, on prendra la convention que le premier bit de l'exposant est le bit de signe de l'exposant, avec 0 pour un exposant négatif et 1 pour un exposant positif.

Dès lors le nombre 124.5625 en format binaire s'exprime comme

$$1111100.1001 \tag{5.91}$$

soit, en ayant pour objectif de tout ramener à un nombre avec un seul chiffre devant la virgule

$$0.11111001001\,10^{111} \tag{5.92}$$

où la base et l'exposant sont écrits eux-même en format binaire. Dès lors, la notation en virgule flottante est

$$\{[a_1 \dots a_{11}], e, b, s\} = \{[11111001001], 10000111, 10, 0\} \tag{5.93}$$

la mantisse étant codée sur 11 bits, et l'exposant sur 8 bits. En effet, on obtient pour la mantisse en suivant l'expression (5.85)

$$\sum_{i=1}^{t} a_i b^{t-i} = b^{10} + b^9 + b^8 + b^7 + b^6 + b^3 + 1 \tag{5.94}$$

qui multiplié par $b^{e-t} = b^{7-11}$, donne le résultat espéré.

2. Prenons encore un autre exemple. Soit le nombre 127.421, travaillons en base 10, avec $t = 9$. Dès lors la notation en virgule flottante est

$$\{[127421000], 3, 10, 0\} \tag{5.95}$$

En effet, en notation scientifique (soit $0, \dots \cdot 10^e$), on a

$$127.421 = 0.127421\,10^3 \tag{5.96}$$

De plus, on peut également vérifier en calculant la mantisse et en la multlpliant par b^{e-t} qu'on obtient le nombre espéré, soit

$$\begin{aligned}
\{[127421000], 3, 10, 0\} &= \{10^{t-1} + 2\,10^{t-2} + \dots + 0\,10^{t-9}\}10^{e-t} && (5.97)\\
&= \{10^{9-1} + 2\,10^{9-2} + \dots + 0\,10^{9-9}\}10^{3-9} && (5.98)\\
&= 10^2 + 2\,10^1 + \dots + 0\,10^{-6} && (5.99)\\
&= 127.421 && (5.100)
\end{aligned}$$

De manière générale, on peut donc dire que la méthode pour obtenir la notation en virgule flottante, à partir d'un nombre écrit dans une base quelconque, est de

1. transformer le nombre dans la base désirée
2. obtenir une notation scientifique du type $0, \dots \cdot B^e$
3. sachant que t est le nombre de chiffres présent après la virgule,
   — il faut tronquer ce nombre si t est plus petit que ce nombre
   — il faut ajouter des 0 sinon
4. dès lors, e est déterminé en notant qu'il s'agit de l'exposant de la base dans la notation scientifique

La méthode pour obtenir dans la base en question, le nombre noté comme $\{[a_1 \dots a_t], e, b, s\}$, est de calculer la mantisse, soit

$$m = a_1 b^{t-1} + a_2 b^{t-2} + \dots + a_t b^0$$

et de la multiplier par b^{e-t} en considérant le signe déterminé par s.

Annexe A

Alphabet Grec

Minuscule	Majuscule	Nom
α	A	alpha
β	B	beta
γ	Γ	gamma
δ	Δ	delta
ϵ	E	epsilon
ζ	Z	zêta
η	H	êta
θ	Θ	thêta
ι	I	iota
κ	K	kappa
λ	Λ	lambda
μ	M	mu
ν	N	nu
ξ	Ξ	xi
o	O	omicron
π	Π	pi
ρ	P	rhô
σ	Σ	sigma
τ	T	tau
υ	Y	upsilon
ϕ	Φ	phi
ψ	Ψ	psi
ω	Ω	ômégа