Chapitre I

Logique propositionnelle

Sommaire

1 Syntaxe 8
1.1 Formules strictes 8
1.2 Formules à priorité 11
2 Sens des formules 11
2.1 Sens des connecteurs 11
2.2 Valeur d'une formule 12
2.3 Définitions et notions élémentaires de logique 12
2.4 Compacité 15
2.5 Équivalences remarquables 16
3 Substitution et remplacement 17
3.1 Substitution 17
3.2 Remplacement 18
4 Formes Normales 19
4.1 Transformation en forme normale 19
4.2 Transformation en forme normale disjonctive (somme de monômes) 20
4.3 Transformation en forme normale conjonctive (produit de clauses) 21
5 Algèbre de Boole 21
5.1 Définition et notations 21
5.2 Propriétés 22
5.3 Dualité 24
6 Fonctions booléennes 25
6.1 Fonctions booléennes et somme de monômes 25
6.2 Fonctions booléennes et produit de clauses 26
7 Circuits 27
7.1 Portes logiques 27
7.2 Exemple de circuit combinatoire 28
8 L'outil BDDC 29
9 Exercices 31

ARISTOTE fut un des premiers à essayer de formaliser le raisonnement en utilisant la logique des syllogismes. La logique sert à préciser ce qu'est un raisonnement correct, indépendamment du domaine d'application. Un raisonnement est un moyen d'obtenir une conclusion à partir d'hypothèses données. Un raisonnement *correct* ne dit rien sur la vérité des hypothèses, il dit seulement qu'à partir de *la vérité des hypothèses, nous pouvons déduire la vérité de la conclusion*. Nous commençons l'étude de la logique par les lois de la logique propositionnelle. La logique propositionnelle est la logique *sans quantificateurs* qui s'intéresse uniquement aux lois gouvernant les opérations logiques suivantes : la négation ($\neg$) , la conjonction, autrement dit le « et » ($\wedge$), la disjonction, autrement dit le « ou » ($\vee$), l'implication ($\Rightarrow$) et l'équivalence ($\Leftrightarrow$). Ces opérations sont également appelées *connecteurs*. La logique propositionnelle permet de construire des raisonnements à partir de ces connecteurs. Considérons l'exemple suivant qui comporte trois hypothèses :

1. si Pierre est grand, alors Jean n'est pas le fils de Pierre,
2. si Pierre n'est pas grand, alors Jean est le fils de Pierre,
3. si Jean est le fils de Pierre alors Marie est la sœur de Jean.

Nous concluons que Marie est la sœur de Jean ou Pierre est grand.

Afin de pouvoir raisonner nous extrayons la structure logique des hypothèses. Nous désignons les phrases « Pierre est grand », « Jean est le fils de Pierre », « Marie est la sœur de Jean » respectivement par les lettres p, j, m. Les hypothèses peuvent donc s'écrire :

1. $p \Rightarrow \neg j$,
2. $\neg p \Rightarrow j$,
3. $j \Rightarrow m$,

et la conclusion se formalise en $m \vee p$. Nous montrons alors que les hypothèses impliquent la conclusion indépendamment de la nature des énoncés p, j, m. Pour cela nous prouvons que la formule suivante est vraie quelle que soit la vérité des propositions p, j, m.

$$((p \Rightarrow \neg j) \wedge (\neg p \Rightarrow j) \wedge (j \Rightarrow m)) \Rightarrow (m \vee p).$$

