1 Motivation

1.1 Motivation de l'étude de la logique en générale

Répondre à des questions :
- Qu'est ce qu'un énoncé mathématique ?
- Qu'est ce qu'une preuve ?
- Peut on écrire des énoncés mathématiques dont les objets sont eux-mêmes des objets mathématiques ?
- Peut on raisonner automatiquement sur les énoncés mathématiques ?

1.2 Motivation du fragment propositionnelle de la logique

Pour illustrer l'intérêt de la logique propositionnelle (celle qui fait l'objet de ce chapitre), considérons le sudoku 4×4 de la Figure 1.

3			2
	4	1	
	3	2	
4			1

FIGURE 1 – sudoku 4×4

Les règles devant être satisfaites par une solution de ce sudoku sont assez simples et peuvent être exprimées à l'aide des connecteurs logiques usuels : $\wedge, \vee, \rightarrow, \neg$. En effet si l'on dénote une solution du sudoku par une matrice m de taille 4×4 à valeurs dans l'ensemble $\{1, 2, 3, 4\}$, nous pouvons exprimer le fait que la première ligne de la solution doit contenir une et une seule fois la valeur 3 par la formule $L_{1,3}$ ci-dessous (le 1 dénote le numéro de ligne et le 3 dénote la valeur considérée) :

$$\begin{aligned} L_{1,3} = \quad & ((m(1,1)=3) \vee (m(1,2)=3) \vee (m(1,3)=3) \vee (m(1,4)=3)) \\ \wedge \quad & ((m(1,1)=3) \rightarrow \neg((m(1,2)=3) \vee (m(1,3)=3) \vee (m(1,4)=3))) \\ \wedge \quad & ((m(1,2)=3) \rightarrow \neg((m(1,1)=3) \vee (m(1,3)=3) \vee (m(1,4)=3))) \\ \wedge \quad & ((m(1,3)=3) \rightarrow \neg((m(1,1)=3) \vee (m(1,2)=3) \vee (m(1,4)=3))) \\ \wedge \quad & ((m(1,4)=3) \rightarrow \neg((m(1,1)=3) \vee (m(1,2)=3) \vee (m(1,3)=3))) \end{aligned} \tag{1}$$

où $m(i, j)$ désigne la valeur située sur la i-ième ligne à la j-ième colonne.
On procède de même pour toutes les lignes et les colonnes du sudoku. De façon générale, la formule $L_{i,n}$ exprime que la ligne i contient une et une seule fois la valeur n. De manière similaire, on écrit les formules $C_{i,n}$ et $Q_{i,n}$ qui expriment respectivement que chaque colonne et chaque quadrant contient une et une seule fois une valeur de $\{1, 2, 3, 4\}$. La conjonction de ces formules décrit exactement les contraintes que doit satisfaire une matrice m pour être solution d'un sudoku. Pour décrire un jeu particulier, il suffit donc d'y rajouter les contraintes initiales. Sur le sudoku de la Figure 1, on ajoute par exemple :

$$\begin{aligned} \mathsf{Ini} = \quad & (m(1,1)=3) \wedge (m(2,2)=4) \wedge (m(1,4)=2) \wedge (m(2,3)=1) \\ \wedge \quad & (m(3,2)=3) \wedge (m(4,1)=4) \wedge (m(3,3)=2) \wedge (m(4,4)=1) \end{aligned}$$

La formule obtenue par conjonction de toutes les formules de ligne, colonne, quadrant et condition initiale est écrite dans un certain langage logique et le sudoku de la Figure 1 admet une solution si et seulement si cette formule est satisfiable.

On peut faire un pas de plus dans "l'appauvrissement" du langage nécessaire pour exprimer un problème de sudoku. En effet, remarquons que chaque formule atomique de la forme $(m(i,j) = k)$ peut prendre la valeur de vérité vrai ou faux. Abstrayons donc cette formule atomique derrière une variable $p_{i,j,k}$, prenant ses valeurs dans le monde des booléens. Cette variable sera interprétée par la valeur booléenne vrai ou faux : si elle est interprétée par vrai alors $m(i,j)$ vaut k, sinon $m(i,j)$ ne vaut pas k. La formule $L_{1,3}$ est donc transformée en :

$$\begin{array}{rl} L_{1,3} = & (p_{1,1,3} \vee p_{1,2,3} \vee p_{1,3,3} \vee p_{1,4,3}) \\ \wedge & (p_{1,1,3} \rightarrow \neg(p_{1,2,3} \vee p_{1,3,3} \vee p_{1,4,3})) \wedge (p_{1,2,3} \rightarrow \neg(p_{1,1,3} \vee p_{1,3,3} \vee p_{1,4,3})) \\ \wedge & (p_{1,3,3} \rightarrow \neg(p_{1,1,3} \vee p_{1,2,3} \vee p_{1,4,3})) \wedge (p_{1,4,3} \rightarrow \neg(p_{1,1,3} \vee p_{1,2,3} \vee p_{1,3,3})) \end{array} \quad (2)$$

Notons que la disparition du prédicat $=$ et des symboles de constante 1, 2, 3 et 4 pour les valeurs des cases nous oblige à ajouter, pour chaque case (i,j), une formule $c_{i,j}$ pour assurer que la case (i,j) contient une et une seule valeur. Par exemple pour la case $(2,3)$, on ajoute :

$$\begin{array}{rl} c_{2,3} = & (p_{2,3,1} \vee p_{2,3,2} \vee p_{2,3,3} \vee p_{2,3,4}) \\ \wedge & (p_{2,3,1} \rightarrow \neg(p_{2,3,2} \vee p_{2,3,3} \vee p_{2,3,4})) \wedge (p_{2,3,2} \rightarrow \neg(p_{2,3,1} \vee p_{2,3,3} \vee p_{2,3,4})) \\ \wedge & (p_{2,3,3} \rightarrow \neg(p_{2,3,1} \vee p_{2,3,2} \vee p_{2,3,4})) \wedge (p_{2,3,4} \rightarrow \neg(p_{2,3,1} \vee p_{2,3,2} \vee p_{2,3,3})) \end{array}$$

La première ligne exprime que la case contient une des valeurs 1, 2, 3 ou 4, les autres, que cette valeur est unique.
La formule H obtenue par la conjonction de toutes les formules lignes, colonnes, quadrant et condition initiale transformées, ainsi que toutes les formules $c_{i,j}$, ne contient plus que des connecteurs propositionnels et des symboles de variables booléennes : c'est ce qu'on appelle la logique propositionnelle. Remarquons que le problème de la résolution du sudoku 4×4 ci-dessus revient alors au problème de trouver une affectation des valeurs des variables booléennes, qui rende la formule "vraie".

2 Syntaxe

Dans cette première section, on se donne la syntaxe d'un ensemble de termes que nous appellerons la "logique propositionnelle". Le fait de qualifier la logique de propositionnelle, sous-entend qu'il existe d'autres logiques, nous en verrons une autre en deuxième année, et sûrement d'autres en DS, en exercices, ….

2.1 Syntaxe de la logique propositionnelle

Définition 1 (Ensemble des variables propositionnelles)**.** Nous supposons donné un ensemble $\mathscr{P}$ et appelons *variables propositionnelles* les éléments de cet ensemble.

Exemple 2. Généralement $\mathscr{P} \supseteq \{p, q, r, \dots\}$

Définition 3 (Formule de la logique propositionnelle)**.** Étant donné un ensemble $\mathscr{P}$ de variables propositionnelles, on définit l'ensemble $\mathscr{F}$ par l'induction structurelle suivante :

- $\bot \in \mathscr{F}$
- $\top \in \mathscr{F}$
- Pour tout $p \in \mathscr{P}$, $p \in \mathscr{F}$
- Si $G \in \mathscr{F}$, alors $\neg G \in \mathscr{F}$
- Si $(G,H) \in \mathscr{F}^2$ alors $G \wedge H \in \mathscr{F}$
- Si $(G,H) \in \mathscr{F}^2$ alors $G \vee H \in \mathscr{F}$
- Si $(G,H) \in \mathscr{F}^2$ alors $G \rightarrow H \in \mathscr{F}$
- Si $(G,H) \in \mathscr{F}^2$ alors $G \leftrightarrow H \in \mathscr{F}$

Les opérateurs binaires $\wedge$, $\vee$, $\rightarrow$, $\leftrightarrow$ seront utilisés en notation infixe.

Remarque 4. Comme tout ensemble défini par induction structurel, il est possible d'écrire l'arbre de dérivation d'une formule de la logique propositionnelle. On appelle cet arbre *l'arbre de syntaxe abstrait* d'une formule

Remarque 5. On n'oubliera pas les parenthèses permettant de préciser l'arbre de la formule.

Exemple 6. Si $\mathscr{P} = \{a, b, c\}$, alors $\top$, $(a \wedge b) \to c$ et $(\neg a) \wedge b$ sont des formules de $\mathscr{F}$.

Définition 7 (Taille d'une formule)**.** On définit par induction structurelle une fonction $\mathsf{taille} : \mathscr{F} \to \mathbb{N}$.

$$\begin{array}{rcll}
\mathsf{taille}(\top) & = & 1 & \\
\mathsf{taille}(\bot) & = & 1 & \\
\mathsf{taille}(p) & = & 1 & \text{si } p \in \mathscr{P} \\
\mathsf{taille}(\neg H) & = & 1 + \mathsf{taille}(H) & \text{avec } H \in \mathscr{F} \\
\mathsf{taille}(H_1 \vee H_2) & = & 1 + \mathsf{taille}(H_1) + \mathsf{taille}(H_2) & \text{avec } H_1, H_2 \in \mathscr{F} \\
\mathsf{taille}(H_1 \wedge H_2) & = & 1 + \mathsf{taille}(H_1) + \mathsf{taille}(H_2) & \text{avec } H_1, H_2 \in \mathscr{F} \\
\mathsf{taille}(H_1 \to H_2) & = & 1 + \mathsf{taille}(H_1) + \mathsf{taille}(H_2) & \text{avec } H_1, H_2 \in \mathscr{F} \\
\mathsf{taille}(H_1 \leftrightarrow H_2) & = & 1 + \mathsf{taille}(H_1) + \mathsf{taille}(H_2) & \text{avec } H_1, H_2 \in \mathscr{F}
\end{array}$$

Définition 8 (Ensemble des variables propositionnelles d'une formule)**.** On définit par induction structurelle une fonction $\mathsf{var} : \mathscr{F} \to \wp(\mathscr{P})$.

$$\begin{array}{rcll}
\mathsf{var}(\top) & = & \emptyset & \\
\mathsf{var}(\bot) & = & \emptyset & \\
\mathsf{var}(p) & = & \{p\} & \text{si } p \in \mathscr{P} \\
\mathsf{var}(\neg H) & = & \mathsf{var}(H) & \text{avec } H \in \mathscr{F} \\
\mathsf{var}(H_1 \vee H_2) & = & \mathsf{var}(H_1) \cup \mathsf{var}(H_2) & \text{avec } H_1, H_2 \in \mathscr{F} \\
\mathsf{var}(H_1 \wedge H_2) & = & \mathsf{var}(H_1) \cup \mathsf{var}(H_2) & \text{avec } H_1, H_2 \in \mathscr{F} \\
\mathsf{var}(H_1 \to H_2) & = & \mathsf{var}(H_1) \cup \mathsf{var}(H_2) & \text{avec } H_1, H_2 \in \mathscr{F} \\
\mathsf{var}(H_1 \leftrightarrow H_2) & = & \mathsf{var}(H_1) \cup \mathsf{var}(H_2) & \text{avec } H_1, H_2 \in \mathscr{F}
\end{array}$$

Définition 9 (Littéral)**.** Si $\mathscr{P}$ est un ensemble de variables propositionnelles, on note $\mathscr{L}(\mathscr{P})$ l'ensemble des *littéraux*, définis comme $\{\neg p \mid p \in \mathscr{P}\} \cup \mathscr{P}$. On dira d'un littéral p qu'il est un *littéral positif*, et d'un littéral $\neg p$ que c'est un *littéral négatif*.

