LOGIQUE MATHÉMATIQUE : INTRODUCTION

Arnaud Durand - Paul Rozière
Paris 7 – 31HU03MI

11 octobre 2016

(*version de travail, ne pas hésiter à indiquer coquilles et erreurs, le polycopié est encore incomplet certaines parties seront développées*)

Chapitre 1

Calcul propositionnel

1.1 Syntaxe et sémantique de la logique propositionnelle

Il y a deux aspects principaux dans la description d'un langage : sa *syntaxe* et sa *sémantique*. Intuitivement, la syntaxe concerne la structure du langage et la forme de ses énoncés; la sémantique s'intéresse au "sens" (ici mathématique) à donner au langage.

1.1.1 Syntaxe

Soit un ensemble de constantes propositionnelles $\mathscr{P}$ (on parlera souvent de variables propositionnelles dans la suite). L'ensemble des formules du calcul propositionnel sur $\mathscr{P}$ — nous dirons formules propositionnelles sur $\mathscr{P}$ — que l'on notera $\mathfrak{F}_{\mathscr{P}}$, ou simplement $\mathfrak{F}$ s'il n'y a pas d'ambiguïté, est l'ensemble engendré *inductivement* et *librement* par les règles de construction suivantes :

atomes si $p \in \mathscr{P}$, p est une formule propositionnelle.

absurde $\bot$ est une formule propositionnelle.

négation Si F est une formule propositionnelle $\neg F$ est une formule propositionnelle.

implication Si F et G sont des formules propositionnelles $(F \to G)$ est une formule propositionnelle.

conjonction Si F et G sont des formules propositionnelles $(F \wedge G)$ est une formule propositionnelle.

disjonction Si F et G sont des formules propositionnelles $(F \vee G)$ est une formule propositionnelle.

Par *engendré inductivement*, on entend qu'une formule ne peut être obtenue que par l'une de ces règles de construction, soit par la règle initiale, soit par l'une des règles suivantes à partir de formules déjà construites. Cela revient à définir $\mathfrak{F}$ comme l'union $\bigcup_{n\in\mathbb{N}} \mathfrak{F}_n$ où :

— $\mathfrak{F}_0 = \mathscr{P}$ et,

— pour tout $n \in \mathbb{N}$, $\mathfrak{F}_{n+1} = \mathfrak{F}_n \cup \{\neg F : F \in \mathfrak{F}_n\} \cup \{(F \odot G) : F, G \in \mathfrak{F}_n, \odot \in \{\wedge, \vee \to\}\}$.

La *hauteur* d'une formule F est le plus petit $n \in \mathbb{N}$ tel que $F \in \mathfrak{F}_n$.

Pour démontrer des propriétés de l'ensemble des formules propositionnelles, on peut raisonner par récurrence sur la hauteur de la formule, mais on utilise plutôt le principe équivalent de *raisonnement par induction sur la structure* des formules.

Raisonnement par induction. Soit une propriété à vérifier sur l'ensemble de toutes les formules propositionnelles $\mathfrak{F}_{\mathscr{P}}$. Alors

— si la propriété est vérifiée pour toutes les variables propositionnelles et l'absurde,

— si, quand elle est vérifiée pour F elle est encore vérifiée pour $\neg F$

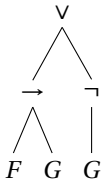
— si, quand elle est vérifiée pour F et G elle est encore vérifiée pour $(F \to G)$, $(F \wedge G)$ et $(F \vee G)$

alors elle est vérifiée pour toutes les formules propositionnelle.

Ce principe de raisonnement est une conséquence directe de la définition inductive de l'ensemble des formules, et du fait qu'une formule ne peut être obtenue que par l'une des règles indiquée. On verra plus loin des utilisations du raisonnement par induction.

On peut représenter de façon arborescente la suite des constructions qui mène à une formule, c'est l'*arbre de décomposition de la formule*[1]. Par exemple la formule $((F \to G) \vee \neg G)$ a pour arbre de décomposition :

1. On définit cette notion formellement plus loin en s'appuyant sur celle d'arbre orienté étiqueté



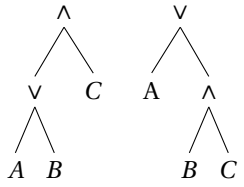
Par *engendré librement* on entend que l'on ne peut pas arriver à la même formule par deux constructions différentes, ou encore qu'une formule possède un seul arbre de décomposition. On exprime ceci plus précisément par la *propriété de lecture unique*.

Propriété de lecture unique. Pour toute formule F de $\mathfrak{F}_{\mathscr{P}}$, exactement un des cas suivants est réalisé :

1. $F \in \mathscr{P}$
2. il existe unique $G \in \mathfrak{F}_{\mathscr{P}}$ telle que $F = \neg G$
3. il existe un unique $\odot \in \{\wedge, \vee \rightarrow\}$ et deux uniques $G, H \in \mathfrak{F}_{\mathscr{P}}$ tels que $F = (G \odot H)$.

Une réalisation concrète. Une façon de réaliser concrètement l'ensemble des formules propositionnelles est de voir les formules propositionnelles comme des mots sur l'alphabet $\Sigma = \mathscr{P} \cup \{\wedge, \vee, \rightarrow,), (, \neg, \bot\}$, l'ensemble des mots sur Σ étant noté Σ^*. L'ensemble $\mathfrak{F}$ peut alors être réalisé comme le plus petit sous-ensemble de Σ^* contenant $\mathscr{P}$ et clos, pour tout mot F et G par les opérations $F \mapsto \neg F$, $(F,G) \mapsto (F \wedge G)$, $(F,G) \mapsto (F \vee G)$, $(F,G) \mapsto (F \rightarrow G)$. Là encore c'est une autre façon de dire que l'ensemble est défini inductivement par les règles ci-dessus. Pour que la réalisation soit correcte, il faut alors démontrer un *théorème de lecture unique* — la propriété de lecture unique ci-dessus — pour s'assurer que ce sous-ensemble de Σ^* est bien librement engendré.

Le parenthésage indiqué dans les opérations de clôture est utile pour montrer cette propriété, par exemple pour distinguer les formules $((A \vee B) \wedge C)$ et $(A \vee (B \wedge C))$ d'arbres de décomposition



On se rend compte cependant que les parenthèses extérieures peuvent toujours être omises, chose que l'on se permettra désormais. On pourrait également obtenir la lecture unique par des règles de priorité, comme pour les expressions arithmétiques où le signe $\times$ est prioritaire sur le signe $+$. On s'en abstiendra dans la suite (une convention courante est que le $\wedge$ est prioritaire sur les autres connecteurs). Une autre convention est de décider arbitrairement de décomposer une expression non parenthésée d'un certaine façon. Dans la suite on décide par exemple de lire $A \vee B \vee C$ comme $(A \vee B) \vee C$, donc $A \vee B \vee C \vee D$ comme $((A \vee B) \vee C) \vee D$, etc. De même pour le $\wedge$. On verra que cela est de peu d'importance car d'un point de vue sémantique ces connecteurs sont associatifs. Une convention parfois utilisée est de lire $A \rightarrow B \rightarrow C$ comme $A \rightarrow (B \rightarrow C)$, donc $A \rightarrow B \rightarrow C \rightarrow D$ comme $A \rightarrow (B \rightarrow (C \rightarrow D)$, etc. On verra pourtant que le connecteur n'est pas associatif et la convention est importante.

