Logique élémentaire: calcul propositionnel

par Denis Monasse, Lycée Louis le Grand, Paris

2 décembre 1999

1 Eléments du calcul propositionnel

1.1 Les propositions

Définition 1.1 *On appelle proposition (ou assertion) toute phrase dont il est possible de dire si elle est vraie ou fausse.*

Exemple 1.1
- *Deux plus deux égale quatre* est une proposition qui est vraie.
- *Deux plus deux égale cinq* est une proposition qui est fausse.
- *Aucun homme n'a été sur Mars* est une proposition qui est vraie à l'heure actuelle mais qui peut devenir fausse dans l'avenir
- $x + y > 0$ n'est pas une proposition car le fait qu'elle soit vraie ou fausse dépend des valeurs prises par x et y, donc on ne peut pas dire si cette assertion est vraie ou fausse
- *Cette phrase est fausse* n'est pas une proposition ; en effet, si elle est vraie, c'est qu'elle est fausse et si elle est fausse, c'est qu'elle est vraie ; on ne peut donc lui attribuer aucune valeur logique.

1.2 Propositions composées

A partir d'une proposition p, on peut construire sa *négation* que nous noterons (très provisoirement) NON p. La règle à appliquer est que si p est vraie, alors NON p est fausse et que si p est fausse alors NON p est vraie.

A partir de deux propositions p et q, on peut construire leur *conjonction* que nous noterons provisoirement p ET q. La proposition p ET q est vraie si et seulement si les deux propositions p et q sont vraies, dans tous les autres cas elle est fausse.

A partir de deux propositions p et q, on peut construire leur *disjonction* que nous noterons provisoirement p OU q. La proposition p OU q est vraie si et seulement si l'une au moins des deux propositions p et q est vraie ; sinon elle est fausse.

Définition 1.2 *Si p et q désignent deux propositions, alors*

- *la proposition p IMPL q est vraie si et seulement si*
  - *soit q est vraie*
  - *soit p est fausse*
- *la proposition p EQUIV q est vraie si et seulement p et q sont*
  - *soit simultanément vraies*
  - *soit simultanément fausses*

1.3 Syntaxe des formules logiques

Des variables (en général p, q, r éventuellement indexées en $p_1, \ldots, p_n$, $q_1, \ldots$, $r_1, \ldots$) auxquelles nous pourrons substituer n'importe quelles propositions : *variables propositionnelles*.

Nous remplacerons

- l'opérateur de négation NON par le symbole $\neg$
- l'opérateur de conjonction ET par le symbole $\wedge$
- l'opérateur de disjonction OU par le symbole $\vee$
- l'opérateur d'exclusion OUX par le symbole $\otimes$
- l'opérateur d'implication IMPL par le symbole $\Rightarrow$
- l'opérateur d'équivalence EQUIV par le symbole $\Longleftrightarrow$

Ensemble A formé

- des variables propositionnelles,
- des symboles d'opérateurs logiques des parenthèses ouvrante et fermante,

$$p, q, r, p_1, \ldots, q_1, \ldots, r_1, \ldots, \neg, \wedge, \vee, \otimes, \Rightarrow, \Longleftrightarrow, (,)$$

et l'ensemble A^* des suites finies d'éléments de A (écrites par juxtaposition de gauche à droite).

Définition 1.3 *On appelle ensemble des formules logiques le sous ensemble de A^* défini inductivement par*

- *toute variable propositionnelle est une formule logique*
- *si F est une formule logique $(\neg F)$ est une formule logique*
- *si F_1 et F_2 sont deux formules logiques, alors $(F_1 \wedge F_2)$, $(F_1 \vee F_2)$, $(F_1 \otimes F_2)$, $(F_1 \Rightarrow F_2)$ et $(F_1 \iff F_2)$ sont des formules logiques*

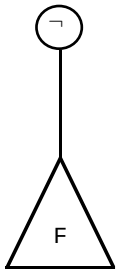
Autorisé : oublier une éventuelle parenthèse ouvrante initiale et la parenthèse fermante finale correspondante.

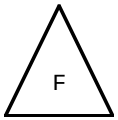
Provisoirement : pas de règle de priorité sur les opérateurs logiques et en conséquence nous ne nous autoriserons pas à supprimer de quelconques parenthèses intermédiaires.

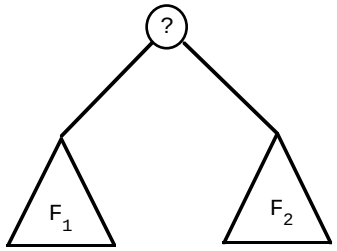
Exemple 1.2
- $((\neg p_1) \vee p_2) \iff (q_1 \wedge q_2)$ est une formule logique
- $\neg p_1 \vee p_2$ n'est pas une formule logique (sans ordre de priorité sur les opérateurs, il y ambiguïté sur la manière de placer des parenthèses : $(\neg p_1) \vee p_2$ ou $\neg(p_1 \vee p_2)$?)
- $p \vee (q \wedge r$ n'est pas une formule logique (manque une parenthèse fermante)

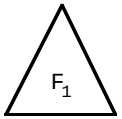
1.4 Représentation arborescente d'une formule logique

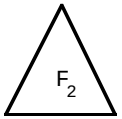
Représenter graphiquement les formules logiques de manière inductive :

- une variable propositionnelle est représentée par son symbole dans un cercle
- si F est une formule logique, $(\neg F)$ est représentée par 

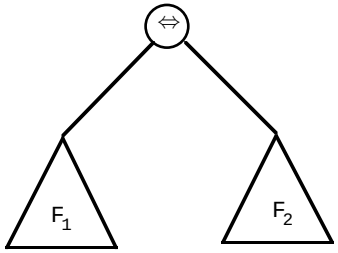
  où 

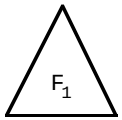
représente la formule F.
- si F_1 et F_2 sont deux formules logiques, alors $(F_1 \wedge F_2)$, $(F_1 \vee F_2)$, $(F_1 \otimes F_2)$, $(F_1 \Rightarrow F_2)$ et $(F_1 \iff F_2)$ sont représentées par 

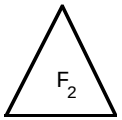
où 

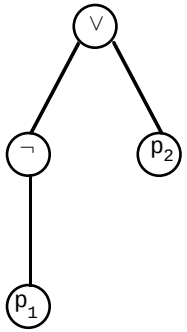
et 

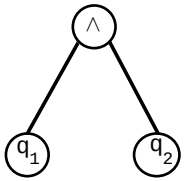
désignent les représentations arborescentes respectives des formules F_1 et F_2 et où le point d'interrogation désigne l'un des opérateurs $\vee, \wedge, \otimes, \Rightarrow, \iff$.

Exemple 1.3 La formule logique $((\neg p_1) \vee p_2) \iff (q_1 \wedge q_2)$ admet la représentation arborescente 

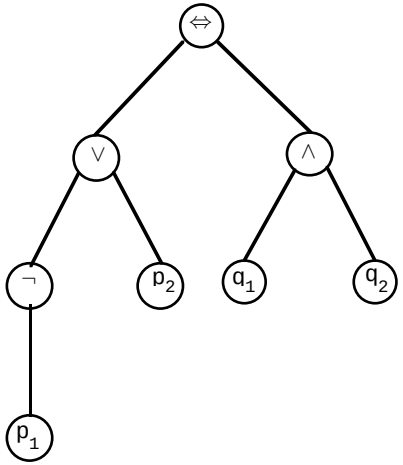
où 

et 

désignent les représentations arborescentes respectives de $(\neg p_1) \vee p_2$ et $q_1 \wedge q_2$, soit encore 

et 

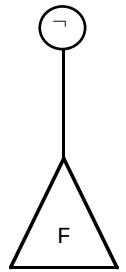
.

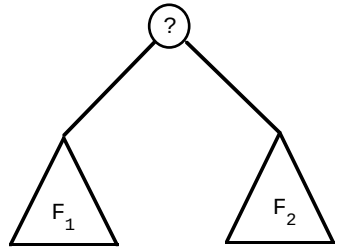
Représentation arborescente 

pour l'expression totale.

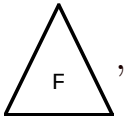
A partir d'un arbre dont les *feuilles*[a] sont des variables propositionnelles, les *noeuds*[b] d'ordre 1 sont *étiquetés* par le symbole $\neg$ et les noeuds d'ordre 2 sont étiquetés par l'un des symboles $\vee, \wedge, \otimes, \Rightarrow$, $\Longleftrightarrow$, il est facile de reconstruire la formule logique de départ en appliquant récursivement les règles

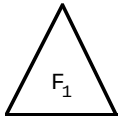
-

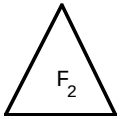
donne la variable propositionnelle p
- 

donne la formule $(\neg F)$
- 

donne la formule $(F_1 \ ? \ F_2)$

lorsque F, F_1 et F_2 désignent les formules représentées par les sous arbres 

, 

et 

et point d'interrogation l'un des opérateurs logiques binaires.

[a] c'est à dire les éléments terminaux

[b] c'est à dire les éléments intermédiaires

1.5 Sémantique : évaluation des formules logiques

A chaque formule logique, nous allons maintenant assigner une signification logique lorsque les variables propositionnelles représentent des propositions susceptibles d'être vraies ou fausses. Pour faciliter le calcul algébrique sur les formules logiques, il est habituel de remplacer les mots *vrai* et *faux* qui sont les deux valeurs prises par les propositions par V et F ou encore mieux par 1 et 0. C'est cette dernière convention que nous suivrons par la suite.

