Logique des propositions

Table des matières

1 Définitions

1.1 Syntaxe du calcul des propositions

Définition 1.1 : Soit P un ensemble quelconque dont les éléments sont appelés **variables propositionnelles**. L'ensemble des **formules du calcul des propositions** sur P est le plus petit ensemble noté $Prop(P)$ tel que :

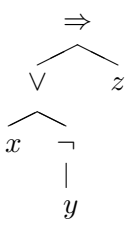
i) *Base* : $\mathcal{V}$, $\mathcal{F}$ ainsi que toute variable propositionnelle est une formule, *i.e.* $P \cup \{\mathcal{V}, \mathcal{F}\} \subset Prop(P)$.

ii) *Induction* : si f et g sont deux formules, alors $\neg f$, $f \vee g$, $f \wedge g$, $f \Rightarrow g$ sont des formules.

Remarque 1.2 : Pour bien mettre en évidence (et éviter les ambiguïtés) les étapes inductives utilisées pour la construction d'une formule, on s'autorise l'utilisation de parenthèses.

Remarque 1.3 : Comme pour tout ensemble défini inductivement, on associe à chaque élément de $Prop(P)$ un arbre de construction.

Exemple 1.4 : Soient $P = \{x, y, z\}$ et $\alpha = (x \vee \neg y) \Rightarrow z$ une formule de $Prop(P)$. L'arbre associé à α est le suivant :



1.2 Sémantique

Définition 1.5 : On appelle **valuation** toute application $\delta : P \to \{0,1\}$. Une telle application associe à chaque variable propositionnelle une **valeur de vérité**.

On considère désormais 4 opérations que l'on a déjà découvertes dans le cadre de l'algèbre booléenne :

$$non_{\mathbb{B}} : \begin{array}{ccc} \{0,1\} & \to & \{0,1\} \\ x & \mapsto & \begin{cases} 1 \text{ si } x = 0 \\ 0 \text{ si } x = 1 \end{cases} \end{array} \qquad \times_{\mathbb{B}} : \begin{array}{ccc} \{0,1\}^2 & \to & \{0,1\} \\ (x,y) & \mapsto & \begin{cases} 1 \text{ si } x = 1 \text{ et } y = 1 \\ 0 \text{ sinon} \end{cases} \end{array}$$

$$+_{\mathbb{B}} : \begin{array}{ccc} \{0,1\}^2 & \to & \{0,1\} \\ (x,y) & \mapsto & \begin{cases} 1 \text{ si } x = 1 \text{ ou } y = 1 \\ 0 \text{ sinon} \end{cases} \end{array} \qquad \Rightarrow_{\mathbb{B}} : \begin{array}{ccc} \{0,1\}^2 & \to & \{0,1\} \\ (x,y) & \mapsto & \begin{cases} 1 \text{ si } x = 0 \text{ ou } y = 1 \\ 0 \text{ sinon} \end{cases} \end{array}$$

Définition 1.6 : On prolonge δ en une application $[\![\, . \,]\!]_\delta$ de $Prop(P)$ dans $\{0,1\}$ de la façon suivante :

i) **Base :** $[\![\mathcal{V}]\!]_\delta = 1$, $[\![\mathcal{F}]\!]_\delta = 0$ et $\forall \alpha \in P$, $[\![\alpha]\!]_\delta = \delta(\alpha)$

ii) **Induction :** $\forall f, g \in Prop(P)$,
$$\begin{cases} [\![\neg f]\!]_\delta &=& non_{\mathbb{B}} \, [\![f]\!]_\delta \\ [\![f \vee g]\!]_\delta &=& [\![f]\!]_\delta +_{\mathbb{B}} [\![g]\!]_\delta \\ [\![f \wedge g]\!]_\delta &=& [\![f]\!]_\delta \times_{\mathbb{B}} [\![g]\!]_\delta \\ [\![f \Rightarrow g]\!]_\delta &=& [\![f]\!]_\delta \Rightarrow_{\mathbb{B}} [\![g]\!]_\delta \end{cases}$$

Définition 1.7 : Soit $\alpha \in Prop(P)$ une formule du calcul des propositions