Toutes ces choses sont indispensables à prendre en compte si on s'intéresse à l'implémentation du calcul propositionnel sur machine. On doit définir une *grammaire* pour le calcul propositionnel, qui est une grammaire *context free* (indépendante du contexte). Il existe des outils spécialisés pour l'*analyse syntaxique* de ce genre d'expressions, soit reconnaître si un mot est bien une expression du langage (ici une formule propositionnelle), et dans ce cas de produire une représentation concrète de son arbre de décomposition.

Sous-formule, hauteur et arbre de décomposition. La propriété de lecture unique permet de montrer que les définitions par induction, comme celle des sous-formules ci-dessous, sont correctes. Ce sont l'analogue des définitions par récurrence pour les entiers naturels.

Définition 1.1.1 (sous-formule) *On définit inductivement l'ensemble* $\mathsf{sf}(F)$ *des sous-formules de* $F \in \mathfrak{F}$ *par :*

— *Si* $F \in \mathscr{P}$, *alors* $\mathsf{sf}(F) = \{F\}$

— *Si* F *s'écrit* $\neg G$, *alors* $\mathsf{sf}(F) = \mathsf{sf}(G) \cup \{F\}$;

— *Si F s'écrit $G \odot H$ (avec $\odot \in \{\wedge, \vee \rightarrow\}$), alors* $\mathsf{sf}(F) = \mathsf{sf}(G) \cup \mathsf{sf}(H) \cup \{F\}$.

La propriété de lecture unique est clairement nécessaire pour que ces cas soient disjoints et que la définition soit donc non ambiguë.

On montre facilement que (du fait du théorème de lecture unique) la hauteur h d'une formule (voir ci-dessus) peut se définir inductivement :

— si $F \in \mathscr{P}$, alors $\mathsf{h}(F) = 0$;

— si F s'écrit $\neg G$, alors $\mathsf{h}(F) = 1 + \mathsf{h}(G)$;

— si F s'écrit $G \odot H$ (avec $\odot \in \{\wedge, \vee \rightarrow\}$), alors $\mathsf{h}(F) = 1 + \sup(\mathsf{h}(F), \mathsf{h}(G))$.

Du fait de la propriété de lecture unique, une formule F possède un unique arbre de décomposition $\mathsf{arb}(F)$, que l'on peut définir de la façon suivante.

Définition 1.1.2 (arbre de décomposition) *On définit inductivement l'arbre de décomposition* $\mathsf{arb}(F)$ *associé à F par :*

1. *si $F \in \mathscr{P}$, alors* $\mathsf{arb}(F)$ *se réduit à un point étiqueté F ;*
2. *si $F = \neg G$, alors* $\mathsf{arb}(F)$ *est l'arbre de racine étiquetée par $\neg$ ayant pour unique sous-arbre* $\mathsf{arb}(G)$ *;*
3. *$F = (G \odot H)$, alors* $\mathsf{arb}(F)$ *est l'arbre de racine étiquetée par $\odot$, ayant pour sous-arbre gauche* $\mathsf{arb}(G)$ *et pour sous-arbre droit* $\mathsf{arb}(H)$ *;*

Noter la disparition du parenthésage qui n'a plus lieu d'être dans un arbre orienté. On identifiera souvent F avec son arbre dans la suite. Noter que chaque sous-arbre de $\mathsf{arb}(F)$ est l'arbre de décomposition d'une sous-formule de F et réciproquement chaque sous-formule de F admet pour arbre de décomposition un sous-arbre de $\mathsf{arb}(F)$.

Dans cette section, on ne s'est préoccupé que de syntaxe c'est à dire de la forme des énoncés du calcul propositionnel. Si l'on veut formaliser un problème dans ce langage, on prendra pour $\mathscr{P}$ un ensemble qui a une certaine structure mais on n'a pas besoin de connaître cette structure pour étudier la sémantique.

Notations. On utilisera également des notations supplémentaires, qui ne font pas partie directement de la syntaxe :

— $\top$ est une abréviation pour $(\bot \rightarrow \bot)$ (le "vrai") ;

— $(F \leftrightarrow G)$ est une abréviation pour $(F \rightarrow G) \wedge (G \rightarrow F)$ (l'équivalence).

On aurait tout aussi bien pu les ajouter à la syntaxe.

On introduit quelques notations qui sont des abréviations plus élaborées (les entiers interviennent, on changerait complètement la syntaxe en les y introduisant directement).

Si $A_1, \ldots, A_n$ sont des formules de $\mathfrak{F}$, $I = \{i_1, ..., i_k\} \subseteq \{1, ..., n\}$, alors on note

$$\bigwedge_{i=1}^{n} A_i = A_1 \wedge A_2 \wedge \cdots \wedge A_n \; ; \quad \bigwedge_{j \in I} A_j = A_{i_1} \wedge A_{i_2} \wedge \cdots \wedge A_{i_k} \; ;$$

de même

$$\bigvee_{i=1}^{n} A_i = A_1 \vee A_2 \vee \cdots \vee A_n \; ; \quad \bigvee_{j \in I} A_j = A_{i_1} \vee A_{i_2} \vee \cdots \vee A_{i_k} \; .$$

Une convention (assez arbitraire) d'association pour les connecteurs $\wedge$, $\vee$ a été a été vue au dessus. Elle permettrait une définition par récurrence de ces notations.

1.1.2 Sémantique : valuations, tables de vérité, ensemble de modèles

La seule chose qui nous intéresse est la valeur de vérité de l'interprétation des constantes propositionnnelles. Une constante propositionnelle est soit vraie, soit fausse. On appelle donc *valuation* une fonction de $\mathscr{P}$ dans $\{0,1\}$ (0 pour "Faux", 1 pour "Vrai").

On peut formuler la définition de l'interprétation d'une formule propositionnelle à l'aide des valuations. Étant donnée une valuation v de $\mathscr{P}$ dans $\{0,1\}$, on définit l'interprétation induite sur les formules du calcul propositionnel, que l'on note $\overline{v}$ et que l'on notera ensuite (abusivement) v, par induction sur la définition des formules propositionnelles :

atome $\overline{v}(p) = v(p)$.

absurde $\overline{v}(\bot) = 0$.

négation $\overline{v}(\neg G) = 1$ ssi $\overline{v}(G) = 0$.

conjonction $\overline{v}(G \wedge H) = 1$ ssi $\overline{v}(G) = 1$ et $\overline{v}(H) = 1$.

disjonction $\overline{v}(G \vee H) = 0$ ssi $\overline{v}(G) = 1$ et $\overline{v}(H) = 1$.

implication $\overline{v}(G \to H) = 0$ ssi $\overline{v}(G) = 1$ et $\overline{v}(H) = 0$.

De façon équivalente, on peut aussi définir $\overline{v}$ en utilisant les opérations usuelles "+" et "·" de $\mathbb{Z}/2\mathbb{Z}$:

négation $\overline{v}(\neg G) = 1 + \overline{v}(G)$.

conjonction $\overline{v}(G \wedge H) = \overline{v}(G) \cdot \overline{v}(H)$.

disjonction $\overline{v}(G \vee H) = \overline{v}(G) + \overline{v}(H) + \overline{v}(G) \cdot \overline{v}(H)$.

implication $\overline{v}(G \to H) = 1 + \overline{v}(G) + \overline{v}(G) \cdot \overline{v}(H)$.