Nous allons commencer par évaluer les opérateurs logiques $\neg, \wedge, \vee, \otimes, \Rightarrow, \Leftrightarrow$ en les rapprochant des opérateurs correspondants introduit précédemment sur les propositions, à savoir NON, ET, OU, OUX, IMPL, EQUIV. A cet effet nous introduirons les fonctions $f_\neg, f_\wedge, f_\vee, f_\otimes, f_\Rightarrow, f_\Leftrightarrow$ définies

- la première sur $\{0,1\}$
- les suivantes sur $\{0,1\}^2$

et à valeurs dans $\{0,1\}$, grâce aux formules suivantes (qui suivent exactement les valeurs de vérité de NON, ET, OU, OUX, IMPL, EQUIV).

$$\begin{cases} f_\neg(0) = 1 \\ f_\neg(1) = 0 \end{cases}, \quad \begin{cases} f_\wedge(1,1) = 1 \\ f_\wedge(1,0) = f_\wedge(0,1) = f_\wedge(0,0) = 0 \end{cases}, \quad \begin{cases} f_\vee(0,0) = 1 \\ f_\vee(1,0) = f_\vee(0,1) = f_\vee(1,1) = 1 \end{cases}$$

$$\begin{cases} f_\otimes(1,1) = f_\otimes(0,0) = 0 \\ f_\otimes(1,0) = f_\otimes(0,1) = 1 \end{cases}, \quad \begin{cases} f_\Rightarrow(1,0) = 0 \\ f_\Rightarrow(1,1) = f_\Rightarrow(0,1) = f_\Rightarrow(0,0) = 1 \end{cases}$$

$$\begin{cases} f_\Leftrightarrow(1,1) = f_\Leftrightarrow(0,0) = 1 \\ f_\Leftrightarrow(1,0) = f_\Leftrightarrow(0,1) = 0 \end{cases}$$

Soit σ une application de l'ensemble des valeurs propositionnelles dans $\{0, 1\}$ (c'est à dire que l'on assigne à chaque variable propositionnelle l'une des valeurs *faux* ou *vrai*).

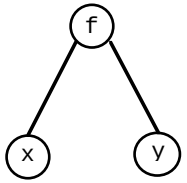
On définit alors facilement la valeur d'une formule logique élémentaire, par exemple par $\text{Val}(p \vee q, \sigma) = f_\vee(\sigma(p), \sigma(q))$.

Nous allons étendre cette évaluation à toute formule logique de manière inductive grâce aux règles suivantes :

- si $F = p$ est une variable propositionnelle, $\text{Val}(F, \sigma) = \sigma(p)$
- si $F = (\neg F_1)$, alors $\text{Val}(F, \sigma) = f_\neg(\text{Val}(F_1, \sigma))$ (autrement dit la valeur de F est l'opposée de la valeur de F_1
- si $F = (F_1 \wedge F_2)$ alors $\text{Val}(F, \sigma) = f_\wedge(\text{Val}(F_1, \sigma), \text{Val}(F_2, \sigma))$
- si $F = (F_1 \vee F_2)$ alors $\text{Val}(F, \sigma) = f_\vee(\text{Val}(F_1, \sigma), \text{Val}(F_2, \sigma))$
- si $F = (F_1 \otimes F_2)$ alors $\text{Val}(F, \sigma) = f_\otimes(\text{Val}(F_1, \sigma), \text{Val}(F_2, \sigma))$
- si $F = (F_1 \Rightarrow F_2)$ alors $\text{Val}(F, \sigma) = f_\Rightarrow(\text{Val}(F_1, \sigma), \text{Val}(F_2, \sigma))$
- si $F = (F_1 \Leftrightarrow F_2)$ alors $\text{Val}(F, \sigma) = f_\Leftrightarrow(\text{Val}(F_1, \sigma), \text{Val}(F_2, \sigma))$

Remarque Cette évaluation des formules logiques est encore équivalente à l'opération suivante (très facile à exécuter à l'aide d'un logiciel de calcul formel) :

- on prend la représentation arborescente de la formule logique ;
- dans cette représentation arborescente, on commence par remplacer dans toutes les feuilles les variables propositionnelles p par leurs valeurs $\sigma(p) \in \{0, 1\}$
- on remplace ensuite dans chaque noeud les symboles logiques $\neg, \wedge, \vee, \otimes, \Rightarrow$ et $\Leftrightarrow$ par la fonction associée
- on simplifie alors l'arbre obtenu avec les conventions que si $x, y \in \{0, 1\}$, 

se simplifie en $f(x)$ et que 

se simplifie en $f(x, y)$

1.6 Tables de vérité

Soit F une formule logique dépendant de n variables propositionnelles distinctes (chacune d'entre elles pouvant apparaître plusieurs fois dans la formule).

A chacune des 2^n applications σ de l'ensemble des variables propositionnelles dans $\{0, 1\}$, on sait donc associer la valeur de F dans l'environnement σ, notée $\mathrm{Val}(F, \sigma) \in \{0, 1\}$.

On regroupe dans un même tableau les différentes valeurs σ attribuées aux variables propositionnelles et l'évaluation de F correspondante.

On construit un tableau à $n + 1$ colonnes et à 2^n lignes. Dans les n premières colonnes on met les valeurs 0 ou 1 attribuées aux valeurs propositionnelles (avec par exemple un ordre lexicographique consistant à faire varier d'abord la n-ième variable propositionnelle, puis la $n - 1$-ième et ainsi de suite jusqu'à la première) et dans la dernière colonne on met la valeur correspondante de $\mathrm{Val}(F, \sigma)$.

Un tel tableau est appelé une table de vérité.

Commençons par construire les tables de vérité des opérations élémentaires sur les variables propositionnelles données par les opérateurs $\neg, \wedge, \vee, \otimes, \Rightarrow$ et $\Leftrightarrow$.

p	$\neg p$
1	0
0	1

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

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

p	q	$p \otimes q$
1	1	0
1	0	1
0	1	1
0	0	0

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

p	q	$p \Leftrightarrow q$
1	1	1
1	0	0
0	1	0
0	0	1

La table de vérité d'une formule logique plus complexe peut alors se construire en introduisant des colonnes intermédiaires pour les sous formules. C'est ainsi que l'on pourra construire en plusieurs étapes la table de vérité de la formule logique

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

de la manière suivante

p	q	r	$\neg p$	$(\neg p) \vee q$	$((\neg p) \vee q) \wedge r$	$p \otimes r$	$(((\neg p) \vee q) \wedge r) \iff (p \otimes r)$
1	1	1	0	1	1	0	0
1	1	0	0	1	0	1	0
1	0	1	0	0	0	0	1
1	0	0	0	0	0	1	0
0	1	1	1	1	1	1	1
0	1	0	1	1	0	0	1
0	0	1	1	1	1	1	1
0	0	0	1	1	0	0	1

1.7 Tautologies, satisfiabilité

On appelle tautologie toute formule logique qui est vraie quelles que soient les valeurs que l'on attribue aux variables propositionnelles. C'est donc une formule dont la table de vérité ne comprend que des 1 dans la dernière colonne, ou encore qui satisfait à la définition suivante.

Définition 1.4 *On dit qu'une formule logique F est une tautologie si pour toute application σ de l'ensemble des variables propositionnelles dans $\{0,1\}$, l'évaluation de F dans l'environnement σ vérifie* $\mathrm{Val}(F,\sigma) = 1$.

Exemple 1.4 La formule logique $(p \Rightarrow q) \iff ((\neg p) \vee q)$ est une tautologie. En effet la table de vérité de cette formule est

p	q	$p \Rightarrow q$	$\neg p$	$(\neg p) \vee q$	$(p \Rightarrow q) \iff ((\neg p) \vee q)$
1	1	1	0	1	1
1	0	0	0	0	1
0	1	1	1	1	1
0	0	1	1	1	1

Une formule est satisfiable si l'on peut attribuer aux variables propositionnelles des valeurs pour lesquelles la formule est vraie. De manière précise, on a la définition suivante

Définition 1.5 *On dit qu'une formule logique F est satisfiable s'il existe une application σ de l'ensemble des variables propositionnelles dans $\{0,1\}$ telle que* $\mathrm{Val}(F,\sigma)=1$.

Une formule qui n'est pas satisfiable est dite contradictoire ; on dit encore que c'est une contradiction. Une formule F est contradictoire si et seulement si la formule $(\neg F)$ est une tautologie. On constate ainsi que le problème de tester si une formule est une tautologie ou de tester si une formule est satisfiable sont intimement reliés.

Moyen évident de tester si une formule logique est satisfiable : construire la table de vérité de la formule et de tester l'existence d'un 1 dans la dernière colonne. Cela revient à donner les 2^n valeurs possibles à la famille des n variables propositionnelles qui interviennent dans la formule et à évaluer la formule dans chacun de ces environnements. Le temps de calcul de cette méthode est clairement proportionnel à 2^n.

On ne connait à l'heure actuelle aucun algorithme réellement plus performant que cet essai systématique et on ne sait même pas s'il peut en exister.

Ce problème de la satisfiabilité d'une formule logique entre dans une catégorie de problèmes dont on peut montrer qu'ils sont en un certain sens équivalents, les problèmes NP-complets, problèmes dont à l'heure actuelle on ne connait que des solutions dont le temps de calcul est exponentiel en la taille des données. Parmi ceux ci on peut citer le problème du déménageur (optimiser le chargement d'un camion avec des colis de différentes tailles) et le problème du voyageur de commerce (trouver le plus court chemin reliant certaines villes données sans repasser deux fois dans la même ville).