Plan : Nous débutons ce chapitre par la *syntaxe des formules logiques*, c'est-à-dire, les règles permettant d'écrire des formules. Une formule peut être vraie ou fausse, ainsi nous devons être capable d'évaluer le *sens d'une formule*. Pour cela nous introduisons le sens de chaque connecteur ainsi que le calcul de la valeur d'une formule qui en dérive. Nous montrons alors un résultat de compacité qui sera principalement utilisé dans la seconde partie de ce livre. Ensuite nous présentons des *équivalences remarquables* utiles pour simplifier les raisonnements logiques. D'autres méthodes permettent de simplifier les raisonnements logiques, par exemple le *remplacement* et la *substitution* de formule. Nous montrons ensuite comment construire les formes normales conjonctives ou disjonctives d'une formule en utilisant les équivalences remarquables. Ces formes normales permettent d'exhiber facilement les modèles ou les contre-modèles d'une formule. Nous montrons ensuite que la logique propositionnelle est une instance d'une *algèbre de Boole*. Nous introduisons la notion de *fonctions booléennes* et nous montrons qu'elles peuvent être utilisées pour formaliser des *circuits électroniques*. Enfin, nous présentons succinctement l'outil `BDDC` [1] développé par Pascal Raymond. Cet outil permet de manipuler les formules propositionnelles.

1 Syntaxe

Avant de raisonner, nous définissons le langage que nous utilisons. Ce langage est celui des formules construites à partir du *vocabulaire* suivant :

- Les constantes : $\top$ et $\bot$ représentant respectivement le *vrai* et le *faux*.
- Les variables : une variable est un identificateur, avec ou sans indice, par exemple x, y_1.
- Les parenthèses : ouvrante (et fermante).
- Les connecteurs : $\neg, \vee, \wedge, \Rightarrow, \Leftrightarrow$ respectivement appelés négation, disjonction (ou), conjonction (et), implication et équivalence.

La syntaxe définit les règles de construction d'une formule de la logique propositionnelle. Nous introduisons deux modes d'écriture d'une formule, l'un strict, l'autre plus souple dans le sens qu'il autorise plusieurs écritures d'une même formule. Cette souplesse est obtenue grâce à l'introduction de priorités entre les connecteurs logiques.

1.1 Formules strictes

Ci-dessous, nous donnons les règles de construction d'une formule stricte à partir du vocabulaire donné précédemment.

Définition 1.1 (Formule stricte) *Une* formule stricte *est définie de manière inductive comme suit :*

- $\top$ *et* $\bot$ *sont des formules strictes.*
- *Une variable est une formule stricte.*
- *Si A est une formule stricte alors* $\neg A$ *est une formule stricte.*
- *Si A et B sont des formules strictes et si* $\circ$ *est une des opérations* $\vee, \wedge, \Rightarrow, \Leftrightarrow$ *alors* $(A \circ B)$ *est une formule stricte.*

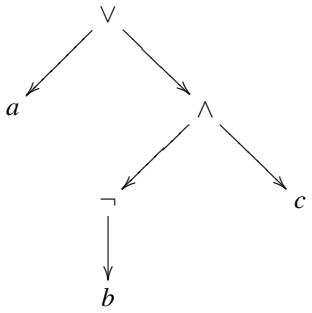
Dans la suite, nous désignons par $\circ$ tout connecteur binaire, et nous appellons simplement formule une formule stricte. Les formules différentes de $\top$, $\bot$ et des variables sont des formules décomposables.

1. `http://www-verimag.imag.fr/~raymond/tools/bddc-manual/bddc-manual-pages.html`.

Exemple 1.2 *L'expression* $(a \vee (\neg b \wedge c))$ *est une formule stricte construite suivant les règles précédentes. En revanche,* $a \vee (\neg b \wedge c)$ *et* $(a \vee (\neg (b) \wedge c))$ *ne sont pas des formules au sens de la définition 1.1 page ci-contre.*

L'intérêt de la définition de formules strictes est que les parenthèses permettent de trouver sans ambiguïté la structure des formules. Nous représentons la structure des formules par un arbre, où les feuilles contiennent les constantes ou les variables et les nœuds les connecteurs logiques. Le nœud racine est le connecteur à appliquer en dernier.

Exemple 1.3 *La structure de la formule* $(a \vee (\neg b \wedge c))$ *est mise en évidence par l'arbre suivant :*



Une formule peut être vue comme une liste de symbole (connecteurs, parenthèses, variables, et constantes). Un facteur d'une telle liste est une suite de symboles consécutifs dans la liste.