On voit que pour chaque connecteur n-aire la sémantique est définie par une fonction de $\{0,1\}^n$ dans $\{0,1\}$. On a déjà décrit ces fonctions, de deux façons différentes. On peut également les décrire par des tableaux de 2^n lignes :

G	$\neg G$
0	1
1	0

G	H	$G \wedge H$	$G \vee H$	$G \to H$
0	0	0	0	1
0	1	0	1	1
1	0	0	1	0
1	1	1	1	1

Il faut se persuader que pour chacun de ces connecteurs, qui sont empruntés à la langue courante, la sémantique dérive bien de la logique intuitive (non spécifiquement mathématique), mais simplifiée drastiquement par le fait que seul deux valeurs de vérité sont possibles (le tiers-exclu).

Il est clair que l'interprétation d'une formule F pour une valuation v ne dépend que des valeurs de v pour les constantes propositionnelles qui apparaissent effectivement dans F et qui sont donc en nombre fini.

Étant donnée une formule qui n'utilise que les constantes propositionnelles $p_1, \ldots, p_n$, la *table de vérité* de la formule F pour $p_1, \ldots, p_n$ est la fonction de l'ensemble des valuations sur $p_1, \ldots, p_n$, soit $\{0,1\}^{\{p_1,\ldots,p_n\}}$, dans $\{0,1\}$, qui à chaque valuation v associe $\overline{v}(F)$. L'ensemble des tables de vérités sur les variables propositionnelles $p_1, \ldots, p_n$ est donc $\{0,1\}^{\{0,1\}^{\{p_1,\ldots,p_n\}}}$.

On peut représenter unetable de vérité comme ci-dessus par un tableau de 2^n lignes. Par exemple voici un calcul de la table de vérité de la formule $p \leftrightarrow q$ définie par $(p \to q) \wedge (q \to p)$ (pour chacune des 4 valuations v on utilise bien la définition du prolongement $\overline{v}$) :

p	q	$p \to q$	$q \to p$	$p \leftrightarrow q$
0	0	1	1	1
0	1	1	0	0
1	0	0	1	0
1	1	1	1	1

La table de vérité d'une formule décrit entièrement la sémantique d'une formule propositionnelle, puisqu'elle décrit toutes les interprétations possibles.

Définition 1.1.3 (satisfaisabilité, validité, équivalence)

— *Une formule propositionnelle F est* satisfaisable *s'il existe au moins une valuation v telle que $\overline{v}(F) = 1$. Par abus de langage, on dira qu'une telle valuation $\overline{v}$ est un modèle de F.*

— *Une formule propositionnelle F est valide (i.e. est une* tautologie*) si pour toute valuation v on a $\overline{v}(F) = 1$.*

— *Deux formules propositionnelles F et G sont équivalentes, on note $F \equiv G$, si elles sont satisfaites par les mêmes valuations :*

$$F \equiv G \text{ ssi pour toute valuation } v,\ \overline{v}(F) = \overline{v}(G).$$

On vérifie que :

$$F \equiv G \text{ ssi } F \leftrightarrow G \text{ est une tautologie,}$$

en utilisant la table de vérité de $\leftrightarrow$. On remarque également (c'est juste une reformulation de ce qui précède) que deux formules sont équivalentes si et seulement si elles ont les mêmes tables de vérité. On a aussi facilement le résultat suivant.

Proposition 1.1.4 ***(satisfaisabilité et validité)*** *Pour toute formule $F \in \mathfrak{F}$: F est satisfaisable si et seulement si $\neg F$ n'est pas valide.*

1.2 Exemples de formalisation en calcul propositionnel.

Il n'est pas très confortable de devoir formaliser en calcul propositionnel. Quand cela est possible on y gagne un langage beaucoup plus simple. Le langage s'avère néanmoins pratique pour formaliser un certain nombre de problèmes combinatoires quand l'espace des solutions est fini et de taille raisonnable. De plus on peut vérifier mécaniquement la satisfaisabilité d'une formule propositionnelle (donc la faisabilité d'un ensemble de contraintes qu'elle exprime). On verra ce qu'il en est plus tard de l'existence de méthodes qui soient efficaces (en théorie ou en pratique) pour tester la satisfaisabilité.

1.2.1 Contraintes de compatibilité ou d'exclusion

On considère le problème suivant. On dispose d'un ensemble de n produits chimiques que l'on doit ranger dans k conteneurs ($k \leqslant n$). On sait aussi que le stockage dans un même conteneur de certaines combinaisons de produits n'est pas possible (pour des raisons de sécurité). Ces contraintes sont données sous la forme d'un ensemble $\mathscr{L}$ de sous-ensembles de $\{1,...,n\}$ tel que tout $I = \{i_1,...,i_h\} \in \mathscr{L}$ s'interprète par : tous les produits $i_1,...,i_h$ de la liste I ne peuvent être ensemble dans le même conteneur.

On exprime petit à petit l'ensemble des contraintes du problème. Pour cela on va utiliser des variables propositionnelles $p(i,j)$, $i \leqslant n$, $j \leqslant k$ dont la signification intuitive est : *le produit i se trouve dans le conteneur j*.

— Chaque produit doit être rangé dans un conteneur (et un seul)

$$F = (\bigwedge_{i\leqslant n} \bigvee_{j\leqslant k} p(i,j)) \wedge (\bigwedge_{i\leqslant n} \bigwedge_{\substack{j,j'\leqslant k \\ j\neq j'}} \neg(p(i,j) \wedge p(i,j')))$$

— Le rangement doit respecter la liste d'incompatibilité :

$$G = \bigwedge_{I\in\mathscr{L}} \bigwedge_{j\leqslant k} \neg(\bigwedge_{i\in I} p(i,j)).$$

Il est clair que le problème de départ admet une solution (de rangement) si la formule $F \wedge G$ est satisfaisable [2]. Une valuation v telle que $v(F) = 1$ décrira une solution possible et l'ensemble des valuations satisfaisantes, l'ensemble des solutions possibles

1.2.2 Le Sudoku

Le jeu du Sudoku consiste à compléter une grille 9×9 partiellement remplie par des chiffres de 1 à 9 avec les contraintes qui suivent.

1. Chaque case contient un et un seul chiffre
2. Les chiffres de 1 à 9 apparaissent sur chacune des lignes. Autrement dit, aucune ligne ne contient deux fois le même chiffre.
3. Les chiffres de 1 à 9 apparaissent sur chacune des colonnes. Autrement dit, aucune colonne ne contient deux fois le même chiffre.
4. Les chiffres de 1 à 9 apparaissent dans chacun des sous-carrés 3×3 qui subdivisent la grille. Autrement dit, aucun de ces carrés ne contient deux fois le même chiffre.

On va modéliser les contraintes de ce jeu en calcul propositionnel. Du fait de la finitude des contraintes, celui ci s'y prête assez bien. Soit $p(i,j,k)$, avec $i,j,k \in \{1,...,9\}$, les variables propositionnelles désignant le fait que la case à l'intersection de la ligne i et de la colonne j contient k.

