Université Libre de Bruxelles

INFO-F302
Informatique Fondamentale

Synthèse

Étudiants :
Hugo Callens

Enseignants :
E. Filiot

31 octobre 2023

Contents

1 Logique propositionnelle

1.1 Construction de formules

Le vocabulaire du langage de la logique propositionnelle est composé de :

1. de propositions x, y, z, ... ; ou X, Y, Z, ... ;
2. de deux constantes vrai ($\top$ ou 1) et faux ($\bot$ ou 0) ;
3. d'un ensemble de connecteurs logiques : $\neg$, $\wedge$, $\vee$, $\rightarrow$, $\leftrightarrow$.
4. de paranthèses ( ).

1.2 Sémantique

Définition 1.1. Sémantique

La sémantique d'une formule est la valeur de vérité de cette formule. La valeur de vérité d'une formule Φ formée àpd propositions d'un ensemble X, évaluée avec la fonction d'interprétation V, est notée $[\![\Phi]\!]_V$. La fonction $[\![\Phi]\!]_V$ est définie par induction sur la syntaxe de Φ de la façon suivante :

- $[\![\top]\!]_V = 1$; $[\![\bot]\!]_V = 0$; $[\![x]\!]_V = V(x)$
- $[\![\neg\Phi]\!]_V = 1 - [\![\Phi]\!]_V$
- $[\![\Phi_1 \vee \Phi_2]\!]_V = \max([\![\Phi_1]\!]_V, [\![\Phi_2]\!]_V)$
- $[\![\Phi_1 \wedge \Phi_2]\!]_V = \min([\![\Phi_1]\!]_V, [\![\Phi_2]\!]_V)$
- $[\![\Phi_1 \leftarrow \Phi_2]\!]_V = \max(1 - [\![\Phi_1]\!]_V, [\![\Phi_2]\!]_V)$
- $[\![\Phi_1 \leftrightarrow \Phi_2]\!]_V = \min([\![\Phi_1 \rightarrow \Phi_2]\!]_V, [\![\Phi_2 \rightarrow \Phi_1]\!]_V)$

Nous notons $V \vDash \Phi \Leftrightarrow [\![\Phi]\!]_V = 1$ soit "V satisfait Φ."

L'information contenue dans la définition est souvent représentée sous forme de table de verité :

Φ_1	Φ_2	$\Phi_1 \vee \Phi_2$	$\Phi_1 \wedge \Phi_2$	$\Phi_1 \rightarrow \Phi_2$	$\Phi_1 \leftrightarrow \Phi_2$
0	0	0	0	1	1
0	1	1	0	1	0
1	0	1	0	0	0
1	1	1	1	1	1

Dans l'implication suivante : $\Phi_1 \rightarrow \Phi_2$, la cas où Φ_1 est faux ne nous intéresse pas. Dans ce cas, l'implication est toujours vraie.

1.3 Validité et Stabilité

1.3.1 Définitions

Définition 1.2. Formule propositionnelle satisfaisable

Une formule propositionnelle Φ est **satisfaisable** $\Leftrightarrow$ il existe une fonction d'interprétation V pour les propositions de Φ, telle que $V \vDash \Phi$.

Définition 1.3. Formule propositionnelle valide

Une formule propositionnelle Φ est **valide** $\Leftrightarrow$ pour toute fonction d'interprétation V pour les propositions de Φ, $V \vDash \Phi$.

1.3.2 Conséquence logique

Définition 1.4. Conséquence Logique

Soit $\Phi_1, ..., \Phi_n, \Phi$ des formules. On dira que Φ est une **conséquence logique** de $\Phi_1, ..., \Phi_n$, noté $\Phi_1, ..., \Phi_n \vDash \Phi$, si $(\Phi_1 \wedge ... \wedge \Phi_n) \rightarrow \Phi$ est valide.

1.3.3 Equivalence

Définition 1.5. Formules équivalentes

Deux formules, Φ et Ψ, sont **équivalentes** si la formule $\Phi \leftrightarrow \Psi$ est valide. On notera $\Phi \equiv \Psi$.