Définition 1.4 (Sous-formule) *Nous appelons* sous-formule *d'une formule (stricte) A tout facteur de A qui est une formule (stricte).*

Par exemple, $(\neg b \wedge c)$ est une sous-formule de $(a \vee (\neg b \wedge c))$.

Nous montrons que les formules sont décomposables d'une façon unique en leurs sous-formules. Ce résultat, précisé par le théorème 1.12 page suivante, est évident sur les exemples. Mais la preuve de ce résultat nécessite de nombreux résultats intermédiaires prouvés par récurrence sur la longueur d'une formule. L'unicité de la décomposition implique que nous pouvons identifier une formule et son arbre de décomposition. Ainsi, une sous-formule de la formule A pourra être identifiée comme un sous-arbre de l'arbre représentant la formule A.

Définition 1.5 (Longueur d'une formule) *La* longueur *d'une formule A est le nombre de symboles utilisés pour écrire A, dénotée* $l(A)$.

Si nous voyons une formule comme un mot sur le vocabulaire dont les éléments sont les constantes, les variables, les parenthèses et les connecteurs. Un mot sur ce vocabulaire est une suite d'éléments sur ce vocabulaire et la longueur du mot est la longueur de la suite. Dans l'exercice 5 page 32 nous demandons de donner la définition inductive de la longueur d'une formule.

Exemple 1.6 *Soient la formule* $A = (a \vee b)$ *et* $B = (A \wedge \neg A)$*, nous avons* $l(A) = 5$ *et* $l(B) = 4+5+5 = 14$.

Lemme 1.7 (Équilibre des parenthèses) *Toute formule a un nombre égal de parenthèses ouvrantes et de parenthèses fermantes.*

Preuve : Par définition des formules, toute parenthèse ouvrante est associée à une parenthèse fermante. Le lemme s'en déduit immédiatement (par une récurrence trop simple pour mériter d'être explicitée). □

Lemme 1.8 (Relation entre les parenthèses) *Tout préfixe d'une formule a un nombre de parenthèses ouvrantes au moins égal à celui des parenthèses fermantes.*

Preuve : Supposons le lemme vérifié pour toute formule de longueur inférieure à n. Soit A une formule de longueur n. Montrons que le lemme est vrai pour A.

1. Soit A une variable ou une constante. Le lemme est vérifié.
2. Soit $A = \neg B$ où B est une formule. Un préfixe de A est vide ou s'écrit $\neg B'$, où B' est un préfixe de B. Par hypothèse de récurrence, B' a un nombre de parenthèses ouvrantes au moins égal à celui des parenthèses fermantes. Par suite il en est de même de tout préfixe de A.

3. Soit $A = (B \circ C)$ où B et C sont des formules. Un préfixe de A est soit vide, soit s'écrit $(B'$ où B' est un préfixe de B, soit s'écrit $(B \circ C'$ où C' est un préfixe de C, soit est égal à A. Examinons ces différents cas.
   (a) Le préfixe vide a un nombre de parenthèses ouvrantes au moins égal à celui des parenthèses fermantes.
   (b) Par hypothèse de récurrence, B' a un nombre de parenthèses ouvrantes au moins égal à celui des parenthèses fermantes, donc il en est de même de $(B'$.
   (c) Par hypothèse de récurrence, B et C' ont un nombre de parenthèses ouvrantes au moins égal à celui des parenthèses fermantes, donc il en est de même de $(B \circ C'$.
   (d) D'après le lemme 1.7 page précédente, le nombre de parenthèses ouvrantes de A est égal (donc au moins égal) au nombre des parenthèses fermantes.

Ainsi, si le lemme est vrai pour toute formule de longueur inférieure à n, il est vrai pour toute formule de longueur n. Donc, par récurrence, il est vrai pour une formule de longueur quelconque. □

Afin de raisonner par induction sur la structure d'une formule, nous définissons la taille d'une formule. Nous remarquons que la taille d'une formule correspond au nombre de connecteurs qu'elle contient.