1. $\bigwedge_{1\leqslant i\leqslant 9} \bigwedge_{1\leqslant j\leqslant 9} \big(p(i,j,1) \vee p(i,j,2) \vee ... \vee p(i,j,k)) \wedge \bigwedge_{\substack{k,k'\leqslant 9 \\ k\neq k'}} (\neg p(i,j,k) \vee \neg p(i,j,k'))\big)$
2. $\bigwedge_{1\leqslant i\leqslant 9} \bigwedge_{1\leqslant j<j'\leqslant 9} \bigwedge_{1\leqslant k\leqslant 9} \neg p(i,j,k) \vee \neg p(i,j',k)$

2. Le prouver formellement

3. $\bigwedge_{1\leqslant j\leqslant 9}\bigwedge_{1\leqslant i<i'\leqslant 9}\bigwedge_{1\leqslant k\leqslant 9}\neg p(i,j,k)\vee\neg p(i',j,k)$

4. $\bigwedge_{i\in\{1,4,7\}}\bigwedge_{j\in\{1,4,7\}}\bigwedge_{(i',j')\neq(i'',j'')\in I}\bigwedge_{1\leqslant k\leqslant 9}\neg p(i+i',j+j',k)\vee\neg p(i+i'',j+j'',k)$
où $I=\{(i,j):i,j\in\{0,1,2\}\}$

Bien entendu, le jeu n'a de sens que si la grille est partiellement remplie au départ. Il faut donc ajouter un certain nombre de contraintes (que l'on dira *unitaire*) de la forme $p(i,j,k)$ pour indiquer que le nombre k se trouve, au départ, sur la case d'indice (i,j).

1.3 Retour sur la sémantique : ensemble de modèles et fonctions Booléennes

On reprend l'idée d'associer à une formule propositionnelle F l'ensemble de ses modèles c'est à dire l'ensemble des valuations v telles que $\overline{v}(F)=1$. Soit $\mathscr{P}$ un ensemble de variables propositionnelles, on définit la fonction

$$\mathsf{mod}:\mathfrak{F}\longrightarrow\{0,1\}^{\{0,1\}^{\mathscr{P}}}$$

de la façon suivante :
— Si $F\in\mathscr{P}$, $\mathsf{mod}(F)=\{v:\overline{v}(F)=1\}$
— Si $F=\neg G$, $\mathsf{mod}(F)=\{0,1\}^{\mathscr{P}}\setminus\mathsf{mod}(G)$.
— Si $F=G\vee H$, $\mathsf{mod}(F)=\mathsf{mod}(G)\cup\mathsf{mod}(H)$.
— Si $F=G\wedge H$, $\mathsf{mod}(F)=\mathsf{mod}(G)\cap\mathsf{mod}(H)$.
— Si $F=G\rightarrow H$, $\mathsf{mod}(F)=(\{0,1\}^{\mathscr{P}}\setminus\mathsf{mod}(G))\cup\mathsf{mod}(H)$.
La fonction mod associe à tout $F\in\mathfrak{F}$ l'ensemble de ses modèles. On peut montrer facilement par induction le résultat suivant.

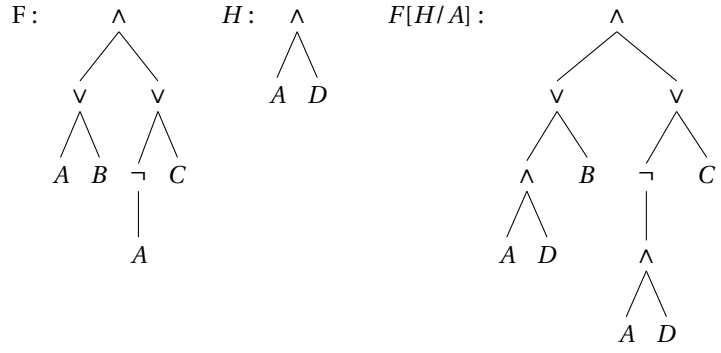
Proposition 1.3.1 *Pour toute formule* $F\in\mathfrak{F}$, $\mathsf{mod}(F)=\{v:\overline{v}(F)=1\}$.

Une *fonction Booléenne* en n variables est une fonction de $\mathbb{B}^n=\{0,1\}^n$ dans $\mathbb{B}=\{0,1\}$ [3]. Les fonctions Booléennes jouent un rôle important dans de nombreux domaines de l'informatique (aide à la décision, recherche opérationnelle, contraintes, codes, cryptographie, ...) et de l'électronique. L'ensemble des valuations v de $\mathscr{P}=\{x_1,...,x_n\}$ dans $\{0,1\}$ est en bijection avec $\mathbb{B}^n$ et donc, pour tout $F\in\mathfrak{F}$, mod$F$ est une fonction Booléenne que l'on dira *représentée* par F.

1.4 Quelques équivalences logiques et tautologies usuelles.

La substitution simultanée de formules propositionnelles $H_1,\ldots,H_n$ aux variables propositionnelles $p_1,\ldots,p_n$ d'une formule propositionnelle F s'écrit $F[H_1/p_1,\ldots,H_n/p_n]$ (F dans laquelle on remplace simultanément [4] la variable p_1 par H_1, p_2 par H_2 etc.) se définit par induction sur F sans difficultés.

L'exemple suivant montre une telle subsitution.



3. Dans la suite, on utilisera souvent la structure d'algèbre de Boole de $(\mathbb{B}^n,\wedge,\vee,\mathbf{0},\mathbf{1},(.)')$ (isomorphe à l'algèbre de Boole des parties d'un ensemble à n éléments)

4. On insiste sur *simultanément* car rien n'empêche les formules H_i, $i\leqslant n$, de contenir certaines des variables $p_1,...,p_n$ et on veut éviter les imbrications dans les substitutions

La propriété de substitution énoncée ci-dessous permet de voir les tautologies comme des schémas de formules propositionnelles valides. Par exemple On sait que $p \to p$ est une tautologie, on en déduit immédiatement que $(p \to q) \to (p \to q)$, est une tautologie.

Proposition 1.4.1 (substitution) *Soit F et G des formules propositionnelles où n'apparaissent que les constantes propositionnelles $p_1, \ldots, p_n$. Soit $H_1, \ldots, H_n$ des formules propositionnelles sur un ensemble $\mathscr{P}$ de variables propositionnelles, alors*

— *si F est une tautologie, la formule*

$$F[H_1/p_1, \ldots, H_n/p_n] \text{ est une tautologie.}$$

— *si $F \equiv G$,*

$$F[H_1/p_1, \ldots, H_n/p_n] \equiv G[H_1/p_1, \ldots, H_n/p_n]$$

Noter que les réciproques des résultats ci-dessus ne sont pas vraies. Si $H_1 = \top$, et $F = p_1$ alors $F[H_1/p_1] = H_1 = \top$ qui est une tautologie, alors que F n'en est pas une.

Démonstration. Il suffit de démontrer la première partie de l'énoncé puisque $F \equiv G$ ssi $F \leftrightarrow G$ est une tautologie. Pour cette première partie on utilise le lemme suivant.