2 Fonctions booléennes

2.1 Fonctions booléennes

Définition 2.1 *On appelle fonction booléenne toute application* $f : \{0,1\}^n \to \{0,1\}$.

Définition 2.2 *Soit* F *une formule logique. Soit* $p_1, \ldots, p_n$ *un ensemble ordonné de variables propositionnelles contenant*[a] *toutes les variables propositionnelles intervenant dans* F*. On associe à* F *la fonction booléenne* f_F *de* $\{0,1\}^n$ *dans* $\{0,1\}$ *définie par* $f_F(x_1, \ldots, x_n) = \mathrm{Val}(F, \sigma)$ *où* σ *est définie par* $\forall i \in [1,n],\ \sigma(p_i) = x_i$.

[a] Pour des raisons que l'on verra lorsqu'interviennent plusieurs formules logiques, il vaut mieux ne pas se limiter strictement aux variables intervenant effectivement dans la formule F.

La fonction booléenne associée à une formule logique et à la famille de ses variables propositionnelles se lit directement sur la table de vérité de la formule logique. C'est la fonction qui aux éléments des n premières colonnes associe l'élément correspondant de la dernière colonne.

Exemple 2.1 En reprenant une table de vérité précédente, si $F = (((\neg p) \vee q) \wedge r) \iff (p \otimes r)$, on a la table de vérité

p	q	r	$\neg p$	$(\neg p) \vee q$	$((\neg p) \vee q) \wedge r$	$p \otimes r$	$(((\neg p) \vee q) \wedge r) \iff (p \otimes r)$
1	1	1	0	1	1	0	0
1	1	0	0	1	0	1	0
1	0	1	0	0	0	0	1
1	0	0	0	0	0	1	0
0	1	1	1	1	1	1	1
0	1	0	1	1	0	0	1
0	0	1	1	1	1	1	1
0	0	0	1	1	0	0	1

ce qui donne $f_F(1,0,1) = f_F(0,1,1) = f_F(0,1,0) = f_F(0,0,1) = f_F(0,0,0) = 1$ et $f_F(1,1,1) = f_F(1,1,0) = f_F(1,0,0) = 0$.

Remarque Une formule F est une tautologie si et seulement si $f_F = 1$. Une formule est satisfiable si et seulement si $f_F \neq 0$.

2.2 Equivalence des formules logiques

Définition 2.3 *On dit que deux formules logiques F_1 et F_2 sont équivalentes si pour toute application σ de l'ensemble des variables propositionnelles dans $\{0,1\}$, on a $\mathrm{Val}(F_1,\sigma) = \mathrm{Val}(F_2,\sigma)$. On notera alors $F_1 \equiv F_2$.*

Théorème 2.1 *Soit F_1 et F_2 deux formules logiques, $p_1,\ldots,p_n$ un ensemble ordonné de variables propositionnelles contenant toutes les variables propositionnelles intervenant soit dans F_1 soit dans F_2, f_{F_1} et f_{F_2} les fonctions booléennes associées à F_1 et F_2. On a équivalence de*

1. $F_1 \equiv F_2$
2. $f_{F_1} = f_{F_2}$
3. $(F_1 \Leftrightarrow F_2)$ *est une tautologie*

Démonstration L'équivalence des deux premières assertions est évidente puisque seules les valeurs attribuées à $p_1,\ldots,p_n$ interviennent dans le calcul de $\mathrm{Val}(F_1,\sigma)$ et de $\mathrm{Val}(F_2,\sigma)$. L'équivalence entre la première assertion et la troisième résulte de ce que $\mathrm{Val}(F_1 \Leftrightarrow F_2) = f_{\Leftrightarrow}(\mathrm{Val}(F_1,\sigma),\mathrm{Val}(F_2,\sigma))$ qui d'après la définition de $f_{\Leftrightarrow}$ vaut 1 si et seulement si $\mathrm{Val}(F_1,\sigma) = \mathrm{Val}(F_2,\sigma)$.

Proposition 2.2 *L'équivalence des formules logiques est une relation d'équivalence sur l'ensemble des formules logiques.*

Démonstration La définition même montre que cette relation est à la fois réfléxive, symétrique et transitive.

Théorème 2.3 (invariance de l'équivalence par substitution) *Soit F_1 et F_2 deux formules logiques équivalentes dépendant des variables propositionnelles $p_1, \ldots, p_n$ et soit $G_1, \ldots, G_n$ une famille de n formules logiques. Soit F_1' et F_2' les formules logiques obtenues en remplaçant dans F_1 et F_2 les variables $p_1, \ldots, p_n$ par $G_1, \ldots, G_n$. Alors F_1' et F_2' sont encore équivalentes.*

Démonstration Soit $q_1, \ldots, q_p$ les variables propositionnelles intervenant dans $G_1, \ldots, G_n$. Soit σ une application de $\{q_1, \ldots, q_p\}$ dans $\{0, 1\}$. On démontre immédiatement par induction structurelle sur F_1 que $\mathrm{Val}(F_1', \sigma) = f_{F_1}(Val(G_1, \sigma), \ldots, Val(G_n, \sigma))$. Mais comme F_1 et F_2 sont équivalentes, on a $f_{F_1} = f_{F_2}$, ce qui montre que $\forall \sigma,\ \mathrm{Val}(F_1', \sigma) = \mathrm{Val}(F_2', \sigma)$, donc F_1' et F_2' sont équivalentes.

Remarque On s'autorisera par la suite à écrire 1 pour une tautologie générique et 0 pour une contradiction générique, si bien que

- *F est une tautologie* s'écrira $F \equiv 1$
- *F est une contradiction* s'écrira $F \equiv 0$

2.3 Equivalences fondamentales

Tant qu'on ne s'intéresse qu'à la validité d'un raisonnement ou d'une assertion, on peut remplacer toute formule logique portant sur des variables propositionnelles $p_1, \ldots, p_n$ par une formule logique équivalente. Ceci nous conduit à examiner un certain nombre d'équivalences fondamentales.

Proposition 2.4 *Soit F_1, F_2 et F_3 trois formules logiques. Alors*

- $F_1 \vee F_2 \equiv F_2 \vee F_1$ *et* $F_1 \wedge F_2 \equiv F_2 \wedge F_1$ *(*commutativité *de la disjonction et de la conjonction)*
- $(F_1 \vee F_2) \vee F_3 \equiv F_1 \vee (F_2 \vee F_3)$ *et* $(F_1 \wedge F_2) \wedge F_3 \equiv F_1 \wedge (F_2 \wedge F_3)$ *(*associativité *de la disjonction et de la conjonction)*
- $(F_1 \vee F_2) \wedge F_3 \equiv (F_1 \wedge F_3) \vee (F_2 \wedge F_3)$ *et* $(F_1 \wedge F_2) \vee F_3 \equiv (F_1 \vee F_3) \wedge (F_2 \vee F_3)$ *(*distributivité *de la conjonction par rapport à la disjonction et de la disjonction par rapport à la conjonction)*

Démonstration D'après l'invariance de l'équivalence par substitution, il suffit de montrer ces équivalences lorsque F_1, F_2 et F_3 sont trois variables propositionnelles p_1, p_2 et p_3. La vérification est évidente sur les tables de vérités pour la première et la deuxième assertion.

Démonstration Pour la troisième, on peut construire les tables de vérités et on obtient par exemple pour la distributivité de la conjonction par rapport à la disjonction

p_1	p_2	p_3	$p_1 \vee p_2$	$(p_1 \vee p_2) \wedge p_3$	$p_1 \wedge p_3$	$p_2 \wedge p_3$	$(p_1 \wedge p_3) \vee (p_2 \wedge p_3)$
1	1	1	1	1	1	1	1
1	1	0	1	0	0	0	0
1	0	1	1	1	1	0	1
1	0	0	1	0	0	0	0
0	1	1	1	1	0	1	1
0	1	0	1	0	0	0	0
0	0	1	0	0	0	0	0
0	0	0	0	0	0	0	0

Remarque A partir de maintenant, nous jouerons sur l'associativité de la disjonction pour noter $F_1 \vee \ldots \vee F_n$ à la place de $(\ldots((F_1 \vee F_2) \vee F_3)\ldots) \vee F_{n-1}) \vee F_n)$. Nous procéderons de même pour la conjonction.

Proposition 2.5 *Soit F_1 et F_2 deux formules logiques. Alors*

- $\neg(\neg F_1) \equiv F_1$ *(loi du* tiers exclu*)*
- $\neg(F_1 \vee F_2) \equiv (\neg F_1) \wedge (\neg F_2)$ *et* $\neg(F_1 \wedge F_2) \equiv (\neg F_1) \vee (\neg F_2)$ *(lois de Morgan)*

Démonstration Comme précédemment, on utilise le théorème d'invariance par substitution pour se limiter au cas où les deux formules sont des variables propositionnelles. Il suffit alors d'établir les tables de vérité.

Proposition 2.6 *Deux tautologies sont équivalentes. Deux contradictions sont équivalentes. Soit F une formule logique.*

- *si T est une tautologie, alors $F \vee T \equiv T$ et $F \wedge T \equiv F$*
- *si C est une contradiction, alors $F \vee C \equiv F$ et $F \wedge C \equiv C$*
- *$F \vee (\neg F)$ est une tautologie, $F \wedge (\neg F)$ est une contradiction.*

Démonstration Même méthode en remplaçant F par une variable propositionnelle et la tautologie (resp. la contradiction) par une variable propositionnelle assujettie à ne prendre que la valeur 1 (resp. 0).