Définition 10 (Sous-formules)**.** On appelle *sous-formule* d'une formule H une formule G telle que $G \preceq H$ où $\preceq$ est la relation d'ordre bien fondée induite par la définition inductive des formules.

Exemple 11. Les formules a, b, c, $a \wedge b$ et $(a \wedge b) \to c$ sont les sous-formules de la formule $(a \wedge b) \to c$.

Remarque 12. La syntaxe des formules de la logique propositionnelle étant définie par une induction structurelle, *il n'est pas vrai de dire* que $(p \wedge q) \wedge r = p \wedge (q \wedge r)$, de même on ne peut avancer que $\top \wedge \top = \top$. En effet les arbres de syntaxe des formules $(p \wedge q) \wedge r$ et $p \wedge (q \wedge r)$ sont représentés ci-dessous et ce ne sont pas les mêmes arbres.

Remarque 13. Dans la suite, lorsque Γ est un ensemble fini de formules admettant une numérotation canonique $\{G_1, G_2, \ldots G_n\}$, $\bigwedge_{G \in \Gamma}$ est la formule $\top$ si $n = 0$ et la formule $G_1 \wedge (G_2 \wedge (\ldots \wedge G_n))$ sinon. Remarquer que l'hypothèse de numérotation canonique est nécessaire dans la définition précédente, puisque pour l'ensemble $\Gamma = \{G, H, I\}$, on ne saurait si $\bigwedge_{G \in \Gamma} G$ représente la formule $G \wedge (H \wedge I)$ ou $(G \wedge H) \wedge I$ ou ….

Homework 14.

1. Donner l'arbre de syntaxe abstrait de la formule $(\neg(a \leftrightarrow (\neg b))) \to (b \vee \top)$, donner l'ensemble de ses sous-formules.
2. Donner une formule G de la logique propositionnelle telle que $\mathsf{taille}(G) = 6$ et $\mathsf{var}(G) = \{a, b\}$.
3. Montrer que $\forall H \in \mathcal{F}, \mathsf{taille}(H) \geqslant |\mathsf{var}(H)|$.
4. Justifier que $\{\frac{\mathsf{taille}(H)}{1+|\mathsf{var}(H)|} \mid H \in \mathcal{F}\}$ n'est pas borné.
5. Définir inductivement une fonction $\mathsf{sf} : \mathcal{F} \to \wp(\mathcal{F})$ associant à une formule G de $\mathcal{F}$ l'ensemble des sous-formules de G. Établir et démontrer une relation pertinente entre $|\mathsf{vars}(G)|$ et $\mathsf{taille}(G)$, valide pour tout $G \in \mathcal{F}$.
6. À l'image de ce qui est fait dans la remarque 13, proposer une définition de $\bigvee_{G \in \Gamma} G$

3 Sémantique

La syntaxe définie dans la section précédente ne vient pas avec un sens prédéfinie : le symbole $\vee$ n'est pas encore attaché au sens du "ou" de la langue française. Ainsi dans cette partie, on munit la syntaxe d'une sémantique.

3.1 Algèbre de Boole

On se munit dans un premier temps d'un monde à deux valeurs : les booléens.

Définition 15 (Ensemble des booléens)**.** Dans toute la suite on note $\mathbb{B} = \{\mathsf{V}, \mathsf{F}\}$ un ensemble à deux éléments nommées *booléens*

Définition 16 $(+, \cdot, \overline{\square})$**.** Sur l'espace $\mathbb{B}$, on définit les trois opérateurs suivants :

a	b	$a \cdot b$	$a + b$
F	F	F	F
F	V	F	V
V	F	F	V
V	V	V	V

et

a	$\overline{a}$
F	V
V	F

Remarque 17. Nous avons les égalités suivantes dans $\mathbb{B}$.

commutativité			
$a \cdot b = b \cdot a$	(E2.1)	$a + b = b + a$	(E3.1)
élément neutre			
$\mathsf{V} \cdot a = a$	(E2.2)	$\mathsf{F} + a = a$	(E3.2)
élément absorbant			
$\mathsf{F} \cdot a = \mathsf{F}$	(E2.3)	$\mathsf{V} + a = \mathsf{V}$	(E3.3)
associativité			
$(a \cdot b) \cdot c = a \cdot (b \cdot c)$	(E2.4)	$(a + b) + c = a + (b + c)$	(E3.4)
idempotence			
$a \cdot a = a$	(E2.5)	$a + a = a$	(E3.5)
distributivité			
$a \cdot (b + c) = a \cdot b + a \cdot c$	(E4.1)	$a + (b \cdot c) = (a + b) \cdot (a + c)$	(E4.2)
complément			
$a \cdot \overline{a} = \mathsf{F}$	(E1.3)	$a + \overline{a} = \mathsf{V}$	(E1.4)
lois de De Morgan			
$\overline{a \cdot b} = \overline{a} + \overline{b}$	(E4.3)	$\overline{a + b} = \overline{a} \cdot \overline{b}$	(E4.4)

On notera spécifiquement la distributivité des deux opérateurs, l'un sur l'autre.

Homework 18.

1. Démontrer (E4.1), (E4.2), (E4.3), et (E4.4).
2. Par un raisonnement purement équationnel (en utilisant uniquement les équations de la Remarque 17), démontrer les égalités suivantes, pour tout $(x, y, z) \in \mathbb{B}^3$. On précisera laquelle des équations du tableau ci-dessus est utilisée entre chaque ligne de calcul.

   - $\overline{(\overline{x} + y)} + z = (\overline{\overline{x}} + z).(\overline{y} + z)$
   - $(\overline{x} + y) \cdot \overline{\overline{\overline{y}} + \overline{x}} = \mathsf{F}$
   - $\overline{x \cdot \overline{y}} + \overline{\overline{x} + y} = \mathsf{V}$
   - $\overline{(x + \overline{y}) \cdot y} + x = \mathsf{V}$
   - $\overline{x + y} + (x \cdot y) = (\overline{x} + y) \cdot (\overline{y} + x)$
   - $(x + y) \cdot (\overline{x} + y) = y$

3.2 Fonctions booléennes

Les formules de la logique propositionnelle ne sont pas seulement constituées de $\top$ et $\bot$ connectés entre eux par des connecteurs logiques. En effet on y trouve aussi des variables propositionnelles. Une formule de la logique propositionnelle ne peut donc recevoir une sémantique booléenne V ou F. La "valeur" de la formule dépend des "valeurs" des variables propositionnelles qui la constituent : $(a \wedge b)$ n'a de sens booléen qu'une fois qu'un sens est attaché aux variables a et b.

Définition 19 (Environnement propositionnel)**.** Étant donné un ensemble de variables propositionnelles $\mathscr{P}$, un *environnement propositionnel* $\mu \in \mathbb{B}^{\mathscr{P}}$ est une fonction de $\mathscr{P}$ dans $\mathbb{B}$.
Lorsque $\mathscr{P} = \{p_1, p_2, \ldots, p_n\}$ est un ensemble fini, on adopte la notation $(p_1 \mapsto b_1, p_2 \mapsto b_2, \ldots, p_n \mapsto b_n)$ pour représenter l'environnement propositionnel ci-dessous.

$$\mu : \left\{ \begin{array}{ll} \mathscr{P} & \to \mathbb{B} \\ p_i & \mapsto b_i \end{array} \right.$$

Exemple 20. $(p \mapsto \mathsf{V}, q \mapsto \mathsf{V})$ est un environnement propositionnel μ sur $\{p, q\}$ tel que $\mu(p) = \mathsf{V}$ et $\mu(q) = \mathsf{V}$. $()$ est l'unique environnement propositionnel sur $\emptyset$.

Remarque 21. Sur un environnement propositionnel $\mathscr{P}$ fini de cardinal n, il y a 2^n environnements propositionnels.

Définition 22 (Fonctions booléennes)**.** Étant donné un ensemble de variables propositionnelles $\mathscr{P}$, l'ensemble des fonctions booléennes sur $\mathscr{P}$, noté $\mathscr{F}(\mathscr{P})$, est l'ensemble des fonctions $f : \mathbb{B}^{\mathscr{P}} \to \mathbb{B}$.

Remarque 23. Si il n'y a pas ambiguïté (comprendre par là : lorsque l'ensemble $\mathscr{P}$ admet une numérotation canonique) on pourra voir une fonction booléenne comme une fonction de $\mathbb{B}^n \to \mathbb{B}$.

Exemple 24. Soit $\mathscr{P} = \{a, b\}$. La fonction f définie par :

$$\begin{array}{lcl} f((a \mapsto \mathsf{F}, b \mapsto \mathsf{F})) & = & \mathsf{F} \\ f((a \mapsto \mathsf{F}, b \mapsto \mathsf{V})) & = & \mathsf{V} \\ f((a \mapsto \mathsf{V}, b \mapsto \mathsf{F})) & = & \mathsf{V} \\ f((a \mapsto \mathsf{V}, b \mapsto \mathsf{V})) & = & \mathsf{V} \end{array}$$

est une fonction booléenne de $\mathscr{F}(\mathscr{P})$. En numérotant la variable a par 1 et la variable b par 2, on peut voir un environnement propositionnel $(a \mapsto \mathsf{F}, b \mapsto \mathsf{V})$ comme le vecteur $(\mathsf{F}, \mathsf{V}) \in \mathbb{B}^2$.

3.3 Interprétation d'une formule comme une fonction booléenne.

Nous avons maintenant tous les outils pour *donner un sens*, pour *évaluer*, pour *interpréter* les formules de la logique propositionnelle.

Définition 25 (Interprétation d'une formule dans un environnement)**.** Étant donnés un ensemble de variables $\mathcal{P}$ et une formule de $\mathcal{F}$ on définit inductivement l'interprétation d'une formule H dans un environnement booléen μ, noté $[\![H]\!]^\mu$ par :

$$\begin{array}{rcll}
[\![\top]\!]^\mu & = & \mathsf{V} & \\
[\![\bot]\!]^\mu & = & \mathsf{F} & \\
[\![p]\!]^\mu & = & \mu(p) & \text{si } p \in \mathcal{P} \\
[\![\neg H]\!]^\mu & = & \overline{[\![H]\!]^\mu} & \text{avec } H \in \mathcal{F} \\
[\![H_1 \vee H_2]\!]^\mu & = & [\![H_1]\!]^\mu + [\![H_2]\!]^\mu & \text{avec } H_1, H_2 \in \mathcal{F} \\
[\![H_1 \wedge H_2]\!]^\mu & = & [\![H_1]\!]^\mu \cdot [\![H_2]\!]^\mu & \text{avec } H_1, H_2 \in \mathcal{F} \\
[\![H_1 \to H_2]\!]^\mu & = & \overline{[\![H_1]\!]^\mu} + [\![H_2]\!]^\mu & \text{avec } H_1, H_2 \in \mathcal{F} \\
[\![H_1 \leftrightarrow H_2]\!]^\mu & = & \overline{[\![H_1]\!]^\mu} + [\![H_2]\!]^\mu & \text{avec } H_1, H_2 \in \mathcal{F}
\end{array}$$

Définition 26 (Fonction booléenne associée à une formule)**.** À une formule $H \in \mathcal{F}$ de la logique propositionnelle on associe une fonction booléenne $[\![H]\!]$ de la manière suivante :

$$\left\{ \begin{array}{ll} \mu & \mapsto [\![H]\!]^\mu \\ \mathbb{B}^{\mathcal{P}} & \to \mathbb{B} \end{array} \right.$$

Lemme 27 (Indépendance en les variables n'apparaissant pas)**.** *Soit H une formule de la logique propositionnelle. Soit μ et ρ deux environnements propositionnels tels que $\mu_{|\mathsf{vars}(H)} = \rho_{|\mathsf{vars}(H)}$ alors $[\![H]\!]^\mu = [\![H]\!]^\rho$.*

Homework 28.