Lemme 1.4.2 *Soit F une formule propositionnelle où n'apparait que les variables propositionnelles $p_1, \ldots, p_n$. Soit $H_1, ..., H_n$ des formules propositionnelles sur un ensemble $\mathscr{P}$ de variables (qui peut contenir ou non $p_1, \ldots, p_n$). Soit v une valuation sur $\mathscr{P}$ telle que $v(H_i) = \delta_i$ ($\delta_i \in \{0,1\}$, $i \in \{1, \ldots, n\}$). Alors pour la valuation v' définie sur $\{p_1, \ldots, p_n\}$ par $v'(p_i) = \delta_i$, pour $i \in \{1, ..., n\}$, vérifie*

$$v(F[H_1/p_1, \ldots, H_n/p_n]) = v'(F)\ .$$

Démonstration (lemme). Le lemme se montre par induction sur F. Par souci de lisibilité, on note $F[\bar{H}/\bar{p}]$ la formule $F[H_1/p_1, \ldots, H_n/p_n]$

— Si F est $\bot$, $v(\bot) = v'(\bot) = 0$.

— Si F est une variable propositionnelle p_i, $i \leqslant n$, on a : $F[\bar{H}/\bar{p}]$ est H_i, et $v'(p_i) = \delta_i = v(H_i)$.

— Si F est $\neg G$, avec par hypothèse d'induction $v(G[\bar{H}/\bar{p}]) = v'(G)$, alors $v(F[\bar{H}/\bar{p}]) = 1 - v(G[\bar{H}/\bar{p}]) = 1 - v'(G) = v'(F)$.

— Si F est $G_1 \wedge G_2$, avec par hypothèse d'induction $v(G_1[\bar{H}/\bar{p}]) = v'(G_1)$, $v(G_2[\bar{H}/\bar{p}]) = v'(G_2)$, alors $v(F[\bar{H}/\bar{p}]) = v(G_1[\bar{H}/\bar{p}]) \cdot v(G_2[\bar{H}/\bar{p}]) = v'(G_1) \cdot v'(G_2) = 1$. Le raisonnement est le même pour les autres connecteurs. ■

Fin de la démonstration (propriété de substitution). Supposons que F est une tautologie. Soit v une valuation quelconque sur $\mathscr{P}$. Par le lemme précédent, il existe v' valuation sur $\{p_1, \ldots, p_n\}$ telle que

$$v(F[H_1/p_1, \ldots, H_n/p_n]) = v'(F)\ .$$

Comme F est une tautologie, $v'(F) = 1$ et donc $v(F[H_1/p_1, \ldots, H_n/p_n]) = 1$. Toute valuation v satisfait bien $F[H_1/p_1, \ldots, H_n/p_n]$ qui est donc une tautologie. ■

Il s'agit d'une propriété finalement très intuitive et que l'on applique spontanément au cas par cas (voir les équivalences qui suivent). La démonstration du lemme permet de faire fonctionner une démonstration par induction sur la structure des formules. Il doit être bien clair qu'elle ne pose aucune difficulté autre que d'écriture. Dans la suite on ne rédige pas forcément ce genre de démonstration.

La propriété de substitution explique que l'on parle de variables propositionnelles, alors que, dès que l'on formalise il s'agit plutôt de constantes.

Dans la suite A, B et C désignent des formules quelconques. Voici quelques équivalences usuelles, qu'il suffit donc de vérifier pour des variables propositionnelles par substitution.

Négation des connecteurs usuels.

$$\neg\neg A \equiv A$$

Lois de de Morgan :

$$\neg(A \wedge B) \equiv (\neg A \vee \neg B)$$

$$\neg(A \vee B) \equiv (\neg A \wedge \neg B)$$

$$\neg(A \rightarrow B) \equiv A \wedge \neg B$$

$$\neg(A \leftrightarrow B) \equiv (A \leftrightarrow \neg B)$$

$$\neg(A \leftrightarrow B) \equiv (\neg A \leftrightarrow B)$$

Expression des connecteurs avec $\neg$, $\wedge$ et $\vee$.

$$\bot \equiv (A \wedge \neg A)$$

$$\top \equiv (A \vee \neg A)$$

$$(A \rightarrow B) \equiv (\neg A \vee B)$$

$$(A \leftrightarrow B) \equiv ((A \rightarrow B) \wedge (B \rightarrow A)) \equiv ((\neg A \vee B) \wedge (\neg B \vee A))$$

$$(A \leftrightarrow B) \equiv ((A \wedge B) \vee (\neg A \wedge \neg B))$$

Propriétés de la disjonction et de la conjonction.

commutativité :

$$(A \wedge B) \equiv (B \wedge A)$$

$$(A \vee B) \equiv (B \vee A)$$

associativité :

$$(A \wedge (B \wedge C)) \equiv ((A \wedge B) \wedge C)$$

$$(A \vee (B \vee C)) \equiv ((A \vee B) \vee C)$$

distributivité :

$$(A \wedge (B \vee C)) \equiv ((A \wedge B) \vee (A \wedge C))$$

$$(A \vee (B \wedge C)) \equiv ((A \vee B) \wedge (A \vee C))$$

idempotence :

$$(A \wedge A) \equiv A$$

$$(A \vee A) \equiv A$$

absorption :

$$(A \wedge \bot) \equiv \bot$$

$$(A \wedge (A \vee B)) \equiv A$$

$$(A \vee \top) \equiv \top$$

$$(A \vee (A \wedge B)) \equiv A$$

neutre :

$$(A \wedge \top) \equiv A$$

$$(A \vee \bot) \equiv A$$

$$(\neg A \wedge (A \vee B)) \equiv (\neg A \wedge B)$$

$$(\neg A \vee (A \wedge B)) \equiv (\neg A \vee B)$$

Remarque : ces deux dernières équivalences ainsi que celles d'absorption et d'idempotence permettent de simplifier les formes normales.

Propriétés de l'implication et de l'équivalence.

$$((A \wedge B) \rightarrow C) \equiv (A \rightarrow (B \rightarrow C))$$

contraposée :

$$(A \rightarrow B) \equiv (\neg B \rightarrow \neg A)$$

$$(A \leftrightarrow B) \equiv (\neg B \leftrightarrow \neg A)$$

Distributivité à droite :

$$(A \rightarrow (B \wedge C)) \equiv ((A \rightarrow B) \wedge (A \rightarrow C))$$

$$(A \rightarrow (B \vee C)) \equiv ((A \rightarrow B) \vee (A \rightarrow C))$$

Pseudo-distributivité à gauche :

$$((A \wedge B) \rightarrow C) \equiv ((A \rightarrow C) \vee (B \rightarrow C))$$

$$((A \vee B) \rightarrow C) \equiv ((A \rightarrow C) \wedge (B \rightarrow C))$$

Remarque : il n'y a pas de propriété analogue à ces 4 dernières pour l'équivalence.

$\bot$ et $\top$:

$$(A \rightarrow \bot) \equiv \neg A \quad (\bot \rightarrow A) \equiv \top \quad (A \rightarrow \top) \equiv \top \quad (\top \rightarrow A) \equiv A$$

$$(A \leftrightarrow \bot) \equiv \neg A \quad (A \leftrightarrow \top) \equiv A$$

Encore quelques équivalences.

$$(\neg A \rightarrow A) \equiv A \quad ((A \rightarrow B) \rightarrow A) \equiv A \quad ((A \rightarrow B) \rightarrow B) \equiv (A \vee B)$$

1.5 Formes normales.

1.5.1 Définitions.

Soit $\mathscr{P} = \{x_1, ..., x_n\}$ un ensemble de variable propositionnelle. On appelle *littéral* une constante propositionnelle ou une négation de constante propositionnelle. On parle de littéral négatif pour un littéral de la forme $\neg x_i$, $x_i \in \mathscr{P}$.