Définition 1.9 (Taille d'une formule) *La* taille d'une formule *A, notée* $|A|$*, est définie inductivement par :*
- $|\top| = 0$ *et* $|\bot| = 0$.
- *Si A est une variable alors* $|A| = 0$.
- $|\neg A| = 1 + |A|$.
- $|(A \circ B)| = |A| + |B| + 1$.

Exemple 1.10 $|(a \vee (\neg b \wedge c))| = 3$.

La *taille* des formules définie dans la définition 1.9 est une mesure utile pour prouver par récurrence des propriétés sur les formules. La preuve du lemme 1.11 illustre comment écrire une preuve par récurrence sur la taille des formules. Rappelons qu'un préfixe strict d'une formule A est un préfixe de longueur strictement plus petite que la longueur de A.

Lemme 1.11 (Préfixe strict) *Tout préfixe strict d'une formule n'est pas une formule.*

Preuve : Nous effectuons une preuve par induction sur la taille de la formule :

Cas de base : Soit A une formule de taille 0, A est donc soit une variable soit une constante. Le seul préfixe strict de A est le mot vide qui n'est pas une formule.

Induction : Supposons le lemme vérifié pour toute formule de longueur inférieure à n, avec $n > 0$. Soit A une formule de longueur n. Montrons que le lemme est vrai pour A.
- Soit $A = \neg B$, où B est une formule. Supposons par contradiction que A ait un préfixe strict qui soit une formule. Ce préfixe doit s'écrire $\neg B'$ où B' est une formule et un préfixe strict de B. C'est impossible d'après l'hypothèse de récurrence.
- Soit $A = (B \circ C)$ où B et C sont deux formules. Supposons par contradiction que A ait un préfixe strict qui soit une formule. Ce préfixe doit s'écrire $(B' \circ C')$ où B' et C' sont des formules. La formule B est un préfixe de la formule B' ou la formule B' est un préfixe de la formule B. Il est impossible, d'après l'hypothèse de récurrence, que ces préfixes soient stricts. Donc $B = B'$. Par suite C') est un préfixe de C. D'après le lemme 1.7 page précédente, C' a autant de parenthèses ouvrantes que de fermantes. Donc C') est un préfixe de la formule C, qui a plus de parenthèses fermantes que de parenthèses ouvrantes, ce qui contredit le lemme 1.8 page précédente. Donc A n'a pas de préfixe strict qui est une formule.

Ainsi par récurrence, tout préfixe strict d'une formule n'est pas une formule. □

Théorème 1.12 *Pour toute formule A, un et un seul de ces cas se présente :*
- *A est une variable,*
- *A est une constante,*
- *A s'écrit d'une unique façon sous la forme* $\neg B$ *où B est une formule,*
- *A s'écrit d'une unique façon sous la forme* $(B \circ C)$ *où B et C sont des formules.*

Preuve : Puisque les vocabulaires constantes, variables, négation, parenthèses sont disjoints, toute formule A est d'une et d'une seule des formes variable, constante, $\neg B$ où B est une formule, $(B \circ C)$ où B et C sont des formules. Il suffit de montrer l'unicité de la dernière décomposition. Supposons que $A = (B \circ C) = (B' \circ C')$ où B, B', C, C' sont des formules.

La formule B est préfixe de B' ou la formule B' est préfixe de B. D'après le lemme 1.11 page ci-contre, il est impossible que ce soient des préfixes stricts, donc $B = B'$ et par suite $C = C'$. □

Avec la définition de formules (strictes) nous écrivons de nombreuses parenthèses inutiles comme les parenthèses qui entourent chaque formule. Nous introduisons maintenant plus de souplesse dans notre syntaxe en définissant des priorités.

1.2 Formules à priorité

Pour éviter la surabondance des parenthèses, nous définissons les *formules à priorité*.