(1) on dit que $\delta : P \to \{0,1\}$ est un modèle de α ssi $[\![\alpha]\!]_\delta = 1$

(2) on dit que α est une tautologie ssi toute valuation est un modèle de α :
$$\forall \delta : P \to \{0,1\}, \; [\![\alpha]\!]_\delta = 1$$

(3) on dit que α est satisfiable ssi il existe un modèle de α :
$$\exists \delta : P \to \{0,1\}, \; [\![\alpha]\!]_\delta = 1$$

(4) on dit que α est contradictoire ssi α n'a aucun modèle
$$\forall \delta : P \to \{0,1\}, \; [\![\alpha]\!]_\delta = 0$$

Exemple 1.8 : Soient $P = \{x, y, z\}$ et $\alpha = (x \vee \neg y) \Rightarrow z$ une formule de $Prop(P)$.
$\delta_1 : \begin{cases} x \mapsto 0 \\ y \mapsto 0 \\ z \mapsto 1 \end{cases}$ est un modèle de α alors que $\delta_2 : \begin{cases} x \mapsto 1 \\ y \mapsto 0 \\ z \mapsto 0 \end{cases}$ n'en est pas un. En effet :

$\delta(x)$	$\delta(y)$	$\delta(z)$	$[\![x \vee \neg y]\!]_\delta$	$[\![\alpha]\!]_\delta$
0	0	1	1	1
1	0	0	1	0

Définition 1.9 : La définition 1.6 nous permet de définir, pour tout $\alpha \in Prop(P)$, l'application partielle :

$$\begin{array}{rccc} [\![\alpha]\!] : & Prop(P) \to \{0,1\} & \to & \{0,1\} \\ & \delta & \mapsto & [\![\alpha]\!]_\delta \end{array}$$

appelée **sémantique** de α.

Définition 1.10 : On définit l'égalité et la relation d'ordre partielle $\leq$ entre deux sémantiques de la façon suivante :

$$\begin{array}{rcl} [\![\alpha]\!] = [\![\beta]\!] & \Leftrightarrow & \forall \delta : Prop(P) \to \{0,1\},\ [\![\alpha]\!]_\delta = [\![\beta]\!]_\delta \\ [\![\alpha]\!] \leq [\![\beta]\!] & \Leftrightarrow & \forall \delta : Prop(P) \to \{0,1\},\ [\![\alpha]\!]_\delta \leq [\![\beta]\!]_\delta \end{array}$$

Remarque 1.11 : On dira que deux formules sont **équivalentes** si leurs sémantiques sont égales.

Remarque 1.12 : Il existe $card(\{0,1\})^{card(Prop(P) \to \{0,1\})} = card(\{0,1\})^{card(\{0,1\}^{Prop(P)}} = 2^{2^{Prop(P)}}$ sémantiques différentes.

Définition 1.13 : Soit $\Gamma \subset Prop(P)$ un ensemble de formules du calcul des propositions. On dit que $\delta : P \to \{0,1\}$ est un modèle de Γ ssi δ est un modèle de toutes les formules de Γ, *i.e.* :

$$\forall \alpha \in \Gamma,\ [\![\alpha]\!]_\delta = 1$$

Remarque 1.14 : Il vient de cette définition que tout modèle d'un ensemble Γ est modèle pour tout sous-ensemble de Γ. On en déduit que toute valuation est modèle de l'ensemble vide $\varnothing$!

Remarque 1.15 : On dit qu'un ensemble de formules Γ est satisfiable ssi il existe un modèle de Γ.

Remarque 1.16 : Un ensemble de formules peut être contradictoire (non satisfiable) alors que tous ces éléments sont satisfiables.

Définition 1.17 : Soit $\Gamma \subset Prop(P)$ et α une formule du calcul des propositions. On dit que α **se déduit sémantiquement** de Γ et on note

$$\Gamma \models \alpha$$

ssi tout modèle de Γ est un modèle de α :

$$\forall \delta : P \to \{0,1\},\ (\forall \beta \in \Gamma, [\![\beta]\!]_\delta = 1) \Rightarrow [\![\alpha]\!]_\delta = 1$$

Exemple 1.18 : Soient $P = \{x, y\}$ et $\Gamma = \{x,\ x \Rightarrow y\}$. Montrons que $\Gamma \models y$.