Une *clause* C est une formule propositionnelle de la forme :

$$C = \bigvee_{i \in A} x_i \vee \bigvee_{i \in B} \neg x_i$$

où $A, B \subseteq \{1, ..., n\}$. La *longueur* d'une clause est son nombre de littéraux (ici $|A| + |B|$). On le note $|C|$. Lorsque $|C| = 1$, on parle de clause *unitaire*.

De même, une *conjonction élémentaire* C est une formule propositionnelle de la forme [5] :

$$C = \bigwedge_{i \in A} x_i \wedge \bigwedge_{i \in B} \neg x_i$$

où $A, B \subseteq \{1, ..., n\}$. A nouveau, la *longueur* d'une conjonction élémentaire est son nombre de littéraux (ici $|A| + |B|$). On le note aussi $|C|$.

Une formule F est sous *forme normale disjonctive*, notée FND, (on dit parfois forme normale disjonctive-conjonctive) si elle s'écrit comme une disjonction de conjonctions de littéraux i.e. comme une disjonction de conjonctions élémentaires. Dans ce cas, F est sous FND si elle est de la forme :

$$F = \bigvee_{i=1}^{m} \left(\bigwedge_{i \in A_k} x_i \wedge \bigwedge_{i \in B_k} \neg x_i \right)$$

Si la longueur de chaque conjonction élémentaire est bornée par l, on dira que la formule est en l-FND.

5. Quand il n'y a pas ambiguité, on omet le parenthésage

Une formule est sous *forme normale conjonctive,* notée FNC, (on dit parfois forme normale conjonctive-disjonctive) si elle s'écrit comme une conjonction de disjonctions de littéraux i.e. comme une conjonction de clauses. Elle est de la forme :

$$F = \bigwedge_{i=1}^{m} (\bigvee_{i \in A_k} x_i \vee \bigvee_{i \in B_k} \neg x_i)$$

Si la longueur de chaque clause est bornée par l, on dira que la formule est en l-FNC.

La longueur d'une formule F en FNC (resp. FND) est égale à la somme de la longueur de ses clauses (resp. de ses conjonctions élémentaires). Ainsi, pour $F = \bigwedge_{i=1}^{m} C_i$, on a :

$$|F| = \sum_{i=1}^{m} |C_i|.$$

Enfin, on notera $\mathsf{cl}(F)$ l'ensemble $\{C_1, ..., C_m\}$ des clauses de F et $\mathsf{var}(F)$ l'ensemble des variables de F (qui est un sous-ensemble de $\mathscr{P}$).

Exemple (). Supposons que p, q, r soient des constantes propositionnelles, alors $p, q, r, \neg p, \neg q, \neg r$ sont des littéraux.

La formule $(p \wedge q \wedge r) \vee (\neg p \wedge r) \vee \neg r$ est sous forme normale disjonctive (la troisième conjonction est réduite à un élément).

La formule $(p \vee \neg q) \wedge \neg p \wedge (p \vee q \vee \neg r)$ est sous forme normale conjonctive (la deuxième disjonction est réduite à un élément).

La formule $p \vee \neg q$ est à la fois sous forme normale disjonctive (deux conjonctions réduites à un élément) et disjonctive (une conjonction réduite à une disjonction de deux littéraux), de même la formule $p \wedge q \wedge \neg r$.

1.5.2 Table de vérité et formes normales.

Voyons tout d'abord l'idée de la démonstration sur un exemple. Il s'agit tout simplement de décrire la table de vérité ligne par ligne, on obtient naturellement ainsi une forme normale disjonctive ou conjonctive suivant la façon dont on procède. Prenons la table de vérité de l'équivalence :

p	q	$p \leftrightarrow q$
0	0	1
0	1	0
1	0	0
1	1	1

On peut la décrire de deux façons. En énumérant les valuations qui rendent la formule vraie :

$$\begin{aligned} v(p \leftrightarrow q) = 1 \quad & \text{ssi } [v(p) = 0 \text{ et } v(q) = 0] \text{ ou } [v(p) = 1 \text{ et } v(q) = 1] \\ & \text{ssi } [v(\neg p) = 1 \text{ et } v(\neg q) = 1] \text{ ou } [v(p) = 1 \text{ et } v(q) = 1] \\ & \text{ssi } v(\neg p \wedge \neg q) = 1 \text{ ou } v(p \wedge q) = 1 \\ & \text{ssi } v((\neg p \wedge \neg q) \vee (p \wedge q)) = 1 \end{aligned}$$

on obtient une forme normale disjonctive de $p \leftrightarrow q$ qui est $(\neg p \wedge \neg q) \vee (p \wedge q)$.

En énumérant les valuations qui rendent la formule fausse :

$$\begin{aligned} v(p \leftrightarrow q) = 0 \quad & \text{ssi } [v(p) = 0 \text{ et } v(q) = 1] \text{ ou } [v(p) = 1 \text{ et } v(q) = 0] \\ & \text{ssi } [v(p) = 0 \text{ et } v(\neg q) = 0] \text{ ou } [v(\neg p) = 0 \text{ et } v(q) = 0] \\ & \text{ssi } v(p \vee \neg q) = 0 \text{ ou } v(\neg p \vee q) = 0 \\ & \text{ssi } v((p \vee \neg q) \wedge (\neg p \vee q)) = 0 \end{aligned}$$

on obtient une forme normale conjonctive de $p \leftrightarrow q$ qui est $(p \vee \neg q) \wedge (\neg p \vee q)$.

On montre maintenant que ce procédé fonctionne pour n'importe quelle table de vérité.

Les deux propriétés suivantes sont des conséquences immédiates de la définition de l'interprétation des conjonctions et disjonctions :

Lemme 1.5.1 *Soient $F_1, \ldots, F_n$ des formules propositionnelles. On a :*

$$\begin{aligned} v(F_1 \wedge \cdots \wedge F_n) = 1 \ \ & \textit{ssi} \ \ v(F_1) = 1 \text{ et } \ldots \text{ et } v(F_n) = 1 \\ v(F_1 \vee \cdots \vee F_n) = 0 \ \ & \textit{ssi} \ \ v(F_1) = 0 \text{ et } \ldots \text{ et } v(F_n) = 0 \end{aligned}$$

On utilise ces propriétés pour le lemme suivant :

Lemme 1.5.2 *Soit a une valuation, et n constantes propositionnelles $p_1, \ldots, p_n$. Il existe des formules F^1 et F^0 telles que pour toute valuation v :*

$$F^1 \text{ est une conjonction de littéraux et } v(F^1) = 1 \text{ ssi pour tout } i \in \{1,\ldots,n\}\ v(p_i) = a(p_i)$$
$$F^0 \text{ est une disjonction de littéraux et } v(F^0) = 0 \text{ ssi pour tout } i \in \{1,\ldots,n\}\ v(p_i) = a(p_i)$$

Démonstration. On pose $p_i^1 = p_i$ si $a(p_i) = 1$, $p_i^1 = \neg p_i$ si $a(p_i) = 0$. On pose $F^1 = p_1^1 \wedge \cdots \wedge p_n^1$. On a bien que $a(p_i^1) = 1$ pour tout i, donc $a(F^1) = 1$.