1. On se donne un environnement $\mu = (a \mapsto \mathsf{V}, b \mapsto \mathsf{F}, c \mapsto \mathsf{F}, d \mapsto \mathsf{V})$, donner les valeurs de :
   - $[\![\top \wedge (a \to b)]\!]^\mu$
   - $[\![(c \leftrightarrow b) \wedge (a \leftrightarrow d)]\!]^\mu$
   - $[\![(a \vee c) \to (d \wedge b)]\!]^\mu$
2. Donner les fonctions $[\![a \to \bot]\!]$, $[\![a \vee b]\!]$ et $[\![a \vee (b \leftrightarrow c)]\!]$.
3. En tant que fonction d'un ensemble $\mathbb{B}^{\mathcal{P}}$ dans $\mathbb{B}$, $[\![H]\!]$ peut être vu comme le sous-ensemble de $\mathbb{B}^{\mathcal{P}}$ des environnements μ tels que $[\![H]\!]^\mu = \mathsf{V}$. Proposer, au moyen de notions ensemblistes, une telle définition inductive de $[\![H]\!]$.
4. Démontrer le lemme 27.

3.4 Liens sémantiques entre formules

Définition 29 (Formules équivalentes)**.** Deux formules H et G sont dites *équivalentes*, ce que l'on note $H \equiv G$, dès lors que $[\![H]\!] = [\![G]\!]$.

Propriété 30. *On a en fait que si $H \equiv G$ alors :*

- $\neg H \equiv \neg G$
- $H \wedge K \equiv G \wedge K$
- $\ldots$

On dit parfois d'une telle relation d'équivalence que c'est une congruence.

Lemme 31. *Soit H une formule telle que $\mathsf{vars}(H) = \emptyset$, alors $H \equiv \top$ ou $H \equiv \bot$.*

Démonstration. Laissée en exercice. □

Définition 32 (Conséquence sémantique)**.** On dit qu'une formule $G \in \mathcal{F}$ est conséquence sémantique d'une autre formule $H \in \mathcal{F}$ dès lors que :

$$\forall \mu \in \mathbb{B}^{\mathcal{P}}, (\llbracket H \rrbracket^{\mu} = \mathsf{V}) \Rightarrow (\llbracket G \rrbracket^{\mu} = \mathsf{V}).$$

On note alors $H \vDash G$. On étend cette notion de conséquence sémantique à des ensembles de formules. Si Γ est un ensemble de formules, on dénote par $\Gamma \vDash G$ le fait que

$$\forall \mu \in \mathbb{B}^{\mathcal{P}}, (\forall H \in \Gamma, \llbracket H \rrbracket^{\mu} = \mathsf{V}) \Rightarrow (\llbracket G \rrbracket^{\mu} = \mathsf{V}).$$

Remarque 33. $\vDash$ n'est pas une relation d'ordre sur $\mathcal{F}$, proposer un contre exemple.

Propriété 34 (Formules équivalentes)**.** *Pour toutes formules H et G de la logique propositionnelle, $H \equiv G$ si et seulement si $H \vDash G$ et $G \vDash H$.*

Démonstration. $\Rightarrow$ Soit H et G deux formules telles que $H \equiv G$, ainsi $\llbracket H \rrbracket = \llbracket G \rrbracket$, soit $\forall \mu \in \mathbb{B}^{\mathcal{P}}, \llbracket H \rrbracket^{\mu} = \llbracket G \rrbracket^{\mu}$. Montrons alors que $H \vDash G$. Soit $\mu \in \mathbb{B}^{\mathcal{P}}$ tel que $\llbracket H \rrbracket^{\mu} = \mathsf{V}$, alors $\llbracket G \rrbracket^{\mu} = \llbracket H \rrbracket^{\mu} = \mathsf{V}$. De même on montre que $G \vDash H$.

$\Leftarrow$ Soit H et G deux formules telles que $H \vDash G$ et $G \vDash H$. Soit $\mu \in \mathbb{B}^{\mathcal{P}}$ un environnement propositionnel.

- Si $\llbracket G \rrbracket^{\mu} = \mathsf{V}$ alors de $G \vDash H$, $\llbracket H \rrbracket^{\mu} = \mathsf{V}$
- Sinon si $\llbracket G \rrbracket^{\mu} = \mathsf{F}$ alors de $H \vDash G$, $\llbracket H \rrbracket^{\mu} = \mathsf{F}$ (sinon on aurait $\llbracket G \rrbracket^{\mu} = \mathsf{V}$).

□

Définition 35 (Formules valides, modèles, satisfiabilité)**.** On dit d'une formule $H \in \mathcal{F}$ et d'un environnement propositionnel μ que :

- H est *valide*, ou est une *tautologie*, lorsque $\forall \mu \in \mathbb{B}^{\mathcal{P}}, \llbracket H \rrbracket^{\mu} = \mathsf{V}$.
- H est *satisfiable*, lorsque $\exists \mu \in \mathbb{B}^{\mathcal{P}}, \llbracket H \rrbracket^{\mu} = \mathsf{V}$.
- H est *insatisfiable* ou est dite *antilogie*, lorsque H n'est pas satisfiable.
- μ est *un modèle* de H lorsque $\llbracket H \rrbracket^{\mu} = \mathsf{V}$.

Exemple 36. Les formules $\top$, $p \vee \neg p$, $(\neg p) \leftrightarrow p$, $(p \wedge q) \leftrightarrow (q \wedge p)$ sont valides. Les formules $\bot$, $p \wedge \neg p$, sont insatisfiables. La formule $(((p \rightarrow q) \rightarrow q) \rightarrow p)$ n'est pas valide, puisque dans l'environnement $(p \mapsto \mathsf{F}, q \mapsto \mathsf{F})$, la formule s'évalue à F, toutefois elle est satisfiable, parce qu'elle s'évalue à V dans $(p \mapsto \mathsf{V}, q \mapsto \mathsf{V})$.

Remarque 37. $\emptyset \vDash H$ si et seulement si H est valide. On notera alors parfois $\vDash H$ le fait que H est une formule valide.

Propriété 38. *1. $H \vDash G$ si et seulement si $H \rightarrow G$ est valide.*

2. Lorsque Γ est un ensemble fini de formules, $(\bigwedge_{H \in \Gamma} H) \rightarrow G$ est valide si et seulement si $\Gamma \vDash G$.

3. Si H est insatisfiable, alors pour toute formule G on a $H \vDash G$.

4. $H \vDash G$ si et seulement si $H \wedge \neg G$ est insatisfiable.

Démonstration. 1. $H \vDash G$ si et seulement si $(\forall \mu \in \mathbb{B}^{\mathcal{P}}, \llbracket H \rrbracket^{\mu} = \mathsf{V} \Rightarrow \llbracket G \rrbracket^{\mu} = \mathsf{V})$, si et seulement si $(\forall \mu \in \mathbb{B}^{\mathcal{P}}, \llbracket H \rrbracket^{\mu} = \mathsf{F}$ ou $\llbracket G \rrbracket^{\mu} = \mathsf{V})$ si et seulement si $(\forall \mu \in \mathbb{B}^{\mathcal{P}}, (\overline{\llbracket H \rrbracket^{\mu}} + \llbracket G \rrbracket^{\mu}) = \mathsf{V})$ si et seulement si $(\forall \mu \in \mathbb{B}^{\mathcal{P}}, \llbracket H \rightarrow G \rrbracket^{\mu} = \mathsf{V})$ si et seulement si $H \rightarrow G$ est valide.

2. De même.

3. Si H est insatisfiable, alors soit $\mu \in \mathbb{B}^{\mathcal{P}}$, $\llbracket H \rrbracket^{\mu} = \mathsf{F}$, ainsi $(\llbracket H \rrbracket^{\mu} = \mathsf{V} \Rightarrow \llbracket G \rrbracket^{\mu} = \mathsf{V})$, ceci étant vrai pour tout G.

4. $H \vDash G$ si et seulement si $(\forall \mu \in \mathbb{B}^{\mathscr{P}}, [\![H]\!]^{\mu} = \mathsf{V} \Rightarrow [\![G]\!]^{\mu} = \mathsf{V})$, si et seulement si $(\forall \mu \in \mathbb{B}^{\mathscr{P}}, [\![H]\!]^{\mu} = \mathsf{F}$ ou $[\![G]\!]^{\mu} = \mathsf{V})$, si et seulement si $(\forall \mu \in \mathbb{B}^{\mathscr{P}}, [\![H \wedge \neg G]\!]^{\mu} = \mathsf{F})$

□

Homework 39.

1. Montrer que, pour toutes formules G, H
   - $G \to H \equiv \neg H \to \neg G$
   - $G \equiv (\neg G \to \bot)$
   - $(G \leftrightarrow H) \equiv (G \to H) \wedge (H \to G)$
2. Démontrer le lemme 31.
3. En reprenant la vision ensembliste de la définition de la sémantique d'une formule du *homework* 28, proposer une définition au moyen d'opérateurs ensemblistes, lorsque H et G sont des formules de la logique propositionnelle et Γ est un ensemble de formules, de :
   - $H \equiv G$;
   - $H \vDash G$;
   - $\Gamma \vDash G$

   Comment interpréter alors la propriété 34 ?
4. Donner un exemple de formule valide, non valide, satisfiable, non satisfiable, et dans le cas d'une formule satisfiable donner un exemple de modèle.
5. Démontrer la remarque 37.

4 Le problème SAT, la validité

Dans cette section on présente les problèmes algorithmiques de la satisfiabilité, de la validité d'une formule de la logique propositionnelle.