Pour toutes formules Φ_1, Φ_2, Φ_3 :

- $\neg\neg\Phi_1 \equiv \Phi_1$
- $\neg(\Phi_1 \wedge \Phi_2) \equiv (\neg\Phi_1 \vee \neg\Phi_2)$
- $\neg(\Phi_1 \vee \Phi_2) \equiv (\neg\Phi_1 \wedge \neg\Phi_2)$
- $\Phi_1 \wedge (\Phi_2 \vee \Phi_3) \equiv (\Phi_1 \wedge \Phi_2) \vee (\Phi_1 \wedge \Phi_3)$
- $\Phi_1 \vee (\Phi_2 \wedge \Phi_3) \equiv (\Phi_1 \vee \Phi_2) \wedge (\Phi_1 \vee \Phi_3)$
- $\Phi_1 \rightarrow \Phi_2 \equiv (\neg\Phi_1 \vee \Phi_2)$

1.3.4 Lien entre satisfaisabilité et validité

Théorème 1.1. Lien entre satisfaisabilité et validité

Une formule Φ est valide $\Leftrightarrow$ $\neg\Phi$ est insatisfaisable.

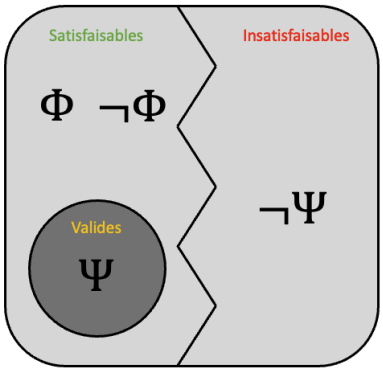

FIGURE 1.1 – Lien entre satisfaisabilité et validité

1.3.5 Tableaux sémantiques

Définition 1.6. Littéral

Un littéral est une proposition x ou la négation d'une proposition $\neg x$.

La méthode des tableaux sémantiques est un algorithme pour tester la satisfaisabilité d'une

formule. Elle consiste à construire un arbre dont les noeuds sont des formules et les feuilles sont des littéraux. On construit l'arbre de la façon suivante :

- On place la formule à tester à la racine de l'arbre.
- On applique les règles suivantes jusqu'à ce que l'arbre soit complet :
  - Si la formule à tester est une constante, on arrête.
  - Si la formule à tester est une conjonction, on ajoute les deux conjoncteurs comme fils de la formule à tester.
  - Si la formule à tester est une disjonction, on ajoute un fils avec le premier disjoncteur et un autre fils avec le deuxième disjoncteur.
  - Si la formule à tester est une implication, on ajoute un fils avec la négation de l'antécédent et un autre fils avec le conséquent.
  - Si la formule à tester est une équivalence, on ajoute un fils avec la négation de la première formule et un autre fils avec la deuxième formule.
  - Si la formule à tester est une négation, on ajoute un fils avec la négation de la formule à tester.

Remarque: algorithme généré par copilot.

2 Déduction naturelle

Définition 2.1. Déduction naturelle

La déduction naturelle est un système de preuve pour la logique propositionnelle. Il est composé de règles d'inférence qui permettent de déduire de nouvelles formules à partir de formules existantes. Une preuve est un arbre dont les noeuds sont des formules et les feuilles sont des axiomes. Une preuve est correcte si elle respecte les règles d'inférence. Une preuve est complète si elle contient toutes les formules qui sont des conséquences logiques des axiomes.

2.1 Règles pour la conjonction

- Règle d'introduction :

$$\frac{\Phi \quad \Psi}{\Phi \wedge \Psi} \bigwedge i$$

- Règle d'élimination :

$$\frac{\Phi \wedge \Psi}{\Phi} \bigwedge e_1 \qquad \frac{\Phi \wedge \Psi}{\Psi} \bigwedge e_2$$

Exemple: La règle d'introduction se lit : si j'ai une preuve de Φ et une preuve de Ψ, alors j'ai une preuve de $\Phi \wedge \Psi$.