On a que $v(F^1) = 1$ ssi $v(p_i^1) = 1$ pour chaque $i \in \{1,\ldots,n\}$, d'où le résultat.

De même en posant $p_i^0 = p_i$ si $a(p_i) = 0$, $p_i^0 = \neg p_i$ si $a(p_i) = 1$, et $F^0 = p_1^0 \vee \cdots \vee p_n^0$, on a bien, $a(p_i^0) = 0$, pour $i \leqslant n$ et donc $a(F^0) = 0$. De plus, $v(F^0) = 0$ ssi $v(p_i^0) = 0$ pour chaque $i \in \{1,\ldots,n\}$, d'où le résultat. ■

Pour en déduire le théorème de mise sous forme normale, on utilise maintenant les propriétés duales du lemme 1.5.2 :

Lemme 1.5.3 *Soient $F_1, \ldots, F_n$ des formules propositionnelles. On a :*

$$v(F_1 \vee \cdots \vee F_n) = 1 \ \textit{ssi}\ v(F_1) = 1 \text{ ou } \ldots \text{ ou } v(F_n) = 1$$
$$v(F_1 \wedge \cdots \wedge F_n) = 0 \ \textit{ssi}\ v(F_1) = 0 \text{ ou } \ldots \text{ ou } v(F_n) = 0$$

Théorème 1.5.4 *Toute table de vérité, est la table de vérité d'une formule. Qui plus est on peut choisir cette formule sous forme normale conjonctive, ou sous forme normale disjonctive.*

Plus précisément, pour tout ensemble de n constantes propositionnelles $\{p_1, \ldots, p_n\}$, toute fonction de $\{0,1\}^{\{p_1,\ldots,p_n\}}$ dans $\{0,1\}$, il existe une formule sous forme normale conjonctive et une formule sous forme normale disjonctive dont c'est la table de vérité.

Démonstration. Soit t une table de vérité sur $p_1, \ldots, p_n$. Soit $v_0, \ldots, v_q$ toutes les valuations dont l'image est 1 par t. A chacune de ces valuations v_i on associe la formule F_i^1 donnée par le lemme 1.5.2. La formule $F_0^1 \vee \cdots \vee F_q^1$ est sous forme normale disjonctive et répond à la question. En effet $v(F_0^1 \vee \cdots \vee F_q^1) = 1$ ssi $v(F_0^1) = 1$ ou …ou $v(F_q^1) = 1$, et donc ssi v est l'une des valuation $v_0, \ldots, v_q$ d'après le lemme 1.5.2.

De façon analogue si $u_0, \ldots, u_r$ sont les valuations dont l'image est 0 par t, on associe à chacune d'entre elles la formule F_i^0 donnée au lemme 1.5.2. La formule $F_0^0 \wedge \cdots \wedge F_q^0$ est sous forme normale conjonctive et répond à la question, car $v(F_0^0 \wedge \cdots \wedge F_q^0) = 0$ ssi $v(F_0^0) = 0$ ou …ou $v(F_0^0) = 0$ i.e. ssi v est l'une des valuations $u_0, \ldots, u_r$. ■

Là encore on s'aperçoit que l'on a essentiellement utilisé les propriétés de la conjonction et de la disjonction dans le meta-langage. C'est à dire que l'on peut décrire une table de vérité à l'aide uniquement de conjonctions de disjonctions et de négations en décrivant toutes les valuations qui prennent la valeur "vrai" (forme normale disjonctive) ou la valeur "faux" (forme normale conjonctive).

La forme normale que l'on trouve à la lecture de la table de vérité, en suivant la méthode indiquée par la preuve du théorème précédent, n'est en général pas la plus courte! Ainsi $p \wedge q$ est sous forme normale conjonctive et disjonctive. La lecture de la table de vérité donnera bien $p \wedge q$ comme forme normale disjonctive, mais $(p \vee q) \wedge (p \vee \neg q) \wedge (\neg p \vee q)$ comme forme normale conjonctive.

1.5.3 Systèmes complets de connecteurs.

On dit qu'un système de connecteurs est *fonctionnellement complet*, ou parfois plus simplement *complet* si toute table de vérité est représentée par une formule utilisant ces seuls connecteurs. Un corollaire immédiat du théorème de mise sous-forme normale est que :

Corollaire 1.5.5 *Le système de connecteurs $\{\neg, \wedge, \vee\}$ est fonctionnellement complet.*

Des diverses équivalences parmi celles du paragraphe 1.4 on déduit que :

Corollaire 1.5.6 *Les systèmes de connecteurs $\{\neg, \wedge\}$, $\{\neg, \vee\}$, $\{\neg, \rightarrow\}$, $\{\bot, \rightarrow\}$ sont fonctionnellement complets.*

Démonstration. Pour chaque système S il suffit de montrer que chacun des connecteurs d'un système complet connu, pour le moment le seul connu est $\{\neg, \wedge, \vee\}$, s'exprime avec les connecteurs du système S.

Le système $\{\neg, \wedge\}$ est complet car $A \vee B \equiv \neg(\neg A \wedge \neg B)$ (loi de de Morgan).
Le système $\{\neg, \vee\}$ est complet car $A \wedge B \equiv \neg(\neg A \vee \neg B)$ (loi de de Morgan).
Le système $\{\neg, \rightarrow\}$ est complet car $A \vee B \equiv \neg A \rightarrow B$ et $\{\neg, \vee\}$ est complet.
Le système $\{\bot, \rightarrow\}$ est complet car $\neg A \equiv A \rightarrow \bot$ et $\{\neg, \rightarrow\}$ est complet. ■

1.5.4 Mise sous forme normale.

Étant donnée une formule, la mettre sous forme normale conjonctive, resp. disjonctive, consiste à trouver une formule équivalente sous forme normale conjonctive, resp. disjonctive.

— Une première méthode est de chercher la table de vérité puis d'en déduire la forme normale cherchée comme dans la preuve du théorème 1.5.4.
— Une seconde méthode un peu plus directe est d'utiliser convenablement les équivalences logiques de la section 1.4.

Ces méthodes ne sont pas très efficaces algorithmiquement. La méthode des tables de vérité nécessite d'inspecter l'ensemble des valuations et, utilisée telle quelle, son coût est prohibitif.

Pratiquement on a souvent besoin seulement d'une forme normale non pas équivalente mais *equisatisfaisable* à une formule donnée (l'une est satisfaite ssi l'autre l'est). Là il existe des algorithmes simples et efficaces que l'on décrira par la suite.

1.6 Formes normales complètes et bornes inférieures de FND

1.6.1 Impliquants, absorption et impliquants premiers

Soient $F \in \mathfrak{F}$ et C une conjonction élémentaire non vide de littéraux, sur $\mathscr{P}$. On dit que C est un *impliquant* de F si $(C \rightarrow F)$ est valide i.e. si pour toute valuation $\overline{v} \in \{0,1\}^{\mathscr{P}}$, $\overline{v}(C) = 1$ implique $\overline{v}(F) = 1$. De façon similaire, on dit que F *absorbe* C si $C \vee F \equiv F$. Il n'est pas dur de vérifier que C est un impliquant de F si et seulement si F absorbe C.