Définition 1.13 (Formule à priorité) *Une* formule à priorité *est définie inductivement par :*
- $\top$ *et* $\bot$ *sont des formules à priorité,*
- *une variable est une formule à priorité,*
- *si A est une formule à priorité alors $\neg A$ est une formule à priorité,*
- *si A et B sont des formules à priorité alors $A \circ B$ est une formule à priorité,*
- *si A est une formule à priorité alors (A) est une formule à priorité.*

Exemple 1.14 *Considérons la formule $a \vee \neg b \wedge c$ qui est une formule à priorité mais pas une formule.*

En général, une formule à priorité n'est pas une formule (stricte). Nous montrons dans l'exercice 2 page 31 que toute formule est une formule à priorité. Afin de pouvoir supprimer des parenthèses sans aucune ambiguïté nous définissons un ordre de priorité entre les différents connecteurs.

Définition 1.15 (Ordre de priorité des connecteurs) *La négation est prioritaire, puis dans l'ordre des priorités décroissantes, nous trouvons la conjonction ($\wedge$), la disjonction ($\vee$), l'implication ($\Rightarrow$) et l'équivalence ($\Leftrightarrow$).*

À priorité égale, le connecteur gauche est prioritaire, **sauf pour l'implication** *qui est associative à droite* [2]

Nous considérons qu'une formule à priorité est *l'abréviation* de la formule reconstituable en utilisant les priorités. Sauf exception, nous identifions une formule et son abréviation. Autrement dit, ce qui nous intéresse dans une formule, ce n'est pas son écriture *superficielle*, c'est sa structure, qui est mise en évidence par la syntaxe « stricte ». Ainsi la taille d'une formule à priorité sera égale à la taille de la formule stricte dont elle est l'abréviation.

Exemple 1.16 *Nous donnons plusieurs exemples d'abréviation de formule par une formule à priorité :*
- *$a \wedge b \wedge c$ est l'abréviation de $((a \wedge b) \wedge c)$.*
- *$a \wedge b \vee c$ est l'abréviation de $((a \wedge b) \vee c)$.*
- *$a \vee b \wedge c$ est l'abréviation de $(a \vee (b \wedge c))$.*

Maintenant que la syntaxe est définie, nous définissons le sens des formules.

2 Sens des formules

Nous cherchons à déterminer si une formule est vraie ou fausse indépendamment des valeurs affectées à ses variables. Nous définissons d'abord le sens des connecteurs logiques. Ensuite nous expliquons comment calculer la valeur d'une formule et montrons le théorème de compacité. Nous terminons cette section par la présentation de définitions de notions de base de la logique qui constituent le langage commun des logiciens.

2.1 Sens des connecteurs

Nous désignons les valeurs de vérité par 0 pour faux et par 1 pour vrai. La constante $\top$ vaut 1 et la constante $\bot$ vaut 0, ce qui nous conduit, le plus souvent, à confondre les constantes et leurs valeurs, et à utiliser indifféremment $\top$, 1 et vrai, respectivement $\bot$, 0 et faux. Le sens des connecteurs logiques est donné par la table I.1 page suivante qui indique les valeurs des formules de la première ligne suivant les valeurs *assignées* aux variables x et y.

2. C'est-à-dire $a \Rightarrow b \Rightarrow c$ est l'abréviation de $(a \Rightarrow (b \Rightarrow c))$.

x	y	$\neg x$	$x \vee y$	$x \wedge y$	$x \Rightarrow y$	$x \Leftrightarrow y$
0	0	1	0	0	1	1
0	1	1	1	0	1	0
1	0	0	1	0	0	0
1	1	0	1	1	1	1

TABLE I.1 – Table de vérité des connecteurs.

2.2 Valeur d'une formule

Chacun sait évaluer une formule : nous associons à chaque variable de la formule une valeur dans l'ensemble $\mathbb{B} = \{0,1\}$. La valeur de la formule est obtenue en remplaçant les variables par leurs valeurs et en effectuant les opérations suivant la table I.1. Néanmoins, pour raisonner sur les formules, nous définissons formellement la valeur d'une formule.

Définition 2.1 (Assignation) *Une* assignation *est une application de l'ensemble de toutes les variables d'une formule dans l'ensemble* $\mathbb{B}$.