Proposition 2.7 *Soit F_1 et F_2 deux formules logiques. Alors*

- $F_1 \Rightarrow F_2 \equiv (\neg F_1) \vee F_2$ *(expression de l'implication en fonction de la négation et de la disjonction)*
- $F_1 \Rightarrow F_2 \equiv (\neg F_2) \Rightarrow (\neg F_1)$ *(loi de* contraposition*)*
- $F_1 \Leftrightarrow F_2 \equiv (F_1 \Rightarrow F_2) \wedge (F_2 \Rightarrow F_1)$

Démonstration Même méthode. La deuxième assertion peut également se montrer à l'aide de la première et de la loi du tiers exclu en écrivant

$$F_1 \Rightarrow F_2 \equiv (\neg F_1) \vee F_2 \equiv (\neg(\neg F_2)) \vee (\neg F_1) \equiv (\neg F_2) \Rightarrow (\neg F_1)$$

Proposition 2.8 *Soit F_1 et F_2 deux formules logiques. Alors $F_1 \otimes F_2 \equiv (F_1 \wedge (\neg F_2)) \vee ((\neg F_1) \wedge F_2)$*

Démonstration Idem

2.4 Formes normales des formules logiques

Définition 2.4 *On appelle disjonction toute formule logique F s'écrivant $F = F_1 \vee \ldots \vee F_n$ où chaque F_i est un* littéral, *c'est à dire soit une variable propositionnelle p_i, soit une négation de variable propositionnelle $\neg p_i$.*

Définition 2.5 *On appelle conjonction toute formule logique F s'écrivant $F = F_1 \wedge \ldots \wedge F_n$ où chaque F_i est soit une variable propositionnelle p_i soit une négation de variable propositionnelle $\neg p_i$.*

Remarque Si une variable p apparaît plusieurs fois dans une conjonction, on peut jouer sur la commutativité et les différentes formules $p \wedge p \equiv p$, $(\neg p) \wedge (\neg p) \equiv \neg p$, $(p \wedge (\neg p)) \equiv 0$ pour transformer la conjonction en une conjonction équivalente où ne figure qu'une seule fois la variable p ou en une contradiction 0.

De même, si la variable p apparaît plusieurs fois dans une disjonction, on peut jouer sur la commutativité et les formules $p \vee p \equiv p$, $(\neg p) \vee (\neg p) \equiv \neg p$, $(p \vee (\neg p)) \equiv 1$ pour transformer la conjonction en une conjonction équivalente où ne figure qu'une seule fois la variable p ou en une tautologie 1.

A équivalence près, on pourra toujours supposer que les variables propositionnelles apparaissent au plus une fois dans les conjonctions ou disjonctions.

Lemme 2.9 *Si F est une conjonction, alors $\neg F$ est équivalente à une disjonction. Si F est une disjonction, alors $\neg F$ est équivalente à une conjonction.*

Démonstration En effet, si F s'écrit$F = F_1 \vee \ldots \vee F_n$ où chaque F_i est soit une variable propositionnelle p_i, soit une négation de variable propositionnelle $\neg p_i$, d'après les lois de Morgan, $\neg F \equiv (\neg F_1) \wedge \ldots \wedge (\neg F_p)$ et chacune des $\neg F_i$ est soit une négation de variable propositionnelle (si F_i est une variable propositionnelle), soit équivalente à une variable propositionnelle (si F_i est une négation de variable propositionnelle). Donc $\neg F$ est équivalente à une conjonction. La démonstration est similaire pour une conjonction F.

Théorème 2.10 *Toute formule logique F est équivalente à une formule logique $F' = C_1 \vee \ldots \vee C_p$ où chacune des C_i est une conjonction ; autrement dit toute formule logique est équivalente à une disjonction de conjonctions. De même, toute formule logique est équivalente à une formule logique $F'' = D_1 \wedge \ldots \wedge D_q$ où chacune des D_j est une disjonction ; autrement dit toute formule logique est équivalente à une conjonction de disjonctions.*

Démonstration On montre les deux résultats à la fois par induction structurelle. Si F est réduite à une variable propositionnelle, alors F est à la fois une conjonction et une disjonction, donc le résultat est clair.
Si $F = \neg F_1$ où F_1 est équivalente à une disjonction de conjonctions, on a $F_1 \equiv C_1 \vee \ldots \vee C_p$ et alors, d'après la loi de Morgan, $F = \neg F_1 \equiv (\neg C_1) \wedge \ldots \wedge (\neg C_p) \equiv D_1 \wedge \ldots \wedge D_p$ si $\neg C_i \equiv D_i$ où D_i est une disjonction (d'après le lemme précédent). De même, si F_1 est équivalente à une conjonction de disjonctions, alors F est équivalente à une disjonction de conjonctions.
Si $F = F_1 \vee F_2$ où F_1 et F_2 sont équivalentes à des disjonctions de conjonctions, on a $F_1 \equiv C_1 \vee \ldots \vee C_p$ et $F_2 \equiv C'_1 \vee \ldots \vee C'_q$ d'où $F_1 \vee F_2 \equiv C_1 \vee \ldots \vee C_p \vee C'_1 \vee \ldots \vee C'_q$ qui est encore une disjonction de conjonctions. Si $F = F_1 \vee F_2$ où F_1 et F_2 sont équivalentes à des conjonctions de disjonctions, on a $F_1 \equiv D_1 \wedge \ldots \wedge D_p$ et $F_2 \equiv D'_1 \wedge \ldots \wedge D'_q$, d'où, par distributivité de la disjonction par rapport à la conjonction

$$F_1 \vee F_2 \equiv (D_1 \wedge \ldots \wedge D_p) \vee (D'_1 \wedge \ldots \wedge D'_q) = \bigwedge_{i,j} (D_i \vee D'_j)$$

où chacune des $D_i \vee D'_j$ est de façon évidente une disjonction.
On montre de même que si $F = F_1 \wedge F_2$ où F_1 et F_2 sont à la fois des disjonctions de conjonctions et des conjonctions de disjonctions, il en est de même de F. (On peut aussi utiliser la loi de Morgan qui dit que $F \equiv \neg((\neg F_1) \vee (\neg F_2))$ pour se ramener aux deux opérations précédentes.)
En ce qui concerne $F = F_1 \otimes F_2$, cela résulte alors de $F \equiv (F_1 \wedge (\neg F_2)) \vee ((\neg F_1) \wedge F_2)$ et des résultats déjà démontrés. Il en est de même pour $F = F_1 \Rightarrow F_2 \equiv (\neg F_1) \vee F_2$. L'équivalence $F_1 \Leftrightarrow F_2 \equiv (F_1 \Rightarrow F_2) \wedge (F_2 \Rightarrow F_1)$ garantit alors la véracité du résultat pour $F = F_1 \Leftrightarrow F_2$.

Exemple 2.2 Soit $F = (((\neg p) \vee q) \wedge r) \Rightarrow (p \otimes r)$. On a alors

$$\begin{aligned} F &\equiv (\neg(((\neg p) \vee q) \wedge r)) \vee (p \otimes r) \equiv (\neg((\neg p) \vee q) \vee (\neg r)) \vee ((p \wedge (\neg r)) \vee ((\neg p) \wedge r) \\ &\equiv (p \wedge (\neg q)) \vee (\neg r) \vee (p \wedge (\neg r)) \vee ((\neg p) \wedge r) \end{aligned}$$

qui est une disjonction de conjonctions.
Pour obtenir une conjonction de disjonctions, il suffit d'utiliser la distributivité de $\vee$ par rapport à $\wedge$ et d'utiliser les règles de simplification pour ne faire apparaître les variables qu'une seule fois dans chaque disjonction. On a alors

$$\begin{aligned} F &\equiv (p \wedge (\neg q)) \vee (\neg r) \vee (p \wedge (\neg r)) \vee ((\neg p) \wedge r) \\ &\equiv \Big((p \vee (\neg r)) \wedge ((\neg q) \vee (\neg r))\Big) \vee \Big((p \vee (\neg p) \wedge (p \vee r) \wedge ((\neg r) \vee (\neg p)) \wedge ((\neg r) \vee r)\Big) \\ &\equiv \Big((p \vee (\neg r)) \wedge ((\neg q) \vee (\neg r))\Big) \vee \Big(1 \wedge (p \vee r) \wedge ((\neg r) \vee (\neg p)) \wedge 1\Big) \\ &\equiv \Big((p \vee (\neg r)) \wedge ((\neg q) \vee (\neg r))\Big) \vee \Big((p \vee r) \wedge ((\neg r) \vee (\neg p))\Big) \\ &\equiv \Big(p \vee (\neg r) \vee p \vee r\Big) \wedge \Big(p \vee (\neg r) \vee (\neg r) \vee (\neg p)\Big) \\ &\quad \wedge \Big((\neg q) \vee (\neg r) \vee p \vee r\Big) \wedge \Big((\neg q) \vee (\neg r) \vee (\neg r) \vee (\neg p)\Big) \\ &\equiv \Big(p \vee 1\Big) \wedge \Big(1 \vee (\neg r)\Big) \wedge \Big(p \vee (\neg q) \vee 1\Big) \wedge \Big((\neg p) \vee (\neg q) \vee (\neg r)\Big) \\ &\equiv 1 \wedge 1 \wedge 1 \wedge (\neg p) \vee (\neg q) \vee (\neg r) \equiv (\neg p) \vee (\neg q) \vee (\neg r) \end{aligned}$$

qui est une conjonction d'une seule disjonction.