Soient C_1, C_2 des impliquants de F. On dit que C_1 est un *impliquant premier* de F si C_1 est absorbé par F et n'est absorbé par aucun autre impliquant C_2 de F.

Pour résumer :

$$\begin{array}{rl} C_1 \text{ est un impliquant de } C_2 \text{ ssi} & (C_1 \rightarrow C_2) \text{ est valide} \\ \text{ssi} & (C_1 \vee C_2) \equiv C_2 \\ \text{ssi} & C_2 \text{ absorbe } C_1. \end{array}$$

De même, C_1 est un impliquant premier de F ssi
— C_1 est un impliquant de F et
— pour tout impliquant $C_2 \neq C_1$ de F, $(C_1 \rightarrow C_2)$ n'est pas valide (i.e. $(C_1 \vee C_2) \not\equiv C_2$)

Par définition, tout impliquant est absorbé par un impliquant premier. Les impliquants premiers sont, en quelque sorte, les impliquants les plus "généraux" d'une formule [6]

Exemple (). $C_2 = (x_1 \wedge \neg x_2 \wedge x_3)$ est un impliquant de $C_1 = (x_1 \wedge \neg x_2)$ (C_1 absorbe C_2). Ils sont tous les deux impliquants de la formule $F = x_1 \vee (\neg x_2 \wedge \neg x_3)$. La clause $C = x_1$ est un impliquant premier de F.

Exemple (). Soient $C_1 = x$, $C_2 = (x \wedge y)$, $C_3 = (\neg x \wedge y)$ et $F = C_1 \vee C_2 \vee C_3$. C_1, C_2 et C_3 sont des impliquants de F. C_1 est un impliquant premier de F mais C_2 n'est pas un impliquant premier (car il implique C_1). On a donc $F \equiv C_1 \vee C_3$. La conjonction C_3 n'est pas non plus un impliquant premier de F car :
— $C_4 = y$, est un impliquant de F (le vérifier)
— C_3 implique C_4
On a donc : $F \equiv x \vee y$.

On laisse l'exercice suivant au lecteur. Bien que facile à démontrer, le résultat sera utilisé de nombreuses fois dans la suite.

6. Proposons une autre façon de voir cela. Appelons $\mathscr{M}$ (resp. $\mathscr{M}_i$) l'ensemble des valuations qui satisfont F (resp. C_i). Si C_1 est un impliquant de F alors cela implique que toutes les valuations qui satisfont C_1 satisfont aussi F donc que $\mathscr{M}_i \subseteq \mathscr{M}$. Si non seulement C_1 est un impliquant mais est premier alors pour tout autre impliquant C_2 de F : $\mathscr{M}_1 \not\subseteq \mathscr{M}_2$ i.e. C_1 "apporte" des valuations qui ne sont satisfaisantes pour aucun autre impliquant.

Exercice 1 Soit $C_1 = \bigwedge_{i\in A_1} x_i \wedge \bigwedge_{i\in B_1} \neg x_i$ et $C_2 = \bigwedge_{i\in A_2} x_i \wedge \bigwedge_{i\in B_2} \neg x_i$. Montrer que C_1 est un impliquant de C_2 (i.e. que C_2 absorbe C_1) si et seulement si $A_2 \subseteq A_1$ et $B_2 \subseteq B_1$.

Une conséquence immédiate de l'exercice 1 est que si C_1 est un impliquant de C_2 alors $|C_2| \leqslant |C_1|$.

Exercice 2 Soit $C = \bigwedge_{i\in I} l_i$ où chaque l_i est un littéral et F une formule propositionnelle. Montrer que C est un impliquant premier de F si et seulement si C est un impliquant de F et aucune des conjonctions $C_j = \bigwedge_{i\in I\setminus\{j\}} l_i$ n'est un impliquant de F.

La proposition suivante complète et raffine l'approche pour construire des FND par les tables de vérités. On remarque tout d'abord que pour toute FND

$$F = \bigvee_{i\in I} C_i,$$

C_i est un impliquant de F.

Proposition 1.6.1 *Toute formule peut être mise sous forme normale disjonctive dont les conjonctions élémentaires sont précisément ses impliquants premiers.*

Une telle forme FND est dite *complète*.

Démonstration. Soit F une formule propositionnelle sur un ensemble $\mathscr{P}$ fini. A équivalence logique près, le nombre de conjonctions élémentaires sur $\mathscr{P}$ est fini et donc le nombre d'impliquants premiers de F est fini. Soit $D_1, ..., D_m$ la liste des impliquants premiers de F. On sait que F admet au moins une FND. Posons $F \equiv \bigvee_{i\in I} C_i$. Comme F absorbe chacun de ses impliquants (donc les impliquants premiers) et par associativité de la disjonction, on a :

$$F \equiv \bigvee_{i\in I} C_i \vee \bigvee_{j=1}^{m} D_j.$$

Chaque C_i, $i \in I$, est un impliquant de F, il est donc absorbé par au moins un impliquant premier D_j, pour $j \in \{1, ..., m\}$. On a donc $C_i \vee D_j \equiv D_j$. Par absorption successive de tous les C_i on obtient que $F \equiv \bigvee_{j=1}^{m} D_j$. ■

Soit $F \equiv \bigvee_{i\in I} C_i$ une formule en FND. On dit que cette FND est *première* si chaque C_i est un impliquant premier de F et qu'elle est *non redondante* si, quel que soit $j \in I$:

$$F \not\equiv \bigvee_{i\in I\setminus\{j\}} C_i.$$

Une FND F est donc non redondante si, pour tout $j \in I$, il existe une valuation qui satisfait C_j mais pas $\bigvee_{i\in I\setminus\{j\}} C_i$. A contrario, F est *redondante* s'il existe $j \in I$ tel que toute valuation satisfaisant C_j satisfait aussi $\bigvee_{i\in I\setminus\{j\}} C_i$ i.e. si C_j est absorbée par $\bigvee_{i\in I\setminus\{j\}} C_i$. La notion de redondance généralise donc celle d'absorption.

Si une FND est complète alors elle est première. Par contre, elle peut être première sans être complète. En effet, une FND complète (ou simplement première) peut être redondante : certains de ses impliquants premiers peuvent être superflus. On peut alors éliminer certains impliquants premiers tout en conservant une formule première équivalente à la formule de départ.

Exemple (). La FND,

$$F = (x \wedge y) \vee (y \wedge z) \vee (z \wedge \neg x)$$

est première (vérifiez que chacune de ses clauses sont (parmi) ses impliquants premiers) mais elle est redondante. En effet, la clause $(y \wedge z)$ est superflue : toutes les valuations qui satisfont $(y \wedge z)$ satisfont aussi soit $(x \wedge y)$, soit $(z \wedge \neg x)$. Il s'en suit que $F \equiv (x \wedge y) \vee (z \wedge \neg x)$.

Toute FND d'une formule peut-être transformée en une FND première en remplaçant successivement chaque conjonction C_i qui n'est pas première par un impliquant premier qui l'absorbe. De même, à partir d'une FND, on peut obtenir une FND équivalente et non redondante en éliminant les clauses redondantes. On se sert de ces notions pour caractériser les formes FND "minimales" de certaines formules (ou tables de vérités).