2.2 Règles pour la double négation

- Règle d'introduction :

$$\frac{\Phi}{\neg\neg\Phi} \neg\neg i$$

- Règle d'élimination :

$$\frac{\neg\neg\Phi}{\Phi} \neg\neg e$$

2.3 Elimination de l'implication : Modus Ponens

Règle d'élimination :

$$\frac{\Phi \quad \Phi \to \Psi}{\Psi} \to_{\text{MP}} (\text{ou } \to_e)$$

Exemple:

1. Φ : "Il pleut"
2. Ψ : "S'il pleut, "je prends mon parapluie"
3. Alors on en déduit que "je prends mon parapluie"

En contraposition, nous avons le Modus Tollens :

$$\frac{\Phi \to \Psi \quad \neg\Psi}{\neg\Phi} \to_{\text{MT}}$$

Exemple:

1. Φ : "s'il pleut, alors la route est mouillée"
2. Ψ : "la route n'est pas mouillée"
3. Alors on en déduit que "il ne pleut pas"

2.4 Règle pour l'introduction de l'implication

$$\dfrac{\begin{array}{l}\Phi\text{hyp.}\\ \ldots\\ \Psi\text{fin hyp.}\end{array}}{\Phi \to \Psi} \to_i$$

Lorsqu'on posera une hypoèse, on indentera l'hypothèse et toutes les lignes de la sous-preuve, jusqu'à la fermeture d'hypothèse.

Exemple: Ici, on va voir un exemple de ce qu'on a vu jusque maintenant :
On veut démontrer : $t \vdash (t \to p) \to (q \to (s \to p))$
Ici le prémisse est "t est vrai", le prémisse sera toujours ce qui se trouve à gauche de la déduction.

1.	t	prémisse.
2.	$t \to p$	hyp_1.
3.	q	hyp_2.
4.	$s \to p$	hyp_3.
5.	p	MP(1,2),fin hyp_3.
6.	$s \to p$	$\to_i$ (4,5),fin hyp_2.
7.	$q \to (s \to p)$	$\to_i$ (3,6),fin hyp_1.
8.	$(t \to p) \to (q \to (s \to p))$	$\to_i$ (2,7).

Nous pouvons également prouver des formules sans prémisse comme suit :

1.	p	hyp
2.	$\neg\neg p$	$\neg\neg_i(1)$, fin hyp.
3.	$p \to \neg\neg p$	$\to_i$ (1, 2).

On a établit que $\vdash p \to \neg\neg p$.

Remarque: Les formules Φ telles que : $\vdash \Phi$ sont appelées des théorèmes.

2.5 Règle pour l'ouverture et la fermeture d'hypothèses

- toute hypothèse introduite doit être fermée.
- on ne peut jamais fermer deux hypothèses en même temps.
- Une fois une hypothèse fermée, on ne peut pas utilsier les formules déduites entre l'ouverture et la fermeture de cette hypothèse.

2.6 Règle pour l'introduction de la disjonction

$$\dfrac{\Phi}{\Phi \vee \Psi} \vee_{i_2} \qquad \dfrac{\Psi}{\Phi \vee \Psi} \vee_{i_2}$$

2.7 Elimination de la disjonction

$$\dfrac{\begin{array}{cc}\Psi_1\text{hyp.} & \Psi_2\text{hyp.}\\ \dots & \dots\\ \Psi_1 \vee \Psi_2 & \Phi\text{fin hyp.} \quad \Phi\text{fin hyp.}\end{array}}{\Phi}\vee_e$$

Exemple: Supposons que les faits suivants soient vrais :

1. si ma note d'examen est excellente, j'irai boire un verre.
2. si ma note d'examen est bonne, j'irai boire un verre.
3. ma note sera excellente ou bonne.

Alors je peux en déduire que j'irai boire un verre.