$\delta(x)$	$\delta(y)$	$[\![x]\!]_\delta$	$[\![x \Rightarrow y]\!]_\delta$	δ modèle de Γ	$[\![y]\!]_\delta$
0	0	0	1	0	0
0	1	0	1	0	1
1	0	1	0	0	0
1	1	1	1	1	1

L'unique modèle de Γ $(\delta : x \mapsto 1,\ y \mapsto 1)$ est modèle de y, on a donc : $\Gamma \models y$.

Remarque 1.19 : Des deux définitions précédentes on déduit qu'une formule α est une tautologie si et seulement si $\varnothing \models \alpha$, ce que l'on écrira désormais $\models \alpha$.

Proposition 1.20 : $\Gamma \models \alpha$ ssi $\Gamma \cup \{\neg\alpha\}$ est contradictoire.

▷ **Démonstration :** *Supposons l'existence d'une valuation δ qui soit modèle de $\Gamma \cup \{\neg\alpha\}$. On a, par définition :*

$$\forall \beta \in \Gamma,\ [\![\beta]\!]_\delta = 1 \ \textit{et}\ [\![\neg\alpha]\!]_\delta = 1$$

Soit :

$$\forall \beta \in \Gamma,\ [\![\beta]\!]_\delta = 1 \ \textit{et}\ [\![\alpha]\!]_\delta = 0$$

Or, $\forall \beta \in \Gamma,\ [\![\beta]\!]_\delta = 1 \Rightarrow [\![\alpha]\!]_\delta = 1$. On a donc une contradiction. ■

Proposition 1.21 : $\Gamma \cup \{\alpha\} \models \beta$ ssi $\Gamma \models (\alpha \Rightarrow \beta)$

Théorème 1.22 : (de finitude ou de compacité) Soit $\Gamma \subset prop(P)$, alors :

- Γ est satisfiable ssi tout sous-ensemble fini de Γ est satisfiable :

$$\left(\exists \delta,\ \forall \beta \in \Gamma,\ [\![\beta]\!]_\delta = 1\right) \Leftrightarrow \left(\forall \Gamma' \subset \Gamma,\ card\left(\Gamma'\right) < \infty, \exists \delta, \forall \beta \in \Gamma',\ [\![\beta]\!]_\delta = 1\right)$$

- Une formule se déduit sémantiquement d'un ensemble de formules ssi elle se déduit sémantiquement de tout sous-ensemble fini de cet ensemble :

$$\forall \alpha \in Prop(P),\ \left(\Gamma \models \alpha\right) \Leftrightarrow \left(\forall \Gamma' \subset \Gamma,\ card\left(\Gamma'\right) < \infty,\ \Gamma' \models \alpha\right)$$

- Γ est contradictoire ssi l'un de ses sous-ensembles finis est contradictoire :

$$\left(\nexists \delta,\ \forall \beta \in \Gamma,\ [\![\beta]\!]_\delta = 1\right) \Leftrightarrow \left(\exists \Gamma' \subset \Gamma,\ card\left(\Gamma'\right) < \infty, \nexists \delta, \forall \beta \in \Gamma',\ [\![\beta]\!]_\delta = 1\right)$$

1.3 Exemple et limites du cadre sémantique

Soit $P = \{a, b, c\}$. On considère la formule $\alpha \in Prop(P)$ suivante :

$$\alpha = \Big[(a \Rightarrow b) \wedge \big((c \vee \neg b) \Rightarrow a\big)\Big] \Rightarrow (b \vee c)$$

On cherche à savoir si cette formule est une tautologie : il nous faut donc étudier $[\![\alpha]\!]_\delta$ pour toute valuation δ :

$\delta(a)$	$\delta(b)$	$\delta(c)$	$[\![\alpha]\!]_\delta$
0	0	0	1
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	0	1
1	1	1	1