Définition 2.2 (Valeur d'une formule) *Soient A une formule et v une assignation,* $[A]_v$ *dénote la valeur de la formule A dans l'assignation v. La* valeur *de* $[A]_v$ *est définie par récurrence sur l'ensemble des formules. Soient A, B des formules, x une variable et v une assignation.*

- $[x]_v = v(x)$.
- $[\top]_v = 1$, $[\bot]_v = 0$.
- $[\neg A]_v = 1 - [A]_v$ *autrement dit pour calculer la valeur de* $\neg A$*, nous soustrayons à* 1 *la valeur de A.*
- $[(A \vee B)]_v = max\{[A]_v, [B]_v\}$.
- $[(A \wedge B)]_v = min\{[A]_v, [B]_v\}$.
- $[(A \Rightarrow B)]_v = si\ [A]_v = 0\ alors\ 1\ sinon\ [B]_v$.
- $[(A \Leftrightarrow B)]_v = si\ [A]_v = [B]_v\ alors\ 1\ sinon\ 0$.

D'après le théorème 1.12 page 10, toute formule (stricte) se décompose de façon unique en l'un des cas ci-dessus. Ainsi l'extension de v aux formules est une application des formules dans $\mathbb{B}$. En effet, soient 4 formules A, A', B, B' et deux opérations $\circ$ et $\circ'$ telles que $(A \circ B) = (A' \circ' B')$. Par unicité de la décomposition, $A = A', B = B', \circ = \circ'$, donc la valeur de la formule $(A \circ B)$ est définie uniquement par une et une seule des lignes de la définition de la valeur. Il est clair que la valeur d'une formule ne dépend que de ses variables et de sa structure, aussi l'évaluation d'une formule est présentée sous la forme d'une *table de vérité*.

Définition 2.3 (Table de vérité d'une formule) *Une* table de vérité d'une formule *A est un tableau qui représente la valeur de A pour toutes les valeurs possibles des variables de A.*

Chaque ligne de la table de vérité définit une assignation pour les variables des formules présentes dans les colonnes de la table et chaque colonne donne la valeur d'une formule.

Exemple 2.4 *Nous donnons la table de vérité des formules suivantes :* $x \Rightarrow y$, $\neg x$, $\neg x \vee y$, $(x \Rightarrow y) \Leftrightarrow (\neg x \vee y)$ *et* $x \vee \neg y$.

x	y	$x \Rightarrow y$	$\neg x$	$\neg x \vee y$	$(x \Rightarrow y) \Leftrightarrow (\neg x \vee y)$	$x \vee \neg y$
0	0	1	1	1	1	1
0	1	1	1	1	1	0
1	0	0	0	0	1	1
1	1	1	0	1	1	1

2.3 Définitions et notions élémentaires de logique

Nous listons les notions élémentaires de la logique. Nous illustrons par des exemples et contre-exemples chacune des définitions introduites afin d'en faire comprendre les subtilités.

Définition 2.5 (Formules équivalentes) *Deux formules A et B sont* équivalentes *si elles ont la même valeur pour toute assignation.*

Exemple 2.6 *Les colonnes des deux formules $x \Rightarrow y$ et $\neg x \vee y$ sont identiques dans l'exemple 2.4 page précédente. Ces deux formules sont donc équivalentes. Par contre les formules $x \Rightarrow y$ et $x \vee \neg y$ ne sont pas équivalentes car elles n'ont pas les mêmes tables de vérité.*

Remarque 2.7 *Nous n'utilisons pas le symbole du connecteur logique $\Leftrightarrow$ pour dire que A et B sont équivalentes. Nous notons que les formules A et B sont équivalentes par $A \equiv B$ ou simplement $A = B$ si le contexte nous permet de comprendre que le signe égal indique l'équivalence. Ainsi, $x \Rightarrow y = \neg x \vee y$ signifie que la formule $x \Rightarrow y$ est équivalente à la formule $\neg x \vee y$.*

Définition 2.8 (Valide, tautologie) *Une formule est* valide *si elle a la valeur* 1 *pour toute assignation. Une formule valide est aussi appelée une* tautologie.