Définition 40 (SAT)**.** Le problème SAT est le problème suivant.

$$\begin{cases} \textbf{Entrée :} \text{ Une formule } H \\ \textbf{Sortie} \ \ \textbf{:} \text{ Existe-t-il un environnement } \mu \text{ tel que } [\![H]\!]^{\mu} = \mathsf{V} \end{cases}$$

Définition 41 (VALIDE)**.** Le problème VALIDE est le problème suivant.

$$\begin{cases} \textbf{Entrée :} \text{ Une formule } H \\ \textbf{Sortie} \ \ \textbf{:} \text{ La formule } H \text{ est elle valide ?} \end{cases}$$

Dans les deux problèmes ci-dessus les instances des problèmes (les entrées) sont fournies sous la forme de l'arbre de syntaxe de la formule. Dans la suite nous nous intéresserons à des variantes de ces problèmes pour lesquels les instances sont supposées fournies sous d'autres formes. Ces deux problèmes sont des problèmes dits "de décision" : la sortie du problème est une valeur booléenne. On considère parfois les variantes des problèmes SAT et VALIDE dans lesquelles on ne demande non pas une réponse mais bien un témoin justifiant de la réponse.

$$\begin{cases} \textbf{Entrée :} \text{ Une formule } H \\ \textbf{Sortie} \ \ \textbf{:} \text{ Un environnement } \mu \text{ tel que } [\![H]\!]^{\mu} = \mathsf{V} \text{ ou None sinon} \end{cases}$$

$$\begin{cases} \textbf{Entrée :} \text{ Une formule } H \\ \textbf{Sortie} \ \ \textbf{:} \text{ Un environnement } \mu \text{ tel que } [\![H]\!]^{\mu} = \mathsf{F} \text{ ou None sinon} \end{cases}$$

4.1 Résolution du problème par tables de vérités étendues

Vocabulaire 42 (Tables de vérités étendue)**.** Lorsque $\mathcal{P}$ est fini, on appelle *table de vérité étendue* d'une formule $H \in \mathcal{F}$ la matrice de dimension $(2^{|\mathsf{vars}(H)|}+1) \times n$ (où n est le nombre de sous formules distinctes ♣ de H) contenant :

- sur sa première ligne : la donnée des n sous-formules distinctes de H ;
- sur chacune des $2^{|\mathsf{vars}(H)|}$ lignes suivantes : la donnée des valeurs de vérités de toutes les sous-formules de H pour chacun des $2^{|\mathsf{vars}(H)|}$ environnements propositionnels.

On illustre et on précise cette définition au moyen d'un exemple.

Exemple 43. Soit $\mathcal{P} = \{a, b, c\}$ et H la formule $(a \wedge b) \rightarrow (\neg b \vee \neg c)$. Ses sous-formules sont : $a, b, c, a \wedge b, \neg b, \neg c$ et $\neg b \vee \neg c$ et H elle-même. Si l'on souhaite calculer l'ensemble de toutes les valeurs de vérité de H, on peut remplir la table de vérité ci-dessous, de la gauche vers la droite, calculant ainsi les valeurs de vérité de toutes les sous-formules de H, jusqu'à H elle-même.

a	b	c	$a \wedge b$	$\neg b$	$\neg c$	$\neg b \vee \neg c$	H
F	F	F	F	V	V	V	V
F	F	V	F	V	F	V	V
F	V	F	F	F	V	V	V
F	V	V	F	F	F	F	V
V	F	F	F	V	V	V	V
V	F	V	F	V	F	V	V
V	V	F	V	F	V	V	V
V	V	V	V	F	F	F	F

Cette table correspond, pour chaque ligne, aux étapes de calcul de la valeur de vérité obtenue par évaluation de l'expression booléenne associée aux formules, pour la structure définie par l'environnement propositionnel correspondant à la ligne considérée.

La table de vérité étendue d'une formule H apporte sur celle-ci plusieurs informations.

Satisfiabilité. H est-elle satisfiable ? Oui, si au moins un V figure dans la dernière colonne de sa table de vérité ; l'environnement propositionnel rendant la formule vraie peut être lu sur cette même ligne.

Validité. H est-elle valide ? Oui si la dernière colonne de sa table de vérité ne contient que des V.

Insatisfiabilité. H est-elle insatisfiable ? Oui si la dernière colonne de sa table de vérité ne contient que des F.
Par exemple, la formule H ci-dessus est non valide, mais satisfiable. En effet, elle est non valide car l'environnement propositionnel $(a \mapsto \mathsf{V}, b \mapsto \mathsf{V}, c \mapsto \mathsf{V})$ ne satisfait pas H, ce qui se lit sur la dernière ligne de la table. Formellement on a $[\![H]\!]^{(a\mapsto\mathsf{V},b\mapsto\mathsf{V},c\mapsto\mathsf{V})} = \mathsf{F}$. Elle est satisfiable car, par exemple, l'environnement propositionnel $(a \mapsto \mathsf{F}, b \mapsto \mathsf{F}, c \mapsto \mathsf{F})$ est un modèle de H, ce qui se lit sur la première ligne de la table. Formellement on a $[\![H]\!]^{(a\mapsto\mathsf{F},b\mapsto\mathsf{F},c\mapsto\mathsf{F})} = \mathsf{V}$.

Homework 44.

1. Dresser la table de vérité étendue de la formule $\neg(a \vee \neg b) \rightarrow (\neg c)$. Dire si cette formule est satisfiable (en donner un modèle le cas échéant), valide, insatisfiable.
2. Proposer une méthode, basée sur l'utilisation de tables de vérités étendues, permettant de tester si deux formules sont équivalentes. De même pour la conséquence sémantique.

♣. qui ne sont pas $\top$ ou $\bot$

3. Existe-t-il une formule qui soit une conséquence sémantique de $H = a \vee (c \rightarrow b)$ mais qui ne soit pas équivalente à H ? Même question pour $H = a \vee ((c \rightarrow b) \vee c)$.

Remarque 45. La méthode des tables de vérité bien qu'apportant beaucoup d'informations est coûteuse à construire : pour donner un ordre de grandeur dans la modélisation du Sudoku nous avions $9 \times 9 \times 9 = 729$ un tableau de taille $2^{729} \times 730$ bits représente $2,576912737 \times 10^{209}$To de données.

Cette remarque justifie de s'intéresser à de meilleurs représentations et algorithmes pour la manipulation des formules de la logique propositionnelle.

5 Représentation des formules booléennes

5.1 Par tables de vérité

Vocabulaire 46. De manière générale, lorsque $\mathscr{P}$ est fini et contient n éléments, la table de vérité d'une fonction booléenne $f : \mathbb{B}^{\mathscr{P}} \rightarrow \mathbb{B}$ est un tableau à $2^n + 1$ lignes et $n + 1$ colonnes. Excepté pour la première ligne qui est utilisée pour donner des noms, la lecture d'une ligne du tableau commence donc par n colonnes formant un environnement booléen μ et se termine par une colonne indiquant la valeur de $f(\mu)$. Si $\mathscr{P}$ n'est pas fini, sa définition par table de vérité n'est pas possible.

Exemple 47. Une fonction booléenne en regard de sa table de vérité.

$$\begin{array}{rcl} f((a \mapsto \mathsf{F}, b \mapsto \mathsf{F})) & = & \mathsf{F} \\ f((a \mapsto \mathsf{F}, b \mapsto \mathsf{V})) & = & \mathsf{V} \\ f((a \mapsto \mathsf{V}, b \mapsto \mathsf{F})) & = & \mathsf{V} \\ f((a \mapsto \mathsf{V}, b \mapsto \mathsf{V})) & = & \mathsf{V} \end{array} \qquad \begin{array}{c|c|c} a & b & f \\ \hline \mathsf{F} & \mathsf{F} & \mathsf{F} \\ \mathsf{F} & \mathsf{V} & \mathsf{V} \\ \mathsf{V} & \mathsf{F} & \mathsf{V} \\ \mathsf{V} & \mathsf{V} & \mathsf{V} \end{array}$$

On prendra bien garde à ne pas confondre la table de vérité étendue d'une formule de la logique propositionnelle avec la table de vérité d'une fonction booléenne. On notera toutefois que la dernière colonne de la table de vérité étendue d'une formule de la logique propositionnelle H et la dernière colonne de la table de vérité de la fonction booléenne $[\![H]\!]$ sont identiques.

5.2 Par des formules de la logique propositionnelles

On s'intéresse maintenant au théorème ci-dessous, énonçant l'exhaustivité de la logique propositionnelle (vis-à-vis des fonctions booléennes). Cette section contient les définitions et lemmes qui aboutissent à la preuve de ce théorème d'exhaustivité.

Théorème 48 (Exhaustivité de la logique propositionnelle)**.** *Pour toute fonction booléenne f sur un ensemble de variables propositionnelles $\mathscr{P}$* fini*, il existe une formule de la logique propositionnelle H telle que $[\![H]\!] = f$.*

Commençons par donner une idée de la preuve sur un exemple.

Exemple 49. Considérons la fonction booléenne f dont la table de vérité est :

a	b	c	f
F	F	F	V
F	F	V	F
F	V	F	F
F	V	V	F
V	F	F	V
V	F	V	F
V	V	F	F
V	V	V	V

Pour construire une formule H dont f est la sémantique, on peut lire la table de vérité en considérant les trois lignes où f vaut V de la manière suivante : "f est vraie si et seulement si (a est faux et b est faux et c est faux) ou (a est vrai et b est faux et c est faux) ou (a est vrai et b est vrai et c est vrai)". On transcrit "a est vrai" par la formule a et "a est faux" par la formule $\neg a$, de même pour b et c. On obtient ainsi la formule $H = (\neg a \wedge \neg b \wedge \neg c) \vee (a \wedge \neg b \wedge \neg c) \vee (a \wedge b \wedge c)$.

Définition 50. Soit $\mathscr{P} = \{p_1, \ldots, p_n\}$ un ensemble fini de variables propositionnelles et $\mu \in \mathbb{B}^{\mathscr{P}}$ un environnement propositionnel. Pour chaque $p_i \in \mathscr{P}$, on pose :

$$\mathsf{lit}_\mu(p_i) = \begin{cases} p_i & \text{si } \mu(p_i) = \mathsf{V} \\ \neg p_i & \text{si } \mu(p_i) = \mathsf{F} \end{cases}$$

La formule H_μ associée à μ est $H_\mu = \bigwedge_{p_i \in \mathscr{P}} \mathsf{lit}_\mu(p_i)$.

On peut à présent définir une méthode générale d'extraction d'une formule à partir d'une table de vérité. Soit une table t de taille $(2^n + 1) \times (n + 1)$, on note $t_{i,j}$ la valeur de t à la ligne i, colonne j.

1. On utilise les symboles de la première ligne (sauf le dernier) pour constituer un ensemble de variables propositionnelles $\mathscr{P}_t = \{t_{0,0}, \ldots, t_{0,n-1}\}$.
2. Pour chaque ligne i (sauf la première), on construit l'environnement $\mu_i \in \mathbb{B}^{\mathscr{P}_t}$ défini par $\mu_i = (t_{0,0} \mapsto t_{i,0}, \ldots, t_{0,n-1} \mapsto t_{i,n-1})$. On a donc $t_{i,j} = \mu_i(t_{0,j})$.
3. Pour chaque ligne i (sauf la première) on construit la conjonction H_{μ_i} selon la définition.
4. On construit la disjonction $H = \bigvee_{\{H_{\mu_i} | t_{i,n}=1\}} H_{\mu_i}$, c-à-d la disjonction de toutes les formules H_{μ_i} extraites des lignes où la fonction prend la valeur 1 ; on a donc $t_{i,n} = f(\mu_i)$.

Pour montrer la correction de cette méthode de construction il faut montrer que la formule H ainsi extraite de la table t d'une fonction booléenne f vérifie $[\![H]\!] = f$ (proposition 52 ci-dessous). Pour cela, montrons d'abord que chaque μ_i est l'unique environnement qui satisfait la formule H_{μ_i} introduite dans la définition 50 ci-dessus.