2.5 Utilisation de tables de vérité

Les méthodes que nous venons de voir pour transformer une formule logique en une formule équivalente qui est soit une conjonction de disjonctions, soit une disjonction de conjonctions sont en fait basées sur des règles formelles de simplification de formules logiques à équivalence près (commutativité, distributivité, etc.).

Nous allons maintenant voir d'autres méthodes qui utilisent la fonction booléenne associée à la formule logique.

Lemme 2.11 *Soit $p_1, \ldots, p_n$ des variables propositionnelles et $\sigma : \{p_1, \ldots, p_n\} \to \{0,1\}$. Posons $F_i = p_i$ si $\sigma(p_i) = 1$ et $F_i = (\neg p_i)$ si $\sigma(p_i) = 0$. Soit C_σ la conjonction $C_\sigma = F_1 \wedge \ldots \wedge F_n$.*

Alors pour toute application $\tau : \{p_1, \ldots, p_n\} \to \{0,1\}$, on a $\mathrm{Val}(C_\sigma, \tau) = \begin{cases} 1 & si\ \sigma = \tau \\ 0 & si\ \sigma \neq \tau \end{cases}$.

Démonstration En effet $\mathrm{Val}(F_\sigma, \tau)$ vaut 1 si et seulement si chacun des $\mathrm{Val}(F_i, \tau)$ vaut 1, c'est à dire si $\tau(p_i) = \begin{cases} 1 & \text{si } \sigma(p_i) = 1 \\ 0 & \text{si } \sigma(p_i) = 0 \end{cases}$ ce qui équivaut à dire que $\tau = \sigma$.

Théorème 2.12 *Soit F une formule logique, $p_1, \ldots, p_n$ les variables propositionnelles intervenant dans F et pour toute application $\sigma : \{p_1, \ldots, p_n\} \to \{0,1\}$, soit C_σ la conjonction définie ci dessus. Soit Σ l'ensemble des applications $\sigma : \{p_1, \ldots, p_n\} \to \{0,1\}$ telles que $\mathrm{Val}(F,\sigma) = 1$. Alors F est équivalente à la disjonction de conjonctions $F \equiv \bigvee_{\sigma\in\Sigma} C_\sigma$.*

Démonstration Posons $G = \bigvee_{\sigma\in\Sigma} C_\sigma$ et soit τ une application de $\{p_1, \ldots, p_n\}$ dans $\{0,1\}$. Alors $\mathrm{Val}(G,\tau) = 1$ si et seulement si au moins l'un des $\mathrm{Val}(C_\sigma,\tau)$ vaut 1 pour un $\sigma \in \Sigma$, autrement dit si et seulement si τ est l'un des $\sigma \in \Sigma$, c'est à dire si et seulement si $\tau \in \Sigma$. Donc $\mathrm{Val}(G,\tau) = 1$ si et seulement si $\mathrm{Val}(F,\tau) = 1$. Ceci montre bien que $\forall\tau,\ \mathrm{Val}(G,\tau) = \mathrm{Val}(F,\tau)$. Donc F et G sont équivalentes.

Définition 2.6 *La formule logique $\bigvee_{\sigma\in\Sigma} C_\sigma$ du théorème précédent s'appelle la forme normale disjonctive de la formule logique F. Les C_σ sont appelés des* minterms *(prononcer mine-termes).*

Remarque Toutes les conjonctions intervenant dans la forme normale disjonctive de F ont la particularité de contenir toutes les variables propositionnelles de F une et une seule fois, soit par elle mêmes, soit par leur négation.

On peut construire la forme normale disjonctive à partir de la table de vérité de la formule F. On commence par supprimer toutes les lignes pour lesquelles l'évaluation de F donne 0 (correspondant aux σ telles que $\mathrm{Val}(F, \sigma) \neq 1$). Pour chaque ligne restante on construit la conjonction obtenue en prenant toutes les variables propositionnelles, celles ayant pour valeur 1 dans la ligne étant prises directement, celles ayant pour valeur 0 étant niées (la conjonction obtenue n'est autre que C_σ). On prend alors la disjonction de toutes les conjonctions ainsi obtenues.

Exemple 2.3 Repartons de $F = (((\neg p) \vee q) \wedge r) \Leftrightarrow (p \otimes r)$. On a précédemment construit la table de vérité

p	q	r	$\neg p$	$(\neg p) \vee q$	$((\neg p) \vee q) \wedge r$	$p \otimes r$	$(((\neg p) \vee q) \wedge r) \iff (p \otimes r)$
1	1	1	0	1	1	0	0
1	1	0	0	1	0	1	0
1	0	1	0	0	0	0	1
1	0	0	0	0	0	1	0
0	1	1	1	1	1	1	1
0	1	0	1	1	0	0	1
0	0	1	1	1	1	1	1
0	0	0	1	1	0	0	1

Exemple 2.4 Si l'on ne garde que les lignes donnant la valeur 1 on obtient une table réduite

p	q	r	$(((\neg p) \vee q) \wedge r) \iff (p \otimes r)$	conjonction correspondante
1	0	1	1	$p \wedge (\neg q) \wedge r$
0	1	1	1	$(\neg p) \wedge q \wedge r$
0	1	0	1	$(\neg p) \wedge q \wedge (\neg r)$
0	0	1	1	$(\neg p) \wedge (\neg q) \wedge r$
0	0	0	1	$(\neg p) \wedge (\neg q) \wedge (\neg r)$

d'où $F \equiv \Big(p \wedge (\neg q) \wedge r\Big) \vee \Big((\neg p) \wedge q \wedge r\Big) \vee \Big((\neg p) \wedge q \wedge (\neg r)\Big) \vee \Big((\neg p) \wedge (\neg q) \wedge r\Big) \vee \Big((\neg p) \wedge (\neg q) \wedge (\neg r)\Big)$

Pour trouver maintenant une conjonction de disjonctions équivalente à une formule F, on peut appliquer la méthode précédente à la formule $\neg F$ puis appliquer les lois de Morgan, c'est à dire remplacer les conjonctions par des disjonctions, les disjonctions par des conjonctions, les p_i par des $\neg p_i$ et les $\neg p_i$ par des p_i. Sur la table de vérité, cela revient à ne garder que les lignes pour lesquelles la valeur de F vaut 0 ; pour chacune de ces lignes on construit la disjonction obtenue en prenant toutes les variables logiques, celles ayant pour valeur 0 étant prises directement, celles ayant pour valeur 1 étant niées. On prend alors la conjonction de toutes les disjonctions ainsi obtenues.

Exemple 2.5 Repartons de $F = (((\neg p) \vee q) \wedge r) \Leftrightarrow (p \otimes r)$. On a précédemment construit la table de vérité

p	q	r	$\neg p$	$(\neg p) \vee q$	$((\neg p) \vee q) \wedge r$	$p \otimes r$	$(((\neg p) \vee q) \wedge r) \iff (p \otimes r)$
1	1	1	0	1	1	0	0
1	1	0	0	1	0	1	0
1	0	1	0	0	0	0	1
1	0	0	0	0	0	1	0
0	1	1	1	1	1	1	1
0	1	0	1	1	0	0	1
0	0	1	1	1	1	1	1
0	0	0	1	1	0	0	1

Exemple 2.6 Si l'on ne garde que les lignes donnant la valeur 0 on obtient une table réduite

p	q	r	$(((\neg p) \vee q) \wedge r) \iff (p \otimes r)$	disjonction correspondante
1	1	1	0	$(\neg p) \vee (\neg q) \vee (\neg r)$
1	1	0	0	$(\neg p) \vee (\neg q) \vee r$
1	0	0	0	$(\neg p) \vee q \vee r$

si bien que $F \equiv \Big((\neg p) \vee (\neg q) \vee (\neg r)\Big) \wedge \Big((\neg p) \vee (\neg q) \vee r\Big) \wedge \Big((\neg p) \vee q \vee r\Big)$

Définition 2.7 *La formule logique* $\bigwedge_{\sigma \in \Sigma} D_\sigma$ *ainsi obtenue s'appelle la forme normale conjonctive de la formule logique* F*.*

Remarque Toutes les disjonctions intervenant dans la forme normale conjonctive de F ont la particularité de contenir toutes les variables propositionnelles de F une et une seule fois, soit par elle mêmes, soit par leur négation.

Bien entendu les deux méthodes précédentes peuvent s'appliquer à toute fonction booléenne, ce qui montre que toute fonction booléenne est la fonction associée à une conjonction de disjonctions et aussi la fonction associée à une disjonction de conjonctions.

Une discussion s'impose alors sur l'utilité de ce que nous venons de faire. La transformation d'une formule logique en une conjonction de disjonctions (ou une disjonction de conjonctions) équivalente est une opération fondamentale dans la programmation logique (dont un exemple est le langage Prolog qui sert en particulier dans de nombreux domaines de l'intelligence artificielle) ; elle est à la base des algorithmes de raisonnement. Deux méthodes sont possibles pour obtenir une telle formule équivalente. La première est celle du paragraphe précédent ; elle utilise certaines règles mécaniques (associativité, commutativité, distributivité, lois de Morgan, tiers exclu) ; elle est d'un emploi pénible à la main, mais relativement facile à implémenter avec un outil de calcul formel rompu à ce genre de manipulations. La deuxième, celle des tables de vérité, est relativement facile à effectuer à la main dans le cas d'un petit nombre de variables ; par contre on sait qu'elle est coûteuse en temps de calcul puisque l'établissement d'une table de vérité pour une formule logique à n variables demande un temps proportionnel à 2^n.