Ici aussi, on ne peut pas utiliser l'hypothèse temporaire faite pour l'autre cas. (sauf si elle a été établie avant)

2.8 Règle de copie

$$\dfrac{\Phi}{\Phi}\text{copie}$$

2.9 Règle pour la négation

Les contradictions sont des formules de la forme :

$$\neg\Phi \wedge \Phi \quad \text{ou} \quad \neg\Phi \wedge \neg\Phi$$

Toutes les contradictions sont logiquement équivalentes à la formule $\bot$. (rappel : $\bot$ est une formule qui est toujours fausse)
Le fait que l'on peut tout déduire à partir d'une contradiction est formalisé par la règle suivante :

$$\dfrac{\bot}{\Phi}\bot_e$$

Le fait que $\bot$ représente une contradiction est formalisé par la règle suivante :

$$\dfrac{\Phi \quad \neg\Phi}{\bot}\neg_e$$

Afin d'introduire une négation, supposons que l'on fasse une hypothèse et que l'on arrive à déduire une contradiction, dans ce cas, l'hypothèse est fausse. Ceci est formaisé par la règle suivante :

$$\dfrac{\begin{array}{c}\Phi \quad \text{hyp.}\\ \dots\\ \bot\text{fin hyp.}\end{array}}{\neg\Phi}\neg_i$$

2.10 Règles pour l'équivalence

$$\dfrac{\Phi_1 \to \Phi_2 \quad \Phi_2 \to \Phi_1}{\Phi_1 \leftrightarrow \Phi_2}\leftrightarrow_i$$

$$\dfrac{\Phi_1 \leftrightarrow \Phi_2}{\Phi_1 \to \Phi_2}\leftrightarrow_{e_1} \qquad \dfrac{\Phi_1 \leftrightarrow \Phi_2}{\Phi_2 \to \Phi_1}\leftrightarrow_{e_2}$$

2.11 Règles dérivées

Il existe de nombreuses formules dérivées qui peuvent s'obtenir à partir des autres règles vues plus haut. (à voir si beaucoup utilisée au TP, si oui, ajouter : MT,RAA,LEM)

2.12 Théorèmes

Définition 2.2. adéquation

Pour tout $\Psi_1, ..., \Psi_n, \Psi$, si $\Psi_1, ..., \Psi_n \vdash \Psi$ alors $\Psi_1, ..., \Psi_n \vDash \Psi$.

Définition 2.3. complétude

Pour tout $\Psi_1, ..., \Psi_n, \Psi$, si $\Psi_1, ..., \Psi_n \vDash \Psi$ alors $\Psi_1, ..., \Psi_n \vdash \Psi$.

2.13 Démontrer une implication

Il existe deux méthodes pour démontrer une implication $(A \to B)$:

1. On suppose A et on en déduit B.
2. On suppose non B et on en déduit non A.

2.14 Démontrer une équivalence

Il existe deux méthodes pour démontrer une équivalence $(A \leftrightarrow B)$:

1. On suppose A et on en déduit B et réciproquement on suppose B et on en déduit A.
2. On prouve une chaîne d'équivalences.

2.15 Preuve par cas

Ce type de preuve repose sur une généralisation de la règle $\vee_e$: si on sait qu'on est soit dans le cas A_1 soit dans le cas A_n, et que pout tout $i \in \{1, ..., n\}$, on peut démontrer une propriété P, alors c'est que P est vraie.

2.16 Preuve par contradiction

On veut démontrer une propriété P. On suppose son contraire $\neg P$ et on en déduit une contradiction. On en déduit que P est vraie.

Remarque: Cette partie du cours nécessite de prendre le temps de bien comprendre les exemples donnés dans le cours ainsi que les exercices vus au TP.

3 Le Problème SAT

Remarque: Soit P un ensemble de propositions. Une formule Φ de la logique propositionnelle sur P est satisfaisable si il echiste une interprétation

$$V : X \to \{1, 0\}, \text{telle que } V \models \Phi$$

3.1 Littéraux et Clauses

Définition 3.1. Littéral

Un **littéral** est une variable x ou sa négation $\neg x$.

Définition 3.2. Clause

Une **clause** est une disjonction de littéraux $l_1 \vee l_2 \vee \cdots \vee l_n$. Elle est satisfaite par une valuation V s'il echiste i tel que $V(l_i) = 1$. Par extension, on appelle .