Lemme 51. *Soit $\mathscr{P}$ un ensemble fini de variables propositionnelles. Pour tout environnement $\mu_1 \in \mathbb{B}^{\mathscr{P}}$, $[\![H_{\mu_1}]\!]^{\mu_1} = V$ et pour tout environnement $\mu_2 \in \mathbb{B}^{\mathscr{P}}$, si $\mu_2 \neq \mu_1$ alors $[\![H_{\mu_1}]\!]^{\mu_2} = F$.*

Démonstration. Posons $\mathscr{P} = \{p_1, \ldots, p_n\}$ et soit $\mu_1 \in \mathbb{B}^{\mathscr{P}}$. On montre que pour chaque $p_i \in \mathscr{P}$, $[\![\mathsf{lit}_{\mu_1}(p_i)]\!]^{\mu_1} = \mathsf{V}$. En effet, si $\mu_1(p_i) = \mathsf{V}$ alors $\mathsf{lit}_{\mu_1}(p_i) = p_i$ et $[\![\mathsf{lit}_{\mu_1}(p_i)]\!]^{\mu_1} = \mathsf{V}$; si $\mu_1(p_i) = \mathsf{F}$ alors $\mathsf{lit}_{\mu_1}(p_i) = \neg p_i$ et $[\![\neg p_i]\!]^{\mu_1} = \mathsf{V}$. Comme H_{μ_1} est la conjonction des formules $\mathsf{lit}_{\mu_1}(p_i)$, $[\![H_{\mu_1}]\!]^{\mu_1} = \mathsf{V}$. Soit $\mu_2 \in \mathbb{B}^{\mathscr{P}}$ un environnement tel que $\mu_2(p_i) \neq \mu_1(p_i)$ pour au moins une variable propositionnelle p_i. On montre que $[\![\mathsf{lit}_{\mu_1}(p_i)]\!]^{\mu_2} = \mathsf{F}$. En effet, si $\mu_1(p_i) = \mathsf{V}$ alors $\mathsf{lit}_{\mu_1}(p_i) = p_i$ et $\mu_2(p_i) = \mathsf{F}$, donc $[\![\mathsf{lit}_{\mu_1}(p_i)]\!]^{\mu_2} = \mathsf{F}$; si $\mu_1(p_i) = \mathsf{F}$ alors $\mathsf{lit}_{\mu_1}(p_i) = \neg p_i$ et $\mu_2(p_i) = \mathsf{V}$, donc $[\![\mathsf{lit}_{\mu_1}(p_i)]\!]^{\mu_2} = [\![\neg p_i]\!]^{\mu_2} = \mu_2(p_i) = H$. Comme H_{μ_1} est la conjonction des formules $\mathsf{lit}_{\mu_1}(p_i)$, on obtient $[\![H_{\mu_1}]\!]^{\mu_2} = \mathsf{F}$. □

Propriété 52. *Soit $\mathcal{P}$ un ensemble fini de variables propositionnelles et $f \in \mathcal{F}(\mathcal{P})$ une fonction booléenne. La formule :*

$$H = \bigvee_{\{\mu \in \mathbb{B}^{\mathcal{P}} | f(\mu) = \mathsf{V}\}} H_\mu$$

est telle que $[\![H]\!] = f$.

Démonstration. Soit $\mu_1 \in \mathbb{B}^{\mathcal{P}}$. Montrons que $[\![H]\!]^{\mu_1} = f(\mu_1)$. Si $f(\mu_1) = \mathsf{F}$, on a $\mu_1 \notin \{\mu \in \mathbb{B}^{\mathcal{P}} \mid f(\mu) = \mathsf{V}\}$ et donc pour tout μ dans $\mathbb{B}^{\mathcal{P}}$ tel que $f(\mu) = \mathsf{V}$ on a $\mu_1 \neq \mu$ et donc d'après le lemme 51 on a $[\![H_\mu]\!]^{\mu_1} = \mathsf{F}$. Par définition de l'interprétation du connecteur $\vee$ on déduit que $[\![H]\!]^{\mu_1} = \mathsf{F}$. Si $f(\mu_1) = \mathsf{V}$, on a $\mu_1 \in \{\mu \in \mathbb{B}^{\mathcal{P}} \mid f(\mu) = \mathsf{V}\}$ et $[\![H_{\mu_1}]\!]^{\mu_1} = \mathsf{V}$. Puisque H_{μ_1} est un des termes de la disjonction H, on obtient $[\![H]\!]^{\mu_1} = \mathsf{V}$. □

Cette proposition permet donc bien d'associer à toute fonction booléenne une formule qui la représente, répondant à l'objectif annoncé de la section.
On peut adopter une lecture duale des tables de vérité qui consiste à considérer que f s'évalue à V dès lors que nous ne sommes dans aucun des cas où f s'évalue à V. On obtient ainsi une autre formule pour représenter une fonction booléenne. En adoptant ce point de vue, il suffit de sélectionner toutes les formules H_μ pour lesquelles $f(\mu) = 0$ et de constuire la formule $\bigvee_{\{\mu | f(\mu) = \mathsf{F}\}} H_\mu$: on peut alors exprimer que f s'évalue à V dès lors qu'aucune des formules H_μ n'est vraie, c-à-d considérer la formule $\neg \bigvee_{\{\mu | f(\mu) = 0\}} H_\mu$. En utilisant les lois de De Morgan, on montre que cette négation est équivalente à la formule $\bigwedge_{\{\mu | f(\mu) = 0\}} \neg H_\mu$ et que chaque formule $\neg H_\mu$ est équivalente à $\bigvee_{p_i \in \mathcal{P}} \neg \mathsf{lit}_\mu(p_i)$. Pour extraire directement cette formule de la table de vérité d'une fonction booléenne, on redéfinit lit_μ, de manière "négative" :

$$\mathsf{lit}'_\mu(p_i) = \begin{cases} \neg p_i & \text{si } \mu(p_i) = \mathsf{V} \\ p_i & \text{si } \mu(p_i) = \mathsf{F} \end{cases}$$

et on pose $H'_\mu = \bigvee_{p_i \in \mathcal{P}} \mathsf{lit}'_\mu(p_i)$. Enfin, on construit la formule $H' = \bigwedge_{\{\mu | f(\mu) = 0\}} H'_\mu$ qui constitue une autre représentation de f. En effet, on montre que $[\![H']\!]^\mu = f(\mu)$ pour tout environnement μ, en utilisant le même principe que dans la démonstration de la proposition 52.

Exemple 53. Reprenons la fonction booléenne de l'exemple 49, que l'on peut lire de la manière suivante : "pour que f s'évalue à V, il ne faut être dans aucun des cas où f s'évalue à F" . Ce point de vue permet d'obtenir la formule :

$$H = (a \vee b \vee \neg c) \wedge (a \vee \neg b \vee c) \wedge (a \vee \neg b \vee \neg c) \wedge (\neg a \vee b \vee \neg c) \wedge (\neg a \vee \neg b \vee c)$$

Les formules propositionnelles extraites d'une table de vérité font apparaître une forme particulière de formule de la logique propositionnelle : des disjonctions de conjonctions ou des conjonctions de disjonctions.
Nous revenons sur ces formes particulières dans les sections suivantes.

Remarque 54. On remarque au passage que les symboles $\rightarrow$ et $\leftrightarrow$ n'ont pas été utilisé pour la fabrication des formules. On a donc un résultat un peu plus fort que celui annoncé ci-avant.

Homework 55.

1. Appliquer l'algorithme de construction pour la formule booléenne $f \in \mathcal{F}\{a, b, c\}$ définie par $f(\mu) = \mathsf{V}$ si et seulement si $\mathsf{card}\{x \in \{a, b, c\} \mid \mu(x) = \mathsf{V}\} = 2$.
2. Donner, et prouver une borne sur la taille des formules obtenues par application de l'algorithme précédent lorsque l'ensemble des variables propositionnelles est de cardinal n.
3. Prouver le résultat avancé dans le paragraphe descriptif précédent l'exemple 53, à savoir que $[\![H']\!] = f$.

5.3 Par des formules sous formes normales

Dans cette section on s'intéresse à deux familles de formules de la logique propositionnelle que sont les formules sous formes normales (conjonctives ou disjonctives).

5.3.1 Définitions

Définition 56 (Clauses, Formes normales)**.** Soit $\mathscr{P}$ un ensemble de variables propositionnelles.

- Une *clause disjonctive* est une disjonction de littéraux de $\mathscr{L}(\mathscr{P})$ (ce peut être la clause disjonctive vide : $\bot$).
- Une *clause conjonctive* est une conjonction de littéraux de $\mathscr{L}(\mathscr{P})$ (ce peut être la clause conjonctive vide : $\top$).
- Une *forme normale conjonctive* (FNC ou CNF) est une conjonction de clauses disjonctives (ce peut être la conjonction vide $\top$).
- Une *forme normale disjonctive* (FND ou DNF) est une disjonction de clauses conjonctives (ce peut être la disjonction vide $\bot$).

Exemple 57. $(p \vee \neg r) \wedge (q \vee \neg r \vee q) \wedge p$ est une forme normale conjonctive. $\top$ est une forme normale conjonctive, c'est aussi une forme normale disjonctive.

Propriété 58. *Pour toute formule $F \in \mathscr{F}$, il existe une formule F_d en forme normale disjonctive telle que $F \equiv F_d$ et une formule F_c en forme normale conjonctive telle que $F \equiv F_c$.*

Démonstration. Ce résultat découle en fait de la section précédente puisque la preuve fournie dans la section précédente avait construit des formules sous formes normales conjonctives ou sous formes normales disjonctives. □

5.3.2 Obtention par réécritures successives

La Figure 2 contient un ensemble de règles de réécriture dérivées des équivalences sémantiques présentées ci-avant. Le symbole $\|$ est utilisé pour factoriser plusieurs règles, ainsi $\mathsf{V} \wedge F \| F \wedge \mathsf{V} \rightsquigarrow F$ désigne l'existence des deux règles $\mathsf{V} \wedge F \rightsquigarrow F$ et $F \wedge \mathsf{V} \rightsquigarrow F$.

Remarque 59. On ne définit pas précisément ici ce que signifie l'application d'une règle de réécriture sur une formule.

$\neg\neg H$	$\rightsquigarrow H$	(R0)
$(H_1 \vee H_2) \wedge H_3$	$\rightsquigarrow (H_1 \wedge H_3) \vee (H_2 \wedge H_3)$	(R1)
$H_1 \wedge (H_2 \vee H_3)$	$\rightsquigarrow (H_1 \wedge H_2) \vee (H_1 \wedge H_3)$	(R2)
$\neg(H_1 \wedge H_2)$	$\rightsquigarrow (\neg H_1) \vee (\neg H_2)$	(R3)
$\neg(H_1 \vee H_2)$	$\rightsquigarrow (\neg H_1) \wedge (\neg H_2)$	(R4)
$\top \wedge H \| H \wedge \top \| \bot \vee H \| H \vee \bot$	$\rightsquigarrow H$	(R5)
$\neg\top \| \bot \wedge H \| H \wedge \bot$	$\rightsquigarrow \bot$	(R6)
$\neg\bot \| \top \vee H \| H \vee \top$	$\rightsquigarrow \top$	(R7)
$H_1 \rightarrow H_2$	$\rightsquigarrow \neg H_1 \vee H_2$	(R8)
$H_1 \leftrightarrow H_2$	$\rightsquigarrow (H_1 \rightarrow H_2) \wedge (H_2 \rightarrow H_1)$	(R9)

FIGURE 2 – Système de réécriture pour la forme normale disjonctive

Exemple 60. “En appliquant” les règles de la Figure 2 on obtient :

$$
\begin{array}{ll}
\top \wedge ((\top \wedge c) \vee (a \wedge \neg b)) \wedge (a \vee \neg(b \vee c)) & \\
\rightsquigarrow ((\top \wedge c) \vee (a \wedge \neg b)) \wedge (a \vee \neg(b \vee c)) & \text{par (R5)} \\
\rightsquigarrow (c \vee (a \wedge \neg b)) \wedge (a \vee \neg(b \vee c)) & \text{par (R5)} \\
\rightsquigarrow ((c \vee (a \wedge \neg b)) \wedge a) \vee ((c \vee (a \wedge \neg b)) \wedge \neg(b \vee c)) & \text{par (R2)} \\
\rightsquigarrow (c \wedge a) \vee (a \wedge \neg b \wedge a) \vee ((c \vee (a \wedge \neg b)) \wedge \neg(b \vee c)) & \text{par (R1)} \\
\rightsquigarrow (c \wedge a) \vee (a \wedge \neg b \wedge a) \vee ((c \vee (a \wedge \neg b)) \wedge (\neg b \wedge \neg c)) & \text{par (R4)} \\
\rightsquigarrow (c \wedge a) \vee (a \wedge \neg b \wedge a) \vee (c \wedge \neg b \wedge \neg c) \vee (a \wedge \neg b \wedge \neg b \wedge \neg c) & \text{par (R1)}
\end{array}
$$

Le résultat obtenu est bien une disjonction de conjonctions de littéraux, c-à-d une forme normale disjonctive de la formule F.