Exemple 2.9 *En regardant la table de vérité de l'exemple 2.4 page ci-contre nous obtenons que :*
- *la formule $(x \Rightarrow y) \Leftrightarrow (\neg x \vee y)$ est valide ;*
- *la formule $x \Rightarrow y$ n'est pas valide car elle est fausse pour l'assignation $x = 1$ et $y = 0$, ce n'est donc pas une tautologie.*

Notation : $\models A$ dénote le fait que la formule A est valide. Nous pouvons écrire $\models x \vee \neg x$, car $x \vee \neg x$ est une tautologie.

Propriété 2.10 *Les formules A et B sont équivalentes si et seulement si la formule $A \Leftrightarrow B$ est valide.*

Preuve : La propriété est une conséquence de la table I.1 page précédente et des définitions précédentes.

$\Rightarrow$ Si les formules A et B sont équivalentes cela signifie qu'elles ont la même table de vérité, ainsi d'après la définition du connecteur $\Leftrightarrow$ donnée dans la table I.1 page ci-contre la table de vérité de $A \Leftrightarrow B$ contient uniquement des 1 donc $A \Leftrightarrow B$ est valide.

$\Leftarrow$ Si la formule $A \Leftrightarrow B$ est valide, nous déduisons que la table de vérité de $A \Leftrightarrow B$ contient uniquement des 1, ainsi d'après la définition du connecteur $\Leftrightarrow$ donnée dans la table I.1 page précédente les tables de vérité de A et de B coïncident donc les formules A et B sont équivalentes.

□

Définition 2.11 (Modèle d'une formule) *Une assignation v qui donne la valeur* 1 *à une formule, est un* modèle *de la formule. Nous dirons aussi que v* satisfait *la formule ou que v rend* vraie *la formule.*

Exemple 2.12 *$x \Rightarrow y$ a pour modèle l'assignation $x = 1, y = 1$ ou l'assignation $x = 0$ et y quelconque. Par contre l'assignation $x = 1$ et $y = 0$ n'est pas un modèle de $x \Rightarrow y$.*

Cette notion de modèle s'étend aux ensembles de formules comme suit.

Définition 2.13 (Modèle d'un ensemble de formules) *Une assignation est un* modèle d'un ensemble de formules *si et seulement si elle est un modèle de chaque formule de l'ensemble.*

Exemple 2.14 *L'assignation $a = 0, b = 0$ (et c quelconque) est modèle de $\{a \Rightarrow b, b \Rightarrow c\}$.*

Propriété 2.15 *Une assignation est un modèle d'un ensemble de formules, si et seulement si elle est un modèle de la conjonction des formules de l'ensemble.*

Preuve : La preuve est demandée dans l'exercice 12 page 33. □

Exemple 2.16 *L'ensemble de formules $\{a \Rightarrow b, b \Rightarrow c\}$ et la formule $(a \Rightarrow b) \wedge (b \Rightarrow c)$ ont les mêmes modèles.*

Définition 2.17 (Contre-Modèle) *Une assignation v qui donne la valeur* 0 *à une formule est un* contre-modèle *de la formule. Nous dirons que v ne satisfait pas la formule ou v rend la formule* fausse.

Exemple 2.18 *$x \Rightarrow y$ a pour contre-modèle l'assignation $x = 1, y = 0$.*

Remarque 2.19 (Contre-modèle d'un ensemble de formules) *La notion de contre-modèle s'étend aux ensembles de formules de la même manière que la notion de modèle.*

Définition 2.20 (Formule satisfaisable) *Une formule (respectivement un ensemble de formules) est* satisfaisable *s'il existe une assignation qui en est un modèle.*

Définition 2.21 (Formule insatisfaisable) *Une formule (respectivement un ensemble de formules) est* insatisfaisable *si elle (respectivement s'il) n'est pas satisfaisable.*

Une formule (respectivement un ensemble de formules) insatisfaisable ne possède pas de modèle. Sa table de vérité ne comporte que des 0. La négation d'une tautologie est donc une formule insatisfaisable.