3 Circuits logiques élémentaires

3.1 Notion de circuit logique

Il s'agit ici, sans rentrer dans les détails techniques, de décrire l'implémentation physique possible de l'évaluation d'une formule logique.

Donnons nous une formule logique F dépendant des variables propositionnelles $p_1, \ldots, p_n$. Nous allons chercher à construire un circuit électronique $\mathcal{C}$ disposant de n entrées $P_1, \ldots, P_n$ et d'une sortie Q. Chacune de ces entrées P_i peut être portée soit à la tension 0 Volts, soit à la tension +5 Volts (les chiffres réels n'ont pas d'importance) suivant que la valeur attribuée à la variable p_i vaut 0 ou 1. A chaque application $\sigma = \{p_1, \ldots, p_n\} \to \{0, 1\}$ correspond donc une distribution de tensions sur les entrées $P_1, \ldots, P_n$ du circuit. On cherche à faire en sorte que la tension sur la sortie Q soit alors 0 Volts si $\mathrm{Val}(F, \sigma) = 0$ et de 5 Volts si $\mathrm{Val}(F, \sigma) = 1$.

Il est clair que si un circuit logique effectue l'évaluation d'une formule logique F, il effectue aussi l'évaluation de toute formule logique équivalente à F.

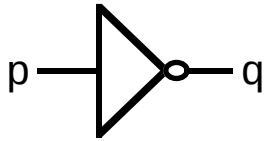
3.2 Portes logiques et circuits élémentaires

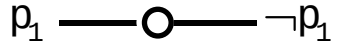
Nous supposerons par la suite que nous disposons d'un certain nombre de circuits élémentaires que nous pourrons assembler en reliant leurs entrées soit à une des entrée P_i soit à la sortie d'un autre circuit élémentaire. Ces circuits élémentaires sont appelés des portes logiques.

Un assemblage de portes logiques réalisant l'évaluation d'une formule logique particulière pourra ensuite être considérée comme un nouveau circuit élémentaire susceptible de réutilisation en tant que porte logique.

Nous allons décrire un système d'assemblage de portes logiques utilisant deux portes élémentaires[a] : une porte "non" et une porte "ou" ; ces deux portes logiques sont facilement réalisables par un assemblage d'un ou deux transistors.

La porte "non" est une porte logique à une entrée P et une sortie Q : Q est à la tension 5V si et seulement si P est à le tension 0V. La porte "non" réalise donc l'évaluation de la formule logique $q = \neg p$. Nous la symboliserons sous la forme



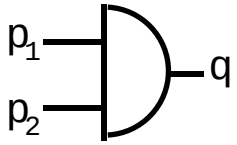
ou même sous forme simplifiée par un simple cercle 

,

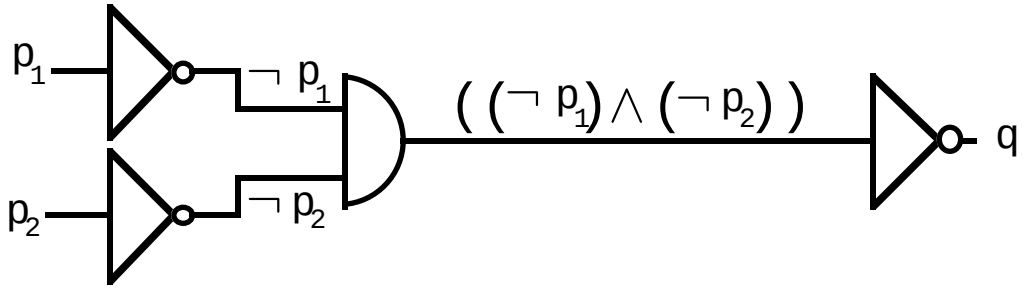
ce simple cercle pouvant être éventuellement accolé à un autre symbole en entrée ou en sortie.

[a] En fait on peut montrer qu'il est possible de construire tout circuit logique en utilisant une seule porte logique bien choisie

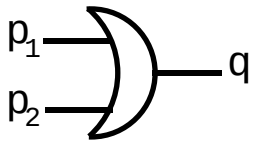
La porte "et" est une porte logique à deux entrées P_1 et P_2 et une sortie Q. La sortie Q est à le tension 5V si et seulement si chacune des deux entrées est à la tension 5V. La porte "et" réalise donc l'évaluation de la formule logique $q = p_1 \wedge p_2$. Nous la symboliserons sous la forme

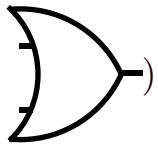


On peut alors réaliser un circuit logique qui effectue l'évaluation de la formule logique $q = p_1 \vee p_2$. Il suffit d'utiliser la loi de Morgan qui nous dit que $p_1 \vee p_2 \equiv \neg((\neg p_1) \wedge (\neg p_2))$ ce qui nous donne un circuit logique

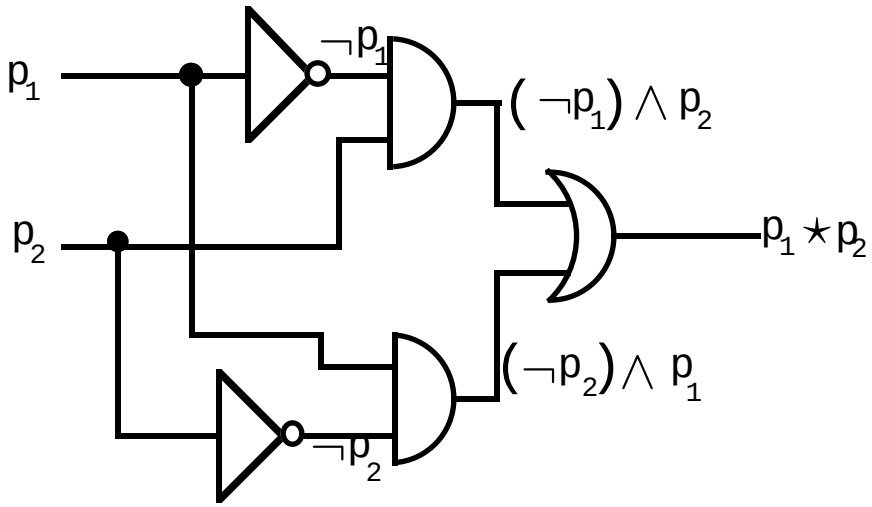


Etant donné l'utilité de ce circuit logique, nous en ferons une nouvelle porte logique, que nous appellerons porte "ou" et que nous symboliserons sous la forme



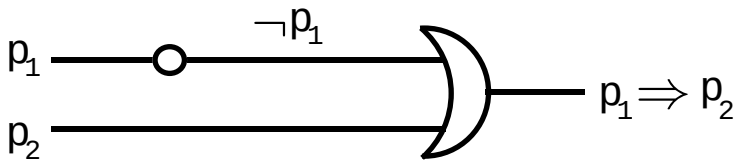
(ou encore avec le symbole )

On peut alors assembler ces trois portes logiques pour évaluer les autres opérateurs logiques élémentaires. En ce qui concerne l'opérateur xor que nous avons noté $\otimes$, on peut utiliser $p_1 \otimes p_2 \equiv ((\neg p_1) \wedge p_2) \vee (p_1 \wedge (\neg p_2))$ d'où le circuit logique

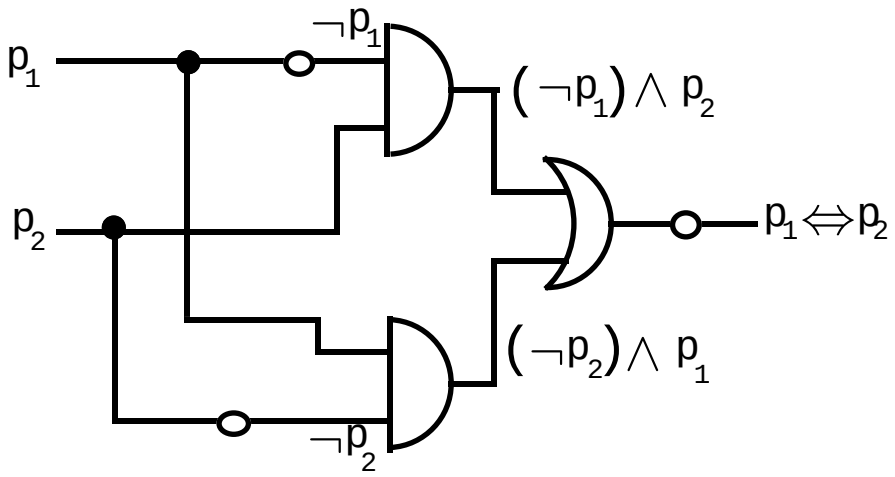


Remarque Par convention, dans un circuit logique, lorsque deux fils se croisent, ils sont considérés comme passant sur deux niveaux différents et donc être isolés l'un de l'autre. Lorsque l'on veut connecter deux fils, on doit expliciter cette connexion par un rond noir qui la symbolise. Dans la réalisation pratique de circuit logique, ces croisements de fils sur plusieurs niveaux sont difficilement réalisables et la conception de circuits intégrés comportant des centaines de milliers ou des millions de portes logiques interconnectées pose de redoutables problèmes de géométrie combinatoire dans lesquels la théorie des graphes joue un rôle essentiel.