Forme normale conjonctive La forme normale conjonctive s'obtient en remplaçant les règles de réécriture (R1) et (R2) par les nouvelles règles :

$$
\begin{array}{lll}
(F_1 \wedge F_2) \vee F_3 & \rightsquigarrow (F_1 \vee F_3) \wedge (F_2 \vee F_3) & \text{(R1')} \\
F_1 \vee (F_2 \wedge F_3) & \rightsquigarrow (F_1 \vee F_2) \wedge (F_1 \vee F_3) & \text{(R2')}
\end{array}
$$

Taille de la forme normale Parmi les règles de réécriture de la Figure 2, les deux règles (R1) et (R2) qui utilisent la distributivité font croître de manière importante la taille des formules obtenues. On peut même précisément mesurer cette explosion combinatoire comme le montre la proposition suivante.

Propriété 61. *Soit $n \geqslant 2$ un entier et H_n la formule :*

$$H_n = (a_1 \vee b_1) \wedge (a_2 \vee b_2) \wedge \cdots \wedge (a_n \vee b_n)$$

définie sur l'ensemble de variables propositionnelles $\mathscr{P}_n = \{a_1, b_1, \ldots, a_n, b_n\}$. L'application des règles de réécriture de la Figure 2 à partir de H_n produit la formule :

$$\bigvee_{P \in \wp([\![1,n]\!])} \bigwedge_{j=1}^{n} \begin{cases} a_j & \textit{si } j \in P \\ b_j & \textit{sinon} \end{cases}$$

Démonstration. On procède par récurrence sur n. Si $n = 2$, la formule construite en appliquant les règles (R1) et (R2) est $(b_1 \wedge b_2) \vee (a_1 \wedge b_2) \vee (b_1 \wedge a_2) \vee (a_1 \wedge a_2)$. Le tableau ci-dessous permet de vérifier que la propriété est donc vérifiée :

$P \in \wp(\{1,2\})$	$\emptyset$	$\{1\}$	$\{2\}$	$\{1,2\}$
$\bigwedge_{j=1}^{2} \begin{cases} a_j & \text{si } j \in P \\ b_j & \text{sinon} \end{cases}$	$(b_1 \wedge b_2)$	$(a_1 \wedge b_2)$	$(b_1 \wedge a_2)$	$(a_1 \wedge a_2)$

Supposons à présent la propriété vérifiée au rang n et montrons la au rang $n+1$.

$$\begin{aligned}
H_{n+1} &= (a_1 \vee b_1) \wedge (a_2 \vee b_2) \wedge \cdots \wedge (a_n \vee b_n) \wedge (a_{n+1} \vee b_{n+1}) \\
&= H_n \wedge (a_{n+1} \vee b_{n+1}) \\
&\rightsquigarrow (H_n \wedge a_{n+1}) \vee (H_n \wedge b_{n+1}) && \text{(règle (R2))} \\
&\rightsquigarrow \left(\left(\bigvee_{P \in \wp(\{1,\ldots,n\})} \bigwedge_{j=1}^{n} \begin{cases} a_j & \text{si } j \in P \\ b_j & \text{sinon} \end{cases} \right) \wedge a_{n+1} \right) \\
&\quad \vee \left(\left(\bigvee_{P \in \wp(\{1,\ldots,n\})} \bigwedge_{j=1}^{n} \begin{cases} a_j & \text{si } j \in P \\ b_j & \text{sinon} \end{cases} \right) \wedge b_{n+1} \right) \\
&&& \text{(hypothèse de récurrence)} \\
&\rightsquigarrow \left(\bigvee_{P \in \{P' \cup \{n+1\} \mid P' \in \wp(\{1,\ldots,n\})\}} \bigwedge_{j=1}^{n+1} \begin{cases} a_j & \text{si } j \in P \\ b_j & \text{sinon} \end{cases} \right) \\
&\quad \vee \left(\bigvee_{P \in \wp(\{1,\ldots,n\})} \bigwedge_{j=1}^{n+1} \begin{cases} a_j & \text{si } j \in P \\ b_j & \text{sinon} \end{cases} \right) && \text{(règle (R1))} \\
&= \bigvee_{P \in \wp(\{1,\ldots,n+1\})} \bigwedge_{j=1}^{n+1} \begin{cases} a_j & \text{si } j \in P \\ b_j & \text{sinon} \end{cases}
\end{aligned}$$

□

Notons alors que l'ensemble $\wp(\llbracket 1, n \rrbracket)$ des parties de l'ensemble $\{1, \ldots, n\}$ est un ensemble de cardinal 2^n, or la taille de la formule H_n initiale est de l'ordre de n. Ainsi la taille d'une forme normale disjonctive associée à une formule est potentiellement exponentielle en la taille de la formule initiale. Il en est de même pour la forme normale conjonctive.

Homework 62.

1. En utilisant les règles de réécriture proposées ci-dessus, proposer une forme normale conjonctive équivalente à $(a \wedge b) \vee (\neg(a \rightarrow (\neg b)))$.
2. Une forme normale disjonctive ?
3. Justifier que lorsque aucune règle de la Figure 2 ne peut s'appliquer, alors la formule est sous forme normale disjonctive.

6 Le problème SAT dans le cas de formes normales

6.1 Satisfiabilité d'une forme normale disjonctive

Le problème de la satisfiabilité d'une forme normale disjonctive admet une réponse qui peut être obtenue très simplement. En effet, soit une clause conjonctive C :

— s'il existe une variable propositionnelle p telle que $p \in C$ et $\neg p \in C$, alors C n'est clairement pas satisfiable pour tout environnement propositionnel μ, on a alors $[\![\bigwedge_{\ell_i \in C} \ell_i]\!]^\mu = \mathsf{F}$;

— sinon, l'environnement μ défini par :

$$\mu(p) = \begin{cases} \mathsf{V} & \text{si } p \in C \\ \mathsf{F} & \text{sinon} \end{cases}$$

satisfait C puisqu'on a alors $[\![\bigwedge_{\ell_i \in C} \ell_i]\!]^\mu = \mathsf{V}$. Notez que le cas "sinon" de la définition de μ inclut le cas où c'est $\neg p$ qui appartient à C et le cas où $p \notin C$.

Une forme normale disjonctive étant une disjonction de clauses conjonctives, elle est satisfiable si et seulement si une de ses clauses conjonctives l'est. Il est alors aisé de tester sa satisfiabilité en inspectant chaque clause conjonctive de la disjonction pour construire (s'il existe) un environnement propositionnel qui la satisfasse.

Nous avons donc ainsi un nouvel algorithme permettant de résoudre le problème de la satisfiabilité d'une formule F de la logique propositionnelle, ne nécessitant pas le calcul complet de sa table de vérité.
Cet algorithme consiste à :

1. calculer une forme normale disjonctive F_d logiquement équivalente à la formule F ;
2. essayer de satisfaire une des clauses conjonctives de F_d.

Rappelons toutefois qu'étant donnée une formule de taille n, la taille de la forme normale obtenue par réécriture peut atteindre 2^n. Ainsi l'algorithme de satisfiabilité présenté ici a un coût pire cas au moins exponentiel, comparable à celui de la méthode des tables de vérité.

Homework 63.

1. Proposer un algorithme de complexité linéaire (en la taille de la formule), permettant de résoudre le problème de la validité d'une formule sous forme normale conjonctive.

6.2 Algorithme de Quine

Dans cette section on s'intéresse à la satisfiabilité d'une formule sous forme normale conjonctive.

Définition 64 (Substitution)**.** Étant donné une formule G, une variable propositionnelle p et une formule H, on définit inductivement $G[p := H]$ par :

$$\begin{aligned}
\top[p := H] &= \top \\
\bot[p := H] &= \bot \\
q[p := H] &= q && \text{si } q \in \mathscr{P} \text{ et } q \neq p \\
p[p := H] &= H \\
(\neg H_1)[p := H] &= \neg(H_1[p := H]) \\
(H_1 \odot H_2)[p := H] &= H_1[p := H] \odot H_2[p := H] && \text{avec } \odot \in \{\wedge, \vee, \rightarrow, \leftrightarrow\}
\end{aligned}$$

Exemple 65. $(p \wedge (p \rightarrow q))[p := (p \wedge q)] = (p \wedge q) \wedge ((p \wedge q) \rightarrow q)$

Lemme 66. *6.2 Pour toute formule H, toute variable propositionnelle p et tout environnement propositionnel μ tel que $\mu(p) = V$, on a $[\![H[p := \top]]\!]^\mu = [\![H]\!]^\mu$. De même avec $\bot$ et F.*

Démonstration. Par induction sur la formule H. □

Lemme 67. *Pour toute formule F et pour toute variable $p \in \mathscr{P}$, H est satisfiable si et seulement si $H[p := \top]$ est satisfiable ou $H[p := \bot]$ est satisfiable.*

Démonstration. Soit H une formule et p une variable propositionnelle, supposons que H est satisfiable, soit alors μ un environnement propositionnel tel que $[\![H]\!]^\mu = \mathsf{V}$.
- Si $\mu(p) = \mathsf{V}$ alors du lemme $[\![H[p := \top]]\!]^\mu = \mathsf{V}$
- Sinon si $\mu(p) = \mathsf{F}$ alors du lemme $[\![H[p := \bot]]\!]^\mu = \mathsf{V}$

Réciproquement si $H[p := \top]$ est satisfiable, il existe μ tel que $[\![H[p := \top]]\!]^\mu = \mathsf{V}$, soit alors $\hat{\mu}$ égale à μ coïncidant avec μ pour toute variables propositionnelles, mais tel que $\mu(p) = \mathsf{V}$. Du lemme $[\![H]\!]^{\hat{\mu}} = [\![H[p := \top]]\!]^{\hat{\mu}} = \mathsf{V}$. De même pour $H[p := \bot]$. □

Des lemmes précédents on déduit une première version de l'algorithme de Quine. Remarquons que cet algorithme fonctionne parfaitement sans que la formule en entrée soit sous forme normale conjonctive. Cet algorithme est en fait un algorithme de retour sur trace "classique".