Une valuation étant une application de P dans $\{0,1\}$, il existe $card\big(\{0,1\}\big)^{card(P)} = 2^{card(P)}$ valuations sur P. Déterminer si une formule est une tautologie devient alors très coûteux dès lors que le cardinal de P devient important. Le nombre d'opérations à réaliser pour obtenir le résultat est alors (grossièrement) $2^{card(P)}\times$ le nombre d'opérateurs dans la formule. On cherche donc un moyen automatique (**syntaxique**) permettant de déterminer si une formule est une tautologie sans avoir à passer par le calcul de valuations. Plus généralement, ce problème se pose dès que l'on cherche à savoir si une formule se déduit d'un ensemble de formules, *i.e.* lorsque l'on cherche à montrer que $\Gamma \models \alpha$ quelque soient $\Gamma \subset Prop(P)$ et $\alpha \in Prop(P)$.

2 Systèmes formels, déduction syntaxique

Une autre approche consiste à se donner un ensemble de formules que l'on considère comme étant « correctes » ainsi qu'un ensemble de règles permettant de « déduire » des formules d'autres formules. C'est l'objet des systèmes formels.

Définition 2.1 : On appelle **système formel** tout triplet $\mathcal{S} = (\mathcal{F}, \mathcal{A}, \mathcal{R})$ où
- $\mathcal{F}$ est un ensemble de formules
- $\mathcal{A} \subset \mathcal{F}$ est un ensemble dont les éléments sont appelés **axiomes**
- $\mathcal{R}$ est un ensemble de relations appelées **règles de déduction** (ou **règles d'inférence**).

Définition 2.2 : Soit $\mathcal{S} = (\mathcal{F}, \mathcal{A}, \mathcal{R})$ un système formel.
$\forall r \in \mathcal{R},\ f_1, \ldots, f_n, g \in \mathcal{F}$:

$$\frac{f_1\ ,\ \ldots\ ,\ f_n}{g}(r)$$

signifie que « g se **déduit syntaxiquement** de $f_1, \ldots, f_n$ par la règle r ». On dit que g est la **conclusion** de la déduction et $\Gamma = \{f_1, \ldots, f_n\}$ l'ensemble des **prémisses** ou **hypothèses**.

Définition 2.3 : Soient un système formel $\mathcal{S} = (\mathcal{F}, \mathcal{A}, \mathcal{R})$, un ensemble de formules $\Gamma \subset \mathcal{F}$ et une formule α. On appelle **preuve** de α dans $\mathcal{S}$ à partir de Γ toute suite de déductions syntaxiques dont les prémisses sont des éléments de $\mathcal{A} \cup \Gamma$ ou des conclusions de déductions précédentes. On dit que α est la **conclusion** de la preuve et on note :

$$\Gamma \vdash_{\mathcal{S}} \alpha$$

Remarque 2.4 : Soient $\mathcal{S} = (\mathcal{F}, \mathcal{A}, \mathcal{R})$ un système formel et $\alpha \in \mathcal{F}$ une formule. On dit que α est un théorème du système formel $\mathcal{S}$ ssi α se démontre dans $\mathcal{S}$ en utilisant uniquement les axiomes, *i.e* ssi

$$\varnothing \vdash_{\mathcal{S}} \alpha$$

ce que l'on notera

$$\vdash_{\mathcal{S}} \alpha$$

Définition 2.5 : On appelle **système formel du calcul des propositions** tout système formel $\mathcal{S} = (\mathcal{F}, \mathcal{A}, \mathcal{R})$ où $\mathcal{F} = Prop(P)$ (avec P un ensemble quelconque de variables propositionnelles) *i.e.* tout système formel travaillant sur le langage des formules du calcul des propsitions.

Définition 2.6 : Soit $\mathcal{S} = (Prop(P), \mathcal{A}, \mathcal{R})$ un système formel du calcul des propositions. On dit que $\mathcal{S}$ est **valide** (ou **correct**) ssi $\forall \Gamma \subset Prop(P),\ \alpha \in Prop(P)$:

$$\Gamma \vdash_{\mathcal{S}} \alpha \quad \Rightarrow \quad \Gamma \models \alpha$$

Autrement dit, toute preuve dans le système formel $\mathcal{S}$ est « valide » sémantiquement (le système formel ne fait que des opérations « correctes »).