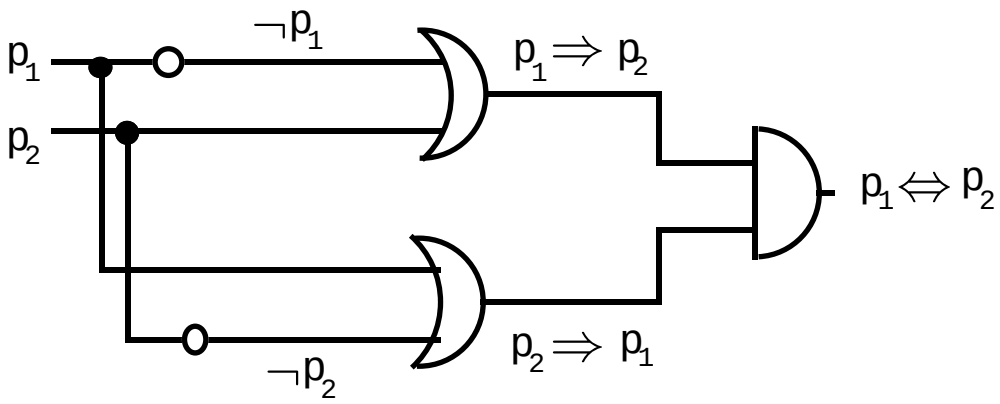
Pour ce qui est de l'opérateur d'implication, on peut utiliser $p_1 \Rightarrow p_2 \equiv p_2 \vee (\neg p_1)$ ce qui donne le circuit



Quant à l'opérateur d'équivalence, on peut remarquer que $p_1 \Leftrightarrow p_2 \equiv \neg(p_1 \otimes p_2)$ ce qui nous fournit le circuit



ou encore que $p_1 \Leftrightarrow p_2 \equiv (p_1 \Rightarrow p_2) \wedge (p_2 \Rightarrow p_1)$ ce qui nous donne le circuit

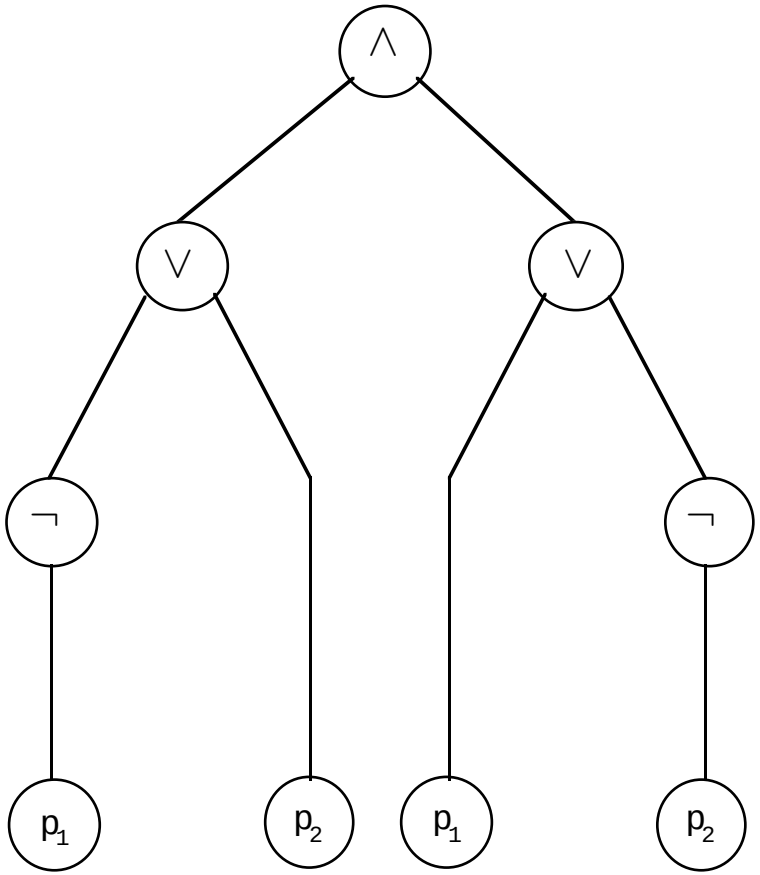


3.3 Circuits logiques et représentations arborescentes

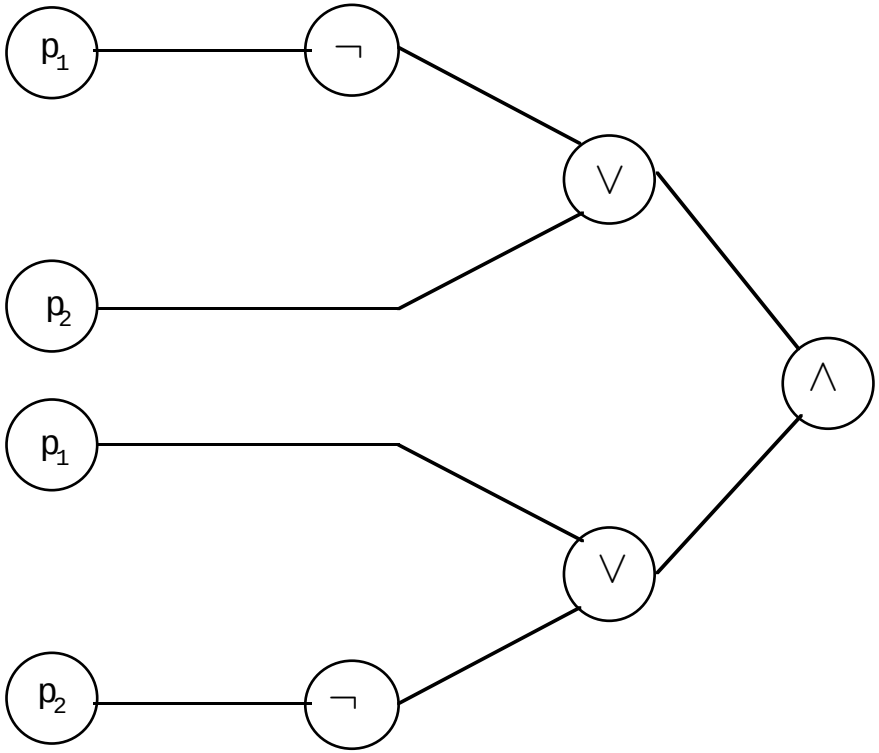
Reprenons le dernier circuit logique que nous avons construits, où nous avons écrit que

$$p_1 \Leftrightarrow p_2 \equiv (p_1 \Rightarrow p_2) \wedge (p_2 \Rightarrow p_1) \equiv (p_2 \vee (\neg p_1)) \wedge ((\neg p_2) \vee p_1)$$

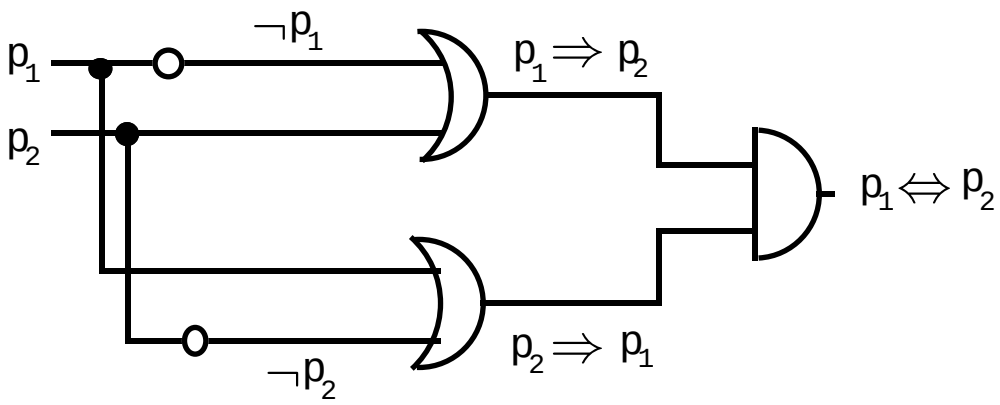
La représentation arborescente de cette dernière formule est



Effectuons une rotation de cette représentation arborescente. Nous obtenons alors un arbre "couché"



que nous pouvons comparer avec le circuit logique qui évalue la formule



La structure du circuit logique et la structure de l'arbre sont tout à fait identiques :

- chaque noeud étiqueté par $\wedge, \vee$ ou $\neg$ correspond à une porte logique "et", "ou" ou "non"
- les feuilles étiquetées par les variables logiques p_i correspondent aux entrées de même nom du circuit logique
- la racine de l'arbre correspond à la sortie du circuit

Ceci se généralise de manière évidente à toute formule logique ne faisant intervenir que les opérateurs $\wedge, \vee$ et $\neg$; à partir de la représentation arborescente de la formule logique, on peut construire un circuit logique évaluant la formule en remplaçant tous les noeuds de l'arbre par les portes logiques correspondantes, en reliant les feuilles de l'arbre aux entrées du circuit portant la même étiquette et en reliant la racine de l'arbre à la sortie du circuit.

Pour les formules logiques faisant intervenir les opérateurs $\otimes, \Rightarrow$ et $\Leftrightarrow$, on remplace les noeuds correspondant par les circuits logiques construits dans le paragraphe précédent, considérés comme de nouvelles portes logiques.