Algorithme 1 : Algorithme de Quine, version 0

Entrée : Une formule H
Sortie : H est elle satisfiable

```
1 si vars(H) = ∅ et H ≡ ⊤ alors
2 |   retourner Vrai ;
3 sinon si vars(H) = ∅ et H ≡ ⊥ alors
4 |   retourner Faux ;
5 sinon
6 |   Soit p ∈ vars(H) ;
7 |   Essayer Quine(H[p := ⊤]) ;
8 |   Puis Quine(H[p := ⊥]) ;
```

Homework 68.

1. Représenter, au moyen d'un arbre, l'exécution de l'algorithme de Quine sur la formule $(p \to q) \wedge (\neg p \to q)$.
2. Proposer un algorithme, de complexité linéaire en la taille de la formule, permettant de faire les tests algorithmiques des lignes 1 et 3 de l'algorithme 1 ci-dessus.

Les remarques ci-dessus nous présentent l'idée de l'algorithme de Quine, que l'on détaille ci-après. On peut en fait remarquer que l'algorithme de Quine opère sur les formules au moyen de quelques opérations simples :
- Tester si la formule H possède une variable, en trouver une si c'est le cas.
- Tester si la formule est équivalente à $\top$, tester si la formule est équivalente à $\bot$.
- Effectuer une substitution d'une variable par $\top$, par $\bot$.

Remarquons par ailleurs que l'algorithme ne considère les formules qu'à équivalence près. Les formules sous forme normale conjonctive admette une représentation en machine permettant d'effectuer simplement les opérations ci-dessus.

Vocabulaire 69 (Vision ensembliste d'une forme normale)**.** Une forme normale (conjonctive ou disjonctive) peut être vue comme un ensemble d'ensemble de littéraux. Les ensembles sont alors entendus conjonctivement ou disjonctivement selon que c'est une forme normale conjonctive ou disjonctive. On omet les clauses contenant un littéral et sa négation (dans le cas d'une clause disjonctive, celle-ci est trivialement vraie et peut donc être retirée au sens où elle se trouve dans une conjonction, dans le cas d'une clause conjonctive, celle-ci est trivialement fausse et peut donc être retirée, au sens où elle se trouve dans une disjonction).

Exemple 70. Considérons la formule sous forme normale conjonctive $(p \vee q \vee p) \wedge (q \vee p) \wedge (p \vee \neg q) \wedge (q \vee \neg q)$. Alors la forme ensembliste de cette forme normale est $\{\{p, q\}, \{p, \neg q\}\}$.

Dans le cas d'une forme normale conjonctive comme celle de l'exemple ci-dessus, chaque clause est lue comme une contrainte à satisfaire : il faut rendre la clause disjonctive $\{p, q\}$ vraie, il faut rendre la clause disjonctive $\{p, \neg q\}$ vraie. Chaque clause disjonctive est lue comme une liste d'alternatives permettant de rendre la clause disjonctive vraie : pour rendre la clause disjonctive $\{p, \neg q\}$ vraie, il suffit de rendre p vraie et il suffit de rendre q faux.
On définit alors, pour une telle représentation des formules, un algorithme permettant de calculer la substitution d'une variable par $\top$, ou par $\bot$.
Remarquons par ailleurs qu'une formule $\mathcal{C}$ sous forme FNC ensembliste :
- est équivalente à $\top$ si et seulement si elle est vide,
- est équivalente à $\bot$ si et seulement si elle contient la clause vide.

On en déduit finalement l'algorithme de Quine.

Algorithme 2 : Assume

Entrée : Une formule sous FNC ensembliste $\mathcal{C}$, une variable p, un booléen b
Sortie : Une formule sous FNC ensembliste équivalente à $\mathcal{C}[p := \top]$ ou $\mathcal{C}[p := \bot]$ selon que b est V ou F

Soit lv le littéral p si b et $\neg p$ sinon ;
Soit lf le littéral $\neg p$ si b et p sinon ;
pour tout $C \in \mathcal{C}$ **faire**
    si lv $\in C$ **alors**
        Supprimer C de $\mathcal{C}$;
    sinon si lf $\in C$ **alors**
        Supprimer lf de C ;
    sinon
        Ne rien faire

Algorithme 3 : Algorithme de Quine

Entrée : Une formule sous forme FNC ensembliste $\mathcal{C}$
Sortie : $\mathcal{C}$ est elle satisfiable

si $\mathcal{C} = \emptyset$ **alors**
    retourner Vrai ;
sinon si $\emptyset \in \mathcal{C}$ **alors**
    retourner Faux ;
sinon si $\exists l \in \mathcal{L}(\mathcal{P}), \{l\} \in \mathcal{C}$ **alors**
    si $l = p \in \mathcal{P}$ **alors**
        Quine(Assume($\mathcal{C}, p,$ V)) ;
    sinon si $l = \neg p \in \mathcal{P}$ **alors**
        Quine(Assume($\mathcal{C}, p,$ F)) ;
sinon
    $p \leftarrow h(\mathcal{C})$;
    Essayer Quine(Assume($\mathcal{C}, p,$ V)) ;
    Puis Quine(Assume($\mathcal{C}, p,$ F)) ;

Remarque 71. h est une fonction “d’heuristique” à déterminer qui permet la sélection d’une des variables de la clause.

Homework 72.

1. Transformer la formule $(p \to q) \wedge (\neg p \to q)$ sous forme FNC ensembliste, puis représenter, au moyen d’un arbre, l’exécution de l’algorithme de Quine sur la formule ainsi obtenue.
2. Modifier l’algorithme de Quine pour qu’il ne résolve non plus le problème SAT mais le problème de retourner un modèle de la formule.
3. Démontrer que si une formule sous forme FNC ensembliste est non vide, alors elle n’est pas équivalente à $\top$. Démontrer qui si elle contient une clause vide, elle est équivalente à $\bot$.

6.3 n-SAT-CNF

Dans les sections précédentes les algorithmes proposés pour résoudre le problème SAT dans le cas d’une forme normale conjonctive ont tous une complexité algorithmique pire cas exponentielle. On essaie donc ici de restreindre le problème à des formules sous forme FNC qui soient plus contraintes.

Définition 73 (n-CNF)**.** On dit qu'une formule est sous forme n-CNF dès lors qu'elle est sous forme CNF et que chacune de ses clauses disjonctives contient au plus n littéraux.

Exemple 74.
- $(p) \wedge (\neg q) \wedge (\neg r)$ est sous forme 1-CNF
- $(p \vee q) \wedge (\neg q \vee \neg p) \wedge (\neg r)$ est sous forme 2-CNF
- $(p \vee q \vee r) \wedge (\neg q \vee \neg p) \wedge (\neg r)$ est sous forme 3-CNF

Remarque 75. Une clause disjonctive C est équivalente à une clause disjonctive dans laquelle chaque variable propositionnelle apparaît au plus une fois. En effet si une variable propositionnelle p apparaît sous la forme positive et négative alors la clause est équivalente à la clause disjonctive $\top$. En effet si une variable propositionnelle p apparaît plusieurs fois sous la même forme alors les doublons peuvent être retirés de la clause tout en préservant sa sémantique. Cette remarque a déjà été faite dans la section précédente.

Propriété 76. *Toute formule sur un ensemble $\mathcal{P}$ fini de cardinal n de variables est équivalente à une formule sous forme n-CNF.*

Démonstration. Il suffit de mettre la formule sous-forme CNF et d'utiliser la remarque précédente nous assurant que chaque variable peut apparaître au plus une fois par clause. □

Propriété 77. *Pour tout environnement propositionnel $\mathcal{P}$ de cardinal fini n, il existe une formule qui n'est équivalente à aucune formule sous forme $n-1$-CNF.*

Démonstration. Laissé en exercice. Considérer la formule $p_1 \vee p_2 \vee \cdots \vee p_n$. □

On définit alors, pour tout $n \in \mathbb{N}^\star$, le problème n-CNF-SAT comme étant :

$$n\text{-CNF-SAT} : \begin{cases} \textbf{Entrée :} \text{ Une formule } G \text{ sous forme } n\text{-CNF} \\ \textbf{Sortie :} \ G \text{ est-elle satisfiable ?} \end{cases}$$

On considère alors le problème 3SAT comme étant le problème 3-CNF-SAT.
La proposition 77 semble nous dire qu'il est vain de chercher à résoudre le problème n-CNF-SAT en espérant que cela permette de résoudre le problème plus général CNF-SAT. Toutefois pour répondre au problème SAT sur une entrée H, il n'est pas nécessaire de trouver une formule sous forme n-CNF-SAT qui soit équivalente à H, mais seulement une qui soit satisfiable si et seulement si H l'est. C'est en ce sens qu'intervient le théorème fondamentale suivant.

Théorème 78. *Pour toute formule G, il existe une formule H sous forme 3-CNF, telle que G et H sont équisatisfiables (comprendre : G est satisfiable si et seulement H l'est). De plus on a un algorithme de complexité polynomiale permettant le calcul de H à partir de l'entrée G.*

Démonstration. Soit une formule G de la logique propositionnelle sur l'ensemble de variables propositionnelles $\mathcal{P}$, Soit $\mathcal{S}$ l'ensemble de ses sous-formules. à chaque sous-formule H de G on associe une variable propositionnelle p_H, et une formule G_H :
- $\top$ si $H = \top$
- $\bot$ si $H = \bot$
- p si $H = p \in \mathcal{P}$
- $\neg p_{H_1}$ si $H = \neg H_1$
- $p_{H_1} \odot p_{H_2}$ si $H = H_1 \odot H_2$ avec $\odot \in \{\wedge, \vee, \rightarrow, \leftrightarrow\}$

On remarque que pour tout H, G_H a au plus 2 variables, la formule $R_H^0 \stackrel{\text{déf}}{=} (p_H \leftrightarrow G_H)$ a donc au plus 3 variables, ainsi R_H^0 est équivalente à une formule R_H sous forme 3-CNF comportant au plus 8 clauses disjonctives. Ce résultat provient des sections précédentes, on en rappelle ici le raisonnement.

- R_H^0 est une formule contenant au plus 3 variables, on peut donc dresser la table de vérité de sa négation. Par exemple pour la formule : $R_H^0 = x \leftrightarrow (y \wedge z)$ on aurait la table de vérité :

x	y	z	$[\![\neg R_H^0]\!]$
F	F	F	F
F	F	V	F
F	V	F	F
F	V	V	V★
V	F	F	V★
V	F	V	V★
V	V	F	V★
V	V	V	F

- On fabrique alors une forme normale disjonctive équivalente à $\neg R_H^0$ par lecture des lignes de la table de vérité indiquant que la formule s'évalue à V. Pour l'exemple où $R_H^0 = x \leftrightarrow (y \wedge z)$, on a donc $\neg R_H^0 \equiv (\neg x \wedge y \wedge z) \vee (x \wedge \neg y \wedge \neg z) \vee (x \wedge \neg y \wedge z) \vee (x \wedge y \wedge \neg z)$.
- On utilise alors les lois de De Morgan pour obtenir une forme normale conjonctive équivalente à R_H^0. Pour l'exemple où $R_H^0 = x \leftrightarrow (y \wedge z)$, on a donc $R_H^0 \equiv (x \vee \neg y \vee \neg z) \wedge (\neg x \vee y \vee z) \wedge (\neg x \vee y \vee \neg z) \wedge (\neg x \vee \neg y \vee z)$.
- On remarque que cette 3-CNF équivalente à R_H^0 est une conjonction d'au plus $8 = 2^3$ disjonctions puisque c'est le nombre de lignes de la table de vérité.