Remarque 2.7 : En particulier, si un système formel $\mathcal{S}$ est valide, alors tout théorème de $\mathcal{S}$ est une tautologie.

Définition 2.8 : Soit $\mathcal{S} = (Prop(P), \mathcal{A}, \mathcal{R})$ un système formel du calcul des propositions. On dit que $\mathcal{S}$ est **complet** ssi $\forall \Gamma \subset Prop(P),\ \alpha \in Prop(P)$:

$$\Gamma \models \alpha \quad \Rightarrow \quad \Gamma \vdash_{\mathcal{S}} \alpha$$

Autrement dit, si α se déduit sémantiquement de Γ, alors on peut trouver une preuve de α dans $\mathcal{S}$ à partir de Γ.

3 Forme clausale d'une formule

3.1 Définitions

Définition 3.1 : Soit P un ensemble de variables propositionnelles. On appelle **atome** ou **littéral** tout objet de la forme x ou $\neg x$ où $x \in P$.

Remarque 3.2 : Un littéral est dit **positif** s'il s'agit d'une variable propositionnelle et **négatif** s'il s'agit d'une négation de variable propositionnelle.

Définition 3.3 : Soit P un ensemble de variables propositionnelles. On appelle **clause** sur P toute disjonction de littéraux, *i.e.* tout objet de la forme :

$$x_1 \vee \ldots \vee x_k \vee \neg x_{k+1} \vee \ldots \vee \neg x_n$$

où $x_1,\ \ldots,\ x_k$ sont les littéraux positifs et $\neg x_{k+1},\ \ldots,\ \neg x_n$ les littéraux négatifs.

Définition 3.4 : (Relation d'ordre de subsomption) On dit qu'une clause c_1 **est subsummée par** c_2 et l'on note

$$c_2 \subseteq c_1$$

ssi l'une des conditions suivantes équivalentes est vérifiée :
- tout littéral de c_2 apparaît dans c_1
- il existe une clause c_3 telle que $c_1 = c_2 \vee c_3$
- tout modèle de c_2 est modèle de c_1

Définition 3.5 : On appelle **forme clausale** d'une formule $\alpha \in Prop(P)$ un ensemble de clauses $c_1, \ldots, c_n$ dont la conjonction est équivalente à α, *i.e.* :

$$Cl(\alpha) = \{c_1, \ldots, c_n\} \quad \Leftrightarrow \quad [\![c_1 \wedge \ldots \wedge c_n]\!] = [\![\alpha]\!]$$

Remarque 3.6 : Soit P un ensemble de variables propositionnelles. On note $Cl(P)$ l'ensemble des formes clausales sur P.

Remarque 3.7 : Soient deux ensembles de clauses C_1 et C_2.

$$C_1 \subseteq C_2 \;\Rightarrow\; \forall \delta : Prop(P) \to \{0,1\},\; \left[\!\!\left[\bigwedge_{c \in C_1} c \right]\!\!\right]_\delta \geq \left[\!\!\left[\bigwedge_{c \in C_2} c \right]\!\!\right]_\delta$$

On en déduit notamment que toute valuation est modèle de l'ensemble vide de clause $\varnothing$, d'où la proposition suivante :

Proposition 3.8 : $\alpha \in Prop(P)$ est une tautologie ssi $Cl(\alpha) = \varnothing$.

Remarque 3.9 : A contrario, on note $\square$ la clause non-satisfiable (ne possédant aucun modèle) :

$$\forall \delta : Prop(P) \to \{0,1\},\; [\![\square]\!]_\delta = 0$$

Remarque 3.10 : Pour plus de lisibilité, l'ensemble de clauses $\{c_1, \ldots, c_n\}$ est noté $\begin{cases} c_1 \\ \vdots \\ c_n \end{cases}$

Remarque 3.11 : Soit un ensemble de clause $C = \begin{cases} c_1 \\ \vdots \\ c_n \end{cases}$. S'il existe une clause c_i telle que $[\![c_i]\!] = 1$ (*i.e.* $\forall \delta,\; [\![c_i]\!]_\delta = 1$), alors C est équivalent à $C \setminus \{c_i\}$

Proposition 3.12 : Soit c une clause sur P. S'il existe une variable propositionnelle $x \in P$ telle que les littéraux x et $\neg x$ apparaissent dans c (*i.e.* si c est subsummée par x et $\neg x$), alors $[\![c]\!] = 1$.