Exemple 2.22 *$x \wedge \neg x$ est insatisfaisable, mais $x \Rightarrow y$ ne l'est pas.*

Remarque 2.23 *Les logiciens utilisent le mot* consistant *comme synonyme de satisfaisable et* contradictoire *comme synonyme d'insatisfaisable.*

Définition 2.24 (Conséquence) *Soient Γ un ensemble de formules et A une formule : A est* conséquence *de l'ensemble Γ d'hypothèses si tout modèle de Γ est modèle de A. Le fait que A soit conséquence de Γ est noté par $\Gamma \models A$.*

Exemple 2.25 *D'après la table de vérité suivante, la formule $a \Rightarrow c$ est conséquence des hypothèses $a \Rightarrow b$ et $b \Rightarrow c$.*

a	b	c	$a \Rightarrow b$	$b \Rightarrow c$	$a \Rightarrow c$
0	0	0	1	1	1
0	0	1	1	1	1
0	1	0	1	0	1
0	1	1	1	1	1
1	0	0	0	1	0
1	0	1	0	1	1
1	1	0	1	0	0
1	1	1	1	1	1

Remarque 2.26 (Validité et conséquence) *Nous notons A est valide par $\models A$, car A est valide si et seulement si A est conséquence de l'ensemble vide.*

Maintenant, nous établissons l'équivalence de la validité d'une formule composée d'hypothèses et d'une conclusion avec la conséquence de la conclusion à partir des hypothèses mais aussi avec l'insatifaisabilité des hypothèses et de la négation de la conclusion. Ces relations sont constamment utilisées dans les exercices et les démonstrations.

Propriété 2.27 *Soient $n+1$ formules $A_1, \ldots, A_n$, B. Soit H_n la* conjonction *des formules $A_1, \ldots, A_n$. Les trois formulations suivantes sont équivalentes :*

1. *$A_1, \ldots, A_n \models B$, c'est-à-dire B est conséquence des hypothèses $A_1, \ldots, A_n$.*
2. *La formule $H_n \Rightarrow B$ est valide.*
3. *$H_n \wedge \neg B$ est insatisfaisable.*

Preuve : La propriété est une conséquence de la table I.1 page 12 et des définitions précédentes. Soient $n+1$ formules $A_1, \ldots A_n, B$ et $H_n = A_1 \wedge \ldots \wedge A_n$.

$1 \Rightarrow 2$: Supposons que $A_1, \ldots, A_n \models B$, c'est-à-dire tout modèle de $A_1, \ldots, A_n$ est aussi modèle de B.
- Considérons une assignation v qui n'est pas modèle de $A_1, \ldots, A_n$ cela signifie qu'il existe un i tel que $[A_i]_v = 0$, par conséquence $[H_n]_v = 0$ (d'après la table I.1 page 12), donc $[H_n \Rightarrow B]_v = 1$.
- Considérons une assignation v modèle de $A_1, \ldots, A_n$ cela signifie que $[A_i]_v = 1$ pour $i = 1, \ldots, n$, par conséquence $[H_n]_v = 1$ (d'après la table I.1 page 12). Par hypothèse v est un modèle de B donc $[B]_v = 1$, nous en déduisons $[H_n \Rightarrow B]_v = 1$.

Nous obtenons donc dans tous les cas que $H_n \Rightarrow B$ est valide.

$2 \Rightarrow 3$: Supposons que $H_n \Rightarrow B$ est valide, cela signifie que pour toute assignation v, $[H_n \Rightarrow B]_v = 1$. D'après la table I.1 page 12 nous avons deux possibilités soit $[H_n]_v = 0$, soit $[H_n]_v = 1$ et $[B]_v = 1$. Or $[H_n \wedge \neg B]_v = \min([H_n]_v, [\neg B]_v) = \min([H_n]_v, 1 - [B]_v)$. Dans les deux cas, nous obtenons $[H_n \wedge \neg B]_v = 0$. Nous concluons donc que $H_n \wedge \neg B$ est insatisfaisable.