Considérons alors la formule $R \overset{\text{déf}}{=} \bigwedge_{H \in \mathcal{S}} R_H$. R est équivalente à la formule $\bigwedge_{H \in \mathcal{S}} R_H^0$, cette formule exprime les "règles" d'interprétation de la formule G. Notons $\tilde{\mathcal{P}} = \mathcal{P} \cup \bigcup_{H \in \mathcal{S}} p_H$. Autrement dit $\tilde{\mathcal{P}}$ est l'ensemble fini de toutes les variables pouvant apparaître dans la formule R.

Lemme 79. *Pour tout $\rho \in \mathcal{P} \to \mathbb{B}$, il existe $\mu \in \tilde{\mathcal{P}}$ tel que $\mu_{|\mathcal{P}} = \rho$ et $[\![R]\!]^\mu = V$. Autrement dit, pour tout environnement propositionnel ρ sur l'ensemble de variables $\mathcal{P}$, il existe un environnement propositionnel μ qui "étend" ρ aux nouvelles variables et tel que μ est un modèle de la formule R.*

Démonstration. Soit $\rho \in \mathcal{P} \to \mathbb{B}$, on construit :

$$\mu : \begin{cases} \tilde{\mathcal{P}} & \to \mathbb{B} \\ x & \mapsto \begin{cases} \rho(x) & \text{si } x \in \mathcal{P} \\ [\![H]\!]^\rho & \text{sinon si } x = p_H \end{cases} \end{cases}$$

Un tel μ convient. Il est clair que $\mu_{|\mathcal{P}} = \rho$. Montrons de plus que $[\![R]\!]^\mu = \mathsf{V}$. Soit donc $H \in \mathcal{S}$, montrons que $[\![R_H]\!]^\mu = [\![R_H^0]\!]^\mu = \mathsf{V}$. Par disjonction de cas sur H.

- Si $H = \top$, alors $R_H^0 = p_H \leftrightarrow \top$, or : $[\![p_H]\!]^\mu = \mu(p_H) = [\![H]\!]^\rho = [\![\top]\!]^\rho = \mathsf{V} = [\![\top]\!]^\mu$. Finalement $[\![p_H]\!]^\mu = [\![\top]\!]^\mu$ donc $[\![R_H^0]\!]^\mu = \mathsf{V}$.
- Si $H = \bot$, alors $R_H^0 = p_H \leftrightarrow \bot$, or : $[\![p_H]\!]^\mu = \mu(p_H) = [\![H]\!]^\rho = [\![\bot]\!]^\rho = \mathsf{F} = [\![\bot]\!]^\mu$. Finalement $[\![p_H]\!]^\mu = [\![\bot]\!]^\mu$ donc $[\![R_H^0]\!]^\mu = \mathsf{V}$.
- Si $H = p \in \mathcal{P}$, alors $R_H^0 = p_H \leftrightarrow p$, or : $[\![p_H]\!]^\mu = \mu(p_H) = [\![H]\!]^\rho = [\![p]\!]^\rho = \mu(p) = [\![p]\!]^\mu$. Finalement $[\![p_H]\!]^\mu = [\![p]\!]^\mu$ donc $[\![R_H^0]\!]^\mu = \mathsf{V}$.
- Si $H = \neg H_1$, alors $R_H^0 = p_H \leftrightarrow \neg p_{H_1}$, or : $[\![p_H]\!]^\mu = \mu(p_H) = [\![H]\!]^\rho = [\![\neg H_1]\!]^\rho = \overline{[\![H_1]\!]^\rho} = \overline{\mu(p_{H_1})} = \overline{[\![p_{H_1}]\!]^\mu} = [\![\neg p_{H_1}]\!]^\mu$. Finalement $[\![p_H]\!]^\mu = [\![\neg p_{H_1}]\!]^\mu$ donc $[\![R_H^0]\!]^\mu = \mathsf{V}$.
- Si $H = H_1 \wedge H_2$, alors $R_H^0 = p_H \leftrightarrow (H_1 \wedge H_2)$, or : $[\![p_H]\!]^\mu = \mu(p_H) = [\![H]\!]^\rho = [\![H_1 \wedge H_2]\!]^\rho = [\![H_1]\!]^\rho [\![H_2]\!]^\rho = \mu(p_{H_1})\mu(p_{H_2}) = [\![p_{H_1}]\!]^\mu [\![p_{H_2}]\!]^\mu = [\![p_{H_1} \wedge p_{H_2}]\!]^\mu$. Finalement $[\![p_H]\!]^\mu = [\![p_{H_1} \wedge p_{H_2}]\!]^\mu$ donc $[\![R_H^0]\!]^\mu = \mathsf{V}$.
- De même pour les autres cas.

Ceci étant vrai pour toute sous-formule H de G, on en conclut que $[\![R]\!]^\mu = [\![\bigwedge_{H \in \mathcal{S}} R_H]\!]^\mu = \mathsf{V}$. □

Lemme 80. *Pour tout modèle μ de R ($[\![R]\!]^\mu = V$), en notant $\rho = \mu_{|\mathcal{P}}$, pour tout $H \in \mathcal{S}$, $[\![H]\!]^\rho = \mu(p_H)$.*

Démonstration. Soit μ tel que $[\![R]\!]^\mu$, soit $\rho = \mu_{|\mathscr{P}}$. Remarquons donc que pour tout $H \in \mathcal{S}$, $[\![R_H]\!]^\mu = V$, donc $[\![p_H \leftrightarrow G_H]\!]^\mu = \mathsf{V}$ donc $[\![p_H]\!]^\mu = \mu(p_H) = [\![G_H]\!]^\mu$. Soit la propriété $\mathscr{P}_H : [\![H]\!]^\rho = \mu(p_H)$. Démontrons que pour tout $H \in \mathcal{S}$, $\mathscr{P}_H$ est vraie, par induction sur les sous-formules de G.

- Si $H = \top$ est dans $\mathcal{S}$ alors $G_H = \top$, or $\mu(p_H) = [\![G_H]\!]^\mu$ donc $\mu(p_H) = \mathsf{V} = [\![\top]\!]^\rho = [\![H]\!]^\rho$.
- Si $H = \bot$ est dans $\mathcal{S}$ alors $G_H = \bot$, or $\mu(p_H) = [\![G_H]\!]^\mu$ donc $\mu(p_H) = \mathsf{F} = [\![\bot]\!]^\rho = [\![H]\!]^\rho$.
- Si $H = p \in \mathscr{P}$ est dans $\mathcal{S}$ alors $G_H = p$, or $\mu(p_H) = [\![G_H]\!]^\mu$ donc $\mu(p_H) = \mu(p) = \rho(p) = [\![p]\!]^\rho = [\![H]\!]^\rho$.
- Si $H = \neg H_1$ est dans $\mathcal{S}$ alors H_1 est dans $\mathcal{S}$ et on suppose $\mathscr{P}_{H_1}$ vrai, à savoir : $[\![H_1]\!]^\rho = \mu(p_{H_1})$. Alors $G_H = \neg p_{H_1}$, or $\mu(p_H) = [\![G_H]\!]^\mu$ donc $\mu(p_H) = [\![\neg p_{H_1}]\!]^\mu = \overline{[\![p_{H_1}]\!]^\mu} = \overline{\mu(p_{H_1})} = \overline{[\![H_1]\!]^\rho} = [\![\neg H_1]\!]^\rho = [\![H]\!]^\rho$.
- Si $H = H_1 \wedge H_2$ est dans $\mathcal{S}$ alors H_1 et H_2 sont dans $\mathcal{S}$ et on suppose $\mathscr{P}_{H_1}$ et $\mathscr{P}_{H_2}$ vrais, à savoir : $[\![H_1]\!]^\rho = \mu(p_{H_1})$ et $[\![H_2]\!]^\rho = \mu(p_{H_2})$. Alors $G_H = p_{H_1} \wedge p_{H_2}$, or $\mu(p_H) = [\![G_H]\!]^\mu$ donc $\mu(p_H) = [\![p_{H_1} \wedge p_{H_2}]\!]^\mu = [\![p_{H_1}]\!]^\mu [\![p_{H_2}]\!]^\mu = \mu(p_{H_1})\mu(p_{H_2}) = [\![H_1]\!]^\rho [\![H_2]\!]^\rho = [\![H_1 \wedge H_2]\!]^\rho = [\![H]\!]^\rho$.
- De même pour les autres cas.

On a donc bien établi, par induction, que toute sous-formule H de G est telle que : $[\![H]\!]^\rho = \mu(p_H)$. □

On considère finalement la formule $\tilde{G} = R \wedge p_G$. $\tilde{G}$ est une formule sous forme 3-CNF. De plus $\tilde{G}$ est satisfiable si et seulement si G est satisfiable. En effet :

⇒ Si $\tilde{G}$ est satisfiable, soit μ tel que $[\![\tilde{G}]\!]^\mu = \mathsf{V}$, alors $[\![R]\!]^\mu = \mathsf{V}$ et $[\![p_G]\!]^\mu = \mu(p_G) = \mathsf{V}$. Soit alors $\rho = \mu_{|\mathscr{P}}$. Du lemme 80 on déduit que pour tout $H \in \mathcal{S}$, $[\![H]\!]^\rho = \mu(p_H)$, en particulier $[\![G]\!]^\rho = \mu(p_G) = \mathsf{V}$. Donc G est satisfiable.

⇐ Si G est satisfiable, soit ρ tel que $[\![G]\!]^\rho = \mathsf{V}$, soit alors $\mu \in \tilde{\mathscr{P}} \to \mathbb{B}$ tel que $\mu_{|\mathscr{P}} = \rho$ et $[\![R]\!]^\mu = \mathsf{V}$ (cette existence découle du lemme 79). En appliquant alors le lemme 80 à μ on en déduit que pour tout $H \in \mathcal{S}$, $[\![H]\!]^\rho = \mu(p_H)$, en particulier $[\![G]\!]^\rho = \mu(p_G) = \mathsf{V}$. Finalement $[\![\tilde{G}]\!]^\mu = [\![R \wedge p_G]\!]^\mu = [\![R]\!]^\mu [\![p_G]\!]^\mu = \mathsf{V} \cdot \mathsf{V} = \mathsf{V}$.

Une disjonction de 3 littéraux est de taille au plus $\underbrace{3}_{\text{variables}} + \underbrace{3}_{\neg} + \underbrace{2}_{\vee} = 8$. La 3-CNF $\tilde{G}$ que l'on vient de construire contient au plus $8|\mathcal{S}| + 1$ tels disjonctions. Finalement la formule $\tilde{G}$ est de taille au plus $\underbrace{8}_{\text{taille disjonction}} (8|\mathcal{S}| + 1) + \underbrace{8|\mathcal{S}|}_{\text{nombre de connecteurs } \wedge}$.

Ainsi il existe un algorithme de complexité polynomiale calculant la formule $\tilde{G}$ à partir de la formule G. □

Homework 81.

1. Appliquer le théorème ci-dessus à la formule $(p \to q) \wedge (\neg p \to q)$.
2. Proposer une démonstration du théorème ci-dessus dans le cas où la formule de départ est sous forme CNF.
3. Pourquoi n'a-t-on pas utilisé le point précédent comme démonstration du théorème.

Ainsi le théorème nous assure le résultat suivant : si l'on trouve un algorithme efficace (de complexité polynomiale) pour résoudre le problème 3SAT, on a l'assurance d'une solution algorithmique de complexité polynomiale au problème plus général SAT. Reste à trouver un algorithme efficace pour le problème 3SAT …