3.4 Additionneurs

Prenons deux nombres écrits en base 2 avec p chiffres, $m = x_p \dots x_0$ et $n = y_p \dots y_0$, chacun des x_i et y_i étant égal soit à 0 soit à 1. On peut alors effectuer l'addition de ces deux nombres suivant le même algorithme qu'en base 10. On additionne les chiffres de droite à gauche en base 2 en tenant compte des retenues éventuelles. Autrement dit on effectue l'algorithme suivant (où c_i désigne la retenue)

- affecter 0 à c_0
- pour i allant de 0 à p faire
- additionner x_i, y_i et c_i, en déduire le chiffre $z_i \in \{0, 1\}$ et une retenue $c_{i+1} \in \{0, 1\}$

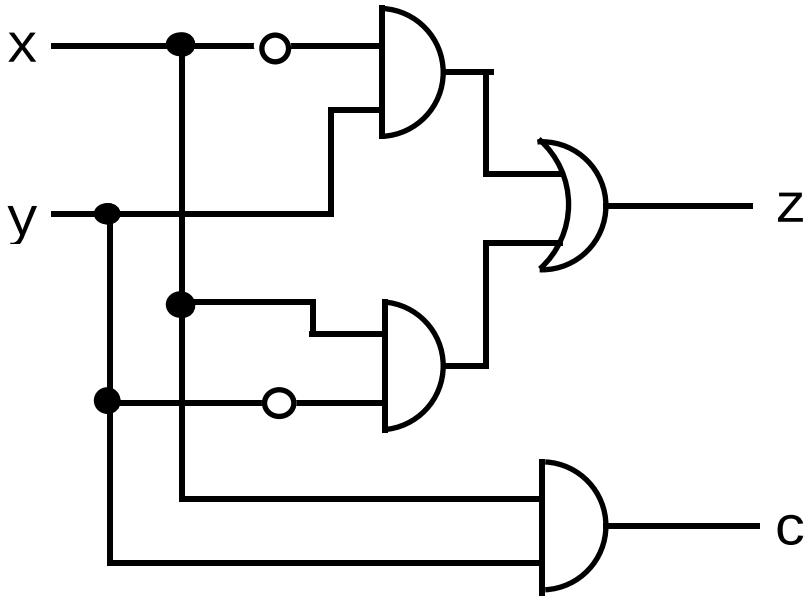
L'écriture en base 2 de $m + n$ est alors $c_{p+1} z_p \dots z_0$.

Remarque Dans la pratique des ordinateurs, $p + 1$ est fixe égal à 8, 16, 32 ou même 64 suivant les processeurs. En règle générale, le résultat retourné par une addition est non pas $c_{p+1} z_p \dots z_0$ mais simplement $z_p \dots z_0$. La dernière retenue c_{p+1} est donc négligée dans le résultat final. Elle sert cependant à positionner un *drapeau* qui signalera un débordement éventuel dans l'addition, autrement dit que le résultat ne tient pas vraiment sur $p + 1$ bits.

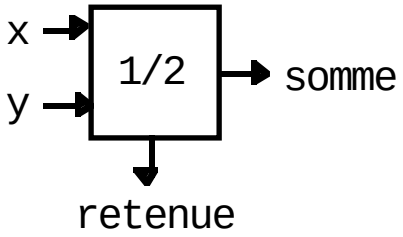
Circuit logique qui étant donné deux entrées x, y calcule z et une retenue c : demi-additionneur parce qu'il ne tient pas compte d'une retenue précédente qu'il faut ajouter ensuite au résultat obtenu : ceci nécessitera un deuxième demi-additionneur pour confectionner un additionneur complet. Pour cela nous allons simplement calculer une table de vérité à deux entrées et deux sorties.

x	y	c	z
1	1	1	0
1	0	0	1
0	1	0	1
0	0	0	0

Nous constatons immédiatement que $z = f_{\otimes}(x, y)$ et que $c = f_{\wedge}(x, y)$. Ceci nous permet de construire notre circuit demi-additionneur.



Nous symboliserons par la suite un tel circuit demi-additionneur sous la forme

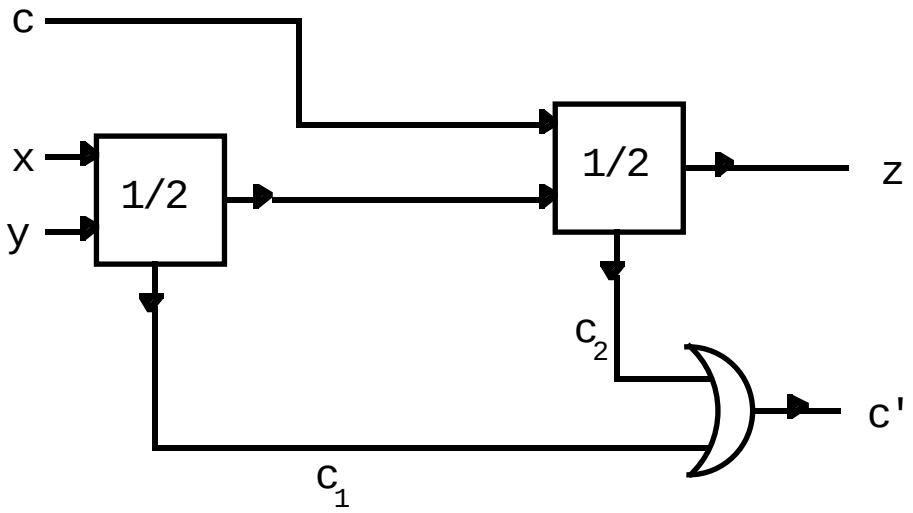


Construisons alors un circuit additionneur complet qui prend trois entrées x, y et c (la retenue précédente) et qui retourne un chiffre z et une nouvelle retenue c'.

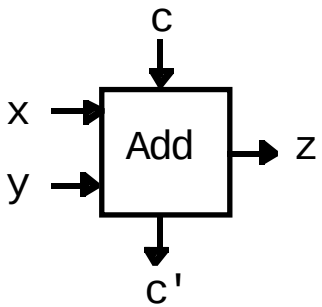
Pour cela on commence par additionner x et y obtenant un chiffre z_1 et une retenue c_1. Il faut ensuite que nous additionnions z_1 et c pour obtenir un chiffre qui n'est autre que z et une retenue c_2. Mais remarquons que si $c_1 = 1$, alors nécessairement $z_1 = 0$ (voir la table de vérité ci dessus) et donc nécessairement $c_2 = 0$, et alors $c' = c_1$. Si par contre $c_1 = 0$, alors $c' = c_2$. Autrement dit, on a toujours $c' = f_\vee(c_1, c_2)$. On peut aussi constater de visu le résultat sur la table de vérité à trois entrées

x	y	c	z_1	c_1	z	c_2	c'
1	1	1	0	1	1	0	1
1	1	0	0	1	0	0	1
1	0	1	1	0	0	1	1
1	0	0	1	0	1	0	0
0	1	1	1	0	0	1	1
0	1	0	1	0	1	0	0
0	0	1	0	0	1	0	0
0	0	0	0	0	0	0	0

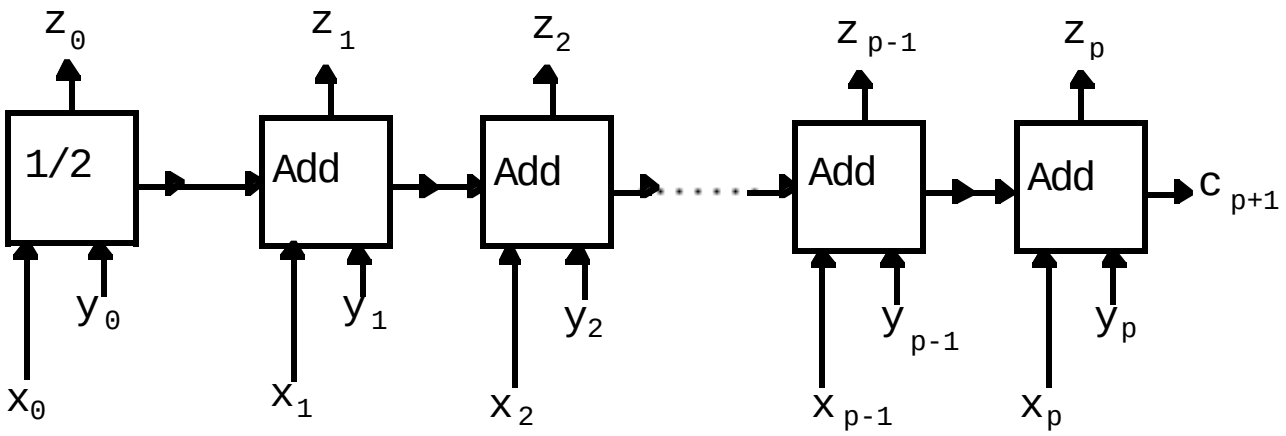
Ceci nous conduit à un circuit additionneur, complet de la forme



Nous symboliserons un tel circuit additionneur sous la forme



Par assemblage, on peut alors construire un circuit additionneur p bits de la manière suivante



Remarque En fait la réalisation d'un tel circuit pose de délicats problèmes de temporisation. En effet l'établissement des tensions dans les sorties n'est pas instantanée et il est essentiel qu'un additionneur ne prenne en compte la retenue provenant de l'additionneur de gauche qu'après qu'elle ait été clairement établie. Le cadencement est donc une étape importante de la conception des circuits intégrés, d'où le rôle des quartz qui jouent le rôle de chefs d'orchestre de tous les circuits logiques.