Exemple: p et $\neg p$ sont des littéraux. $x \vee \neg y \vee r$ est une clause mais pas $x \wedge y$.

Théorème 3.1. Satisfaction d'ensemble de clauses

Un ensemble de clauses $A = \{C_1, \ldots, C_n\}$ est satisfait par une valuation V, notée $V \models A$, si pour tout i, $V \models C_i$. En particulier, tout valuation satisfait l'ensemble vide $A = \emptyset$.

3.1.1 Lien avec les formes normales

- Une formule est en forme normale conjonctive (**FNC**) si et seulement si c'est une conjonction de disjonctions de littéraux, de la forme :

$$\bigwedge_i (\bigvee_j (\neg) x_{i,j})$$

- Une formule est en forme normale disjonctive (**FND**) si et seulement si c'est une disjonction de conjonctions de littéraux, de la forme :

$$\bigvee_i (\bigwedge_j (\neg) x_{i,j})$$

Lemme 3.1. Equivalence formule FNC Tout ensemble non-vide de clauses $A = \{C_1, \ldots, C_n\}$ est équivalent à la formule en FNC $\Phi_A = \bigwedge_{i=1}^{n} C_i$, au sens où pour toute valuation, $V \models A \Leftrightarrow V \models \Phi_A$.

Remarque: Pour mettre une formule sous forme de clauses, il suffit de la mettre en FNC.

3.1.2 Mise sous FNC et FND

Afin de parvenir à une FNC ou une FND, on utilise les transformations successives suivantes pour obtenir les formes normales :

1. élimination des connecteurs $\to$ et $\leftrightarrow$ grâce aux équivalences suivantes :

$$(\Phi \to \Psi) \equiv (\neg\Phi \vee \Psi)$$
$$(\Phi \leftrightarrow \Psi) \equiv (\neg\Psi \vee \Psi) \wedge (\Phi \vee \neg\Psi)$$

2. entrer les négations le plus à l'intérieur possible :

$$\neg(\Phi \wedge \Psi) \equiv (\neg\Phi \vee \neg\Psi) \qquad \neg\neg\Phi \equiv \Psi$$
$$\neg(\Psi \vee \Psi) \equiv (\neg\Phi \wedge \Psi)$$

3. utilisation des distributivité de $\vee$ et $\wedge$:

$$(\Phi \vee (\Psi \wedge \chi)) \equiv (\Phi \vee \Psi) \wedge (\Phi \vee \chi) \text{(mise sous FNC)}$$
$$(\Phi \wedge (\Psi \vee \chi)) \equiv (\Phi \wedge \Psi) \vee (\Phi \wedge \chi) \text{(mise sous FND)}$$

Exemple: Illustrons cela à travers un exemple, mettons la formule $\neg(x \leftrightarrow (y \rightarrow r))$

1. on retire les équivalences et implications :
$$\neg((\neg x \vee \neg y \vee r) \wedge (\neg(\neg y \vee r) \vee x))$$
2. on pousse les négations à l'intérieur :
$$(x \wedge y \wedge \neg r) \vee ((\neg y \vee r) \wedge \neg x)$$
3. on distribue :
$$(x \wedge y \wedge \neg r) \vee ((\neg y \vee r) \wedge \neg x)$$
4. et encore :
$$(x \vee \neg y \vee r) \wedge (y \vee \neg y \vee r) \wedge (\neg r \vee \neg y \vee r) \wedge (x \vee \neg x) \wedge (y \vee \neg x) \wedge (\neg r \vee \neg x)$$
5. on retire les formules équivalentes à $\top$:
$$(x \vee \neg y \vee r) \wedge (y \vee \neg x) \wedge (\neg r \vee \neg x)$$

Invitation à prendre le temps de faire cette démonstrations sur une feuille étape par étape pour voir si tout a bine été compris.

3.2 Problème SAT

Définition 3.3. Problème SAT

Une **entrée** est un ensemble de clauses S.
Une **sortie** est de la forme : Est-ce que S est **sat**isfaisable ?