Exemple 3.13 : Soient $P = \{x, y, z\}$ et $C = \left\{ \begin{array}{l} x \vee z \\ y \vee x \vee \neg y \\ z \vee \neg z \end{array} \right.$.

C est équivalent à $\left\{ \; x \vee z \; \right.$.

Remarque 3.14 : On s'autorise à écrire $C = C'$ lorsque C est équivalent à C'.

3.2 Algorithme de mise sous forme clausale

Théorème 3.15 : Toute formule du calcul des propositions admet une forme clausale.

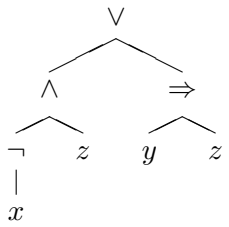
Algorithme 3.16 : Puisque toute formule du calcul des propositions admet une forme clausale, on peut définir un algorithme qui à toute formule associe sa forme clausale de façon inductive sur la structure de $Prop(P)$:

- *Base* : $Cl(\mathcal{V}) = \varnothing,\ Cl(\mathcal{F}) = \{\square\},\ \forall x \in Prop(P),\ Cl(x) = \{x\}$
- *Induction* : si f et g sont deux formules, alors :
  - $Cl(f \vee g) = \{c_1 \vee c_2 \mid c_1 \in Cl(f),\ c_2 \in Cl(g)\}$
  - $Cl(f \wedge g) = Cl(f) \cup Cl(g)$
  - $Cl(f \Rightarrow g) = Cl(\neg f \vee g)$
  - $Cl(\neg f) = \overline{Cl}(f)$

où $\overline{Cl}$ est défini de la façon suivante :

- *Base* : $\overline{Cl}(\mathcal{V}) = \{\square\},\ \overline{Cl}(\mathcal{F}) = \varnothing,\ \forall x \in Prop(P),\ \overline{Cl}(x) = \{\neg x\}$
- *Induction* : si f et g sont deux formules, alors :
  - $\overline{Cl}(f \vee g) = Cl((\neg f) \wedge (\neg g))$
  - $\overline{Cl}(f \wedge g) = Cl((\neg f) \vee (\neg g))$
  - $\overline{Cl}(f \Rightarrow g) = Cl(f \wedge (\neg g))$
  - $\overline{Cl}(\neg f) = Cl(f)$

Exemple 3.17 : Soient $P = \{x, y, z\}$ et $\alpha = (\neg x \wedge z) \vee (y \Rightarrow z) \in Prop(P)$. Pour mieux visualiser l'ordre des « appels » à effectuer, on dresse l'arbre de construction de α :



Ainsi, on a :

(i) $Cl(\alpha) = \{c_1 \vee c_2 \mid c_1 \in Cl(\neg x \wedge z),\ c_2 \in Cl(y \Rightarrow z)\}$

(ii) $Cl(\neg x \wedge z) = Cl(\neg x) \cup Cl(z)$

(iii) $Cl(\neg x) = \overline{Cl}(x) = \{\neg x\}$

(iv) $Cl(z) = \{z\}$

(v) $Cl(y \Rightarrow z) = Cl(\neg y \vee z) = \{c_1 \vee c_2 \mid c_1 \in Cl(\neg y),\ c_2 \in Cl(z)\}$

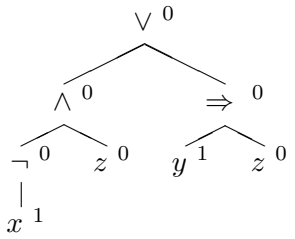
(vi) $Cl(\neg y) = \overline{Cl}(y) = \{\neg y\}$