Si S est satisfaisable, on aimerait que l'algorithme nous retourne une valuation V qui satisfait S.

Remarque: Actuellement, on ne sait toujours pas s'il existe un algorithme pour ce problème dont la complexité en temps est polynomiale. L'intuition veut que ce ne soit pas le cas mais aucune preuve n'a pu être fournie jusqu'à présent. Si on parvenait à le prouver, on pourrait gagner un prix d'un million de dollars ;). C'est un problème qui appartient à la classe NP. (cf. chapitre sur la complexité)

3.3 Introduction aux solveurs SAT

Définition 3.4. Solveur SAT

Un solveur est un programme qui décidé le problème SAT. Si la formule est satisfaisable, une interprétation qui la satifsait est retournée. Ils ont une complexité au pire cas exponentielle.

3.3.1 Notations pour les grandes disjonctions et conjonctions

Définition 3.5. disjonction et conjonction

- On appelera *disjonction* une formule de la forme $\Phi_1 \vee \Phi_2 \vee \cdots \vee \Phi_n$ où $n \geq 1$. Attention, lorsque $n = 1$, la disjonction est réduite à la seule forme Φ_1. On utilisera l'abbréviation suivante :
$$\bigvee_{i=1}^{n} \Phi_i$$

- On appelera *conjonction* une formule de la forme $\Phi_1 \wedge \Phi_2 \wedge \cdots \wedge \Phi_n$ où $n \geq 1$. Attention, lorsque $n = 1$, la conjonction est réduite à la seule forme Φ_1. On utilisera l'abbréviation suivante :
$$\bigwedge_{i=1}^{n} \Phi_i$$

3.4 Modélisation

Dans cette section, nous feront la modélisation du problème des 8 reines, d'autres exemples sont présents dans le cours mais le raisonnement demeure identique.

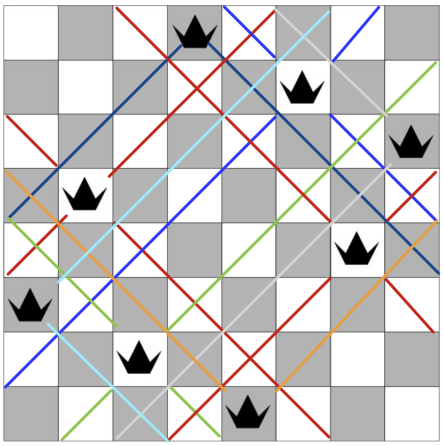

FIGURE 3.1 – Problème des 8 reines

3.4.1 Choix des variables

$$X = \{x_{i,j} | i, j \in \{1, \ldots, 8\}\}$$

La sémantique est la suivante : "$x_{i,j}$ est vraie si et seulement s'il y a une reine en case (i, j)".

3.4.2 Expression des contraintes

1. Pas d'attaque horizontale
2. Pas d'attaque verticale
3. Pas d'attaque en diagonale
4. Au moins une reine par ligne

Pas d'attaque horizontale

"Pour toute ligne, il n'existe pas deux reines sur cette ligne."

$$\bigwedge_{i \in \{1,\ldots,8\}} \neg \Big(\bigvee_{j,j' \in \{1,\ldots,8\} | j \neq j'} x_{i,j} \wedge x_{i,k} \Big)$$
$$\equiv \bigwedge_{i,j,j' \in \{1,\ldots,8\} | j \neq j'} (\neg x_{i,j} \vee \neg x_{i,j'})$$

Pas d'attaque verticale

"Pour toute colonne, il n'existe pas deux reines sur cette colonne."

$$\bigwedge_{j \in \{1,\ldots,8\}} \neg \Big(\bigvee_{i,i' \in \{1,\ldots,8\} | i \neq i'} x_{i,j} \wedge x_{k,j} \Big)$$

$$\equiv \bigwedge_{i,i',j\in\{1,\dots,8\}|i\neq i'} (\neg x_{i,j} \vee \neg x_{i',j})$$

Pas d'attaque en diagonale

"Pour toute diagonale, il n'existe pas deux reines sur cette diagonale."

$$\bigwedge_{i,i',j,j'\in\{1,\dots,8\}|i\neq i'\wedge j\neq j'} \neg(x_{i,j} \wedge x_{i',j'} \wedge |i-i'| = |j-j'|)$$
$$\equiv \bigwedge_{i,i',j,j'\in\{1,\dots,8\}|i\neq i'\wedge j\neq j'} (\neg x_{i,j} \vee \neg x_{i',j'} \vee |i-i'| \neq |j-j'|)$$

Au moins une reine par ligne

"Pour toute ligne, il existe au moins une reine sur cette ligne."

$$\bigwedge_{i\in\{1,\dots,8\}} (\bigvee_{j\in\{1,\dots,8\}} x_{i,j})$$

3.5 Algorithme DPLL

Ne sera pas à l'examen.

3.6 Transformation de Tseitin

Il se peut que le problème ne s'exprime pas facilement par une formule en FNC. Le but de la transformation de Tseitin est d'ajouter de nouvelles variables et des équivalences.

Exemple: Considérons $\Phi = (x \wedge q) \vee \neg(y \vee r)$, Dans la transformation de Tseitin, on va remplacer $x \wedge q$ par une nouvelle variable x_1 et $\neg(y \vee r)$ par une nouvelle variable x_2. Voici la formule considérée :

$$(x_1 \vee x_2) \wedge (x_1 \leftrightarrow x \wedge q) \wedge (x_2 \leftrightarrow \neg(y \vee r))$$

Il reste encore à mattre les deux formules en FNC : (cf. 3.1.2)

- $x_1 \leftrightarrow x \wedge q$
$$\equiv (x_1 \rightarrow (x \wedge q)) \wedge ((x \wedge q) \rightarrow x_1)$$
$$\equiv (\neg x_1 \vee (x \wedge q)) \wedge (\neg(x \wedge q) \vee x_1)$$
$$\equiv (\neg x_1 \vee x) \wedge (\neg x_1 \vee q) \wedge (\neg x \vee \neg q \vee x_1)$$
- $x_2 \leftrightarrow \neg(y \vee r)$
$$\equiv (x_2 \rightarrow \neg(y \vee r)) \wedge (\neg(y \vee r) \rightarrow x_2)$$
$$\equiv (\neg x_2 \vee \neg y \wedge \neg r) \wedge (y \vee r \vee x_2)$$
$$\equiv (\neg x_2 \vee \neg y) \wedge (\neg x_2 \vee \neg r) \wedge (y \vee r \vee x_2)$$

Dès lors, nous pouvons écrire la formule sous FNC suivante :

$\Psi = (x_1 \vee x_2) \wedge (\neg x_1 \vee p) \wedge (\neg x_1 \vee q) \wedge (\neg x \vee \neg y \vee x_1) \wedge (\neg x_2 \vee \neg q) \wedge (\neg x_2 \vee \neg r) \wedge (y \vee r \vee x_2)$

La transformation de Tseitin est intéressante lorsqu'on devra mettre sous FNC des formules qui sont sous forme normale disjonctive $C_1 \vee C_2 \vee \dots \vee C_n$. Car il suffit d'introduire une variable x_i pour chaque C_i et on obtient la formule :

$$(x_1 \vee x_2 \vee \dots \vee x_n) \wedge (x_1 \leftrightarrow C_1) \wedge \dots \wedge (x_n \leftrightarrow C_n)$$

Si on suppose que $C_i = l_1 \wedge \dots l_k$ où l_i sont des littéraux, alors mettre $x_i \leftrightarrow C_i$ sous FNC est assez simple :

$$x_i \leftrightarrow C_i \equiv (\bigwedge_{j=1}^{k} (\neg x_i \vee l_j)) \wedge (x_i \vee \bigvee_{j=1}^{k} (\neg l_j))$$

Il ne faut surtout pas hésiter à ajouter des variables pour minimiser le nombre de clauses.