Soit $(v) = \{\neg y \vee z\}$, $(ii) = \left\{ \begin{array}{l} \neg x \\ z \end{array} \right.$ donc $Cl(\alpha) = \left\{ \begin{array}{l} \neg y \vee z \vee x \\ \neg y \vee z \end{array} \right.$

Remarque 3.18 : Le calcul de la forme clausale d'une formule nécessite autant d'appels que d'étapes inductives nécessaires à la construction de la formule. On décide alors de travailler « directement » sur l'arbre de construction de la formule en associant à chaque noeud un attribut :

- égal à 0 si l'on doit calculer $Cl(\alpha)$ où α représente la (sous-)formule correspondante à l'arbre dont le noeud que l'on étudie est la racine
- égal à 1 si l'on doit calculer $\overline{Cl}(\alpha)$

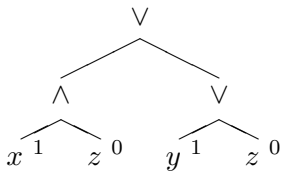
Exemple 3.19 : En reprenant l'exemple précédent, on obtient l'arbre suivant :



Remarque 3.20 : On effectue ensuite les transformations des opérateurs induites par l'algorithme 3.16 :

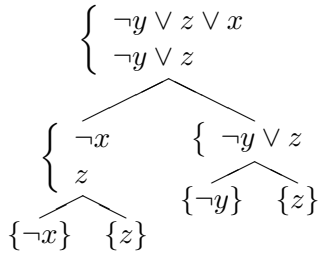
symbole d'origine	symbole de substitution
$\Rightarrow^0$	$\vee$
$\Rightarrow^1$	$\wedge$
$\vee^0$	$\vee$
$\vee^1$	$\wedge$
$\wedge^0$	$\wedge$
$\wedge^1$	$\vee$
$\neg$	$aucun$($\neg$ est supprimé)

Exemple 3.21 : En reprenant l'exemple précédent, on obtient :



Remarque 3.22 : Désormais, seules les feuilles, qui sont des variables propositionnelles, possèdent encore un attribut. Comme $\forall x \in P,\ Cl(x) = \{x\}$ et $\overline{Cl}(x) = \{\neg x\}$, on remplace x^0 par $\{x\}$ et x^1 par $\{\neg x\}$. On « remonte » ensuite dans l'arbre (en partant des feuilles) jusqu'à la racine en effectuant les opérations de l'algorithme 3.16. On note chaque résultat intermédiaire au noeud auquel il est associé.

Exemple 3.23 : Ainsi :



4 La Méthode de Résolution

Remarque 4.1 : Le système formel que nous considérons n'a pas pour but de démontrer une formule quelconque mais seulement la clause vide $\square$.

Théorème 4.2 : Soit $\Gamma \subset \text{Prop(P)}$. On a $\Gamma \models \alpha$ ssi l'union des ensembles de clauses associées aux formules de $\Gamma \cup \{\neg\alpha\}$ est contradictoire.

Théorème 4.3 : (Théorème de Robinson) Soit C un ensemble de clause, cet ensemble est contradictoire ssi en prenant C comme ensemble de prémisses, on peut déduire $\square$ en utilisant la résolution comme seule règle de déduction.

Définition 4.4 : (Règle de résolution)

$$\frac{a \vee A, \neg a \vee B}{A \vee B}(resolution)$$

Ce qui donne en français :
Soient C_1 et C_2 deux clauses appartenant à une formule F mise sous forme clausale. S'il existe un atome a tel que $a \in C_1$ et $\neg a \in C_2$ alors la clause $R \equiv C_1 \setminus \{a\} \cup C_2 \setminus \{\neg a\}$ dite résolvante de C_1 et C_2 est une conséquence logique de F.

Remarque 4.5 : On dit que deux clauses peuvent entrer en résolution s'il en existe une troisième qui s'en déduit par la règle de résolution. Deux clauses peuvent entrer en résolution ssi elles ont un même littéral "a" qui est positif dans l'une et négatif dans l'autre. Alors la troisième clause, déduite des deux premières par résolution, est obtenue en recopiant à la suite des deux clauses de départ sauf le littéral commun qui est effacé.

À compléter