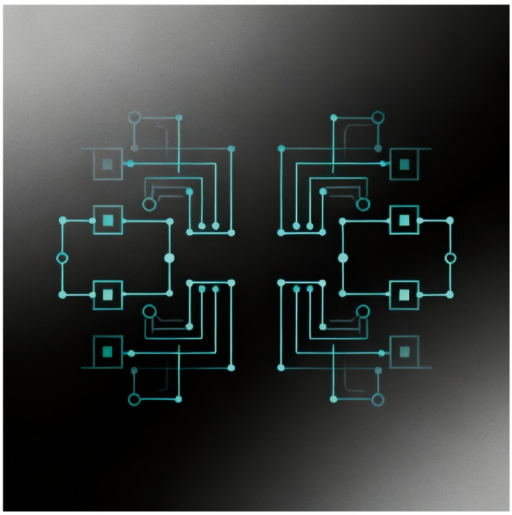
Logique formelle

Logique propositionnelle, logique des prédicats, logique modale et logiques non classiques



Lucien Sina

Logique formelle

Logique propositionnelle, logique des prédicats, logique modale et logiques non classiques

Lucien Sina

1.6.2026

Table des matières

Préface

La logique formelle constitue l'un des fondements de la pensée scientifique. Elle fournit un langage précis pour analyser les propositions, les concepts, les raisonnements et les preuves, et met ainsi à disposition des outils essentiels en mathématiques, en informatique, en philosophie, en linguistique et dans de nombreuses autres disciplines. Comprendre les relations logiques, c'est acquérir non seulement une compréhension plus profonde des systèmes formels, mais aussi de la structure des arguments et des limites du raisonnement lui-même.

Ce livre entend offrir un accès à la fois clair et systématique à la logique formelle. Il s'adresse aux lectrices et aux lecteurs qui souhaitent aborder les méthodes logiques dans leurs bases ou les approfondir, et les accompagne depuis les premiers concepts de la logique propositionnelle jusqu'à la logique des prédicats, aux logiques modales et à certaines logiques non classiques. Une attention particulière est accordée au fait de ne pas seulement introduire les différents thèmes de manière formelle, mais

aussi de rendre visibles les liens qui les unissent.

L'objectif principal n'est pas seulement l'apprentissage de définitions et de théorèmes, mais le développement d'une compréhension solide des méthodes logiques. C'est pourquoi des notions centrales telles que la syntaxe, la sémantique, le modèle, la conséquence logique, l'équivalence et la forme normale sont introduites avec soin, puis complétées par des exemples, des exercices et des solutions. Le parcours mène des fondements du raisonnement formel aux procédés algorithmiques tels que la résolution et les solveurs SAT, jusqu'à des thèmes plus avancés comme la théorie de Herbrand, la logique de description, la logique épistémique, la logique floue, la logique par défaut et la programmation par ensembles-réponses.

L'ouvrage adopte une approche didactique qui associe précision formelle et progression graduelle. De nombreuses sections sont conçues de manière à ne pas présenter seulement des résultats isolés, mais aussi à en faire apparaître la motivation, la signification et les applications. La logique ne doit ainsi pas apparaître comme une simple collection de règles séparées, mais comme un domaine cohérent, doté d'une structure interne claire et d'une grande portée pratique.

Mes remerciements vont à toutes celles et à tous ceux qui, par leurs suggestions, leurs retours et leurs remarques critiques, ont contribué à la réalisation de ce livre. Je remercie également les lectrices et les lecteurs qui acceptent de s'engager dans ce parcours à travers la logique

formelle, parfois exigeant, mais toujours enrichissant. Les retours, corrections et propositions d'amélioration sont à tout moment les bienvenus, car ils contribuent à affiner et à améliorer encore cet ouvrage.

Je vous souhaite une lecture stimulante et riche en découvertes.

Lucien Sina

0.1 Introduction et vue d'ensemble

La logique est la discipline qui étudie de manière systématique les formes du raisonnement. Elle s'appuie principalement sur des langages formels (syntaxe), qui fixent les règles de formation des expressions bien formées, et sur des sémantiques, qui attribuent une signification à cette syntaxe. Grâce à ces outils, les questions de vérité, de conséquence logique et de représentation des connaissances peuvent être formulées et analysées avec précision. Une logique fournit ainsi à la fois une description des expressions valides et un cadre permettant d'interpréter les énoncés par rapport à des structures, appelées modèles.

0.1.1 Motivation et pertinence

L'étude de la logique est essentielle pour plusieurs raisons. D'une part, elle constitue le fondement formel des mathématiques et de nombreux domaines de l'informatique. Des notions telles qu'algorithme, calculabilité et complexité sont historiquement étroitement liées à la logique formelle ; leurs concepts et leurs méthodes de preuve sont indispensables pour comprendre ce qui peut être résolu par des moyens automatiques. D'autre part, la logique possède de nombreuses applications directes : spécification et vérification de logiciels et de matériels, représentation des connaissances en intelligence artificielle, interrogation de bases de données, raisonnement automatique dans les démonstrateurs de théorèmes, ainsi

que modélisation dans les formalisations de nombreuses sciences.

0.1.2 Contexte historique

La logique formelle moderne trouve ses racines dans la philosophie et les mathématiques des XIXe et XXe siècles. Parmi les étapes marquantes figurent les travaux de Frege, Russell et Hilbert, qui ont fait progresser les systèmes formels et les présentations axiomatiques, ainsi que ceux de Turing, Church et Gödel, dont les résultats ont clarifié les rapports entre calculabilité et démontrabilité formelle. Ces développements ont donné naissance à des concepts centraux qui structurent encore aujourd'hui l'approche théorique et technique de la logique.

0.1.3 Objectifs de ce livre

Ce livre poursuit trois objectifs complémentaires :

1. offrir une introduction précise et autonome aux principales logiques (logique propositionnelle, logique des prédicats, logiques modales, logiques non classiques choisies et programmation par ensembles-réponses) ;
2. établir un pont avec l'informatique : les aspects algorithmiques, les questions de décidabilité et de complexité, ainsi que des procédures concrètes (par exemple les solveurs SAT, le model checking et la programmation par ensembles-réponses) y sont étudiés ;

3. développer des compétences méthodologiques : formuler correctement, raisonner par induction (complète et structurelle) et manier avec rigueur les notions de syntaxe et de sémantique.

0.1.4 Méthode et structure

Dans ce livre, nous suivons le schéma récurrent suivant :

— **Syntaxe** : nous définissons le langage formel au moyen de la signature, des termes et des formules ; nous précisons les règles de formation et les notations utilisées.

— **Sémantique** : nous décrivons la manière dont les formules sont interprétées dans des structures, puis nous définissons les notions de vérité, de satisfaisabilité et de validité.

— **Procédures de preuve et de décision** : pour chaque logique, nous examinons les systèmes de preuve usuels (axiomatiques, déduction naturelle, résolution, etc.) ainsi que les procédures algorithmiques et leurs limites.

— **Exemples et applications** : à la fin de chaque chapitre central, nous présentons des applications typiques et proposons des exercices.

0.1.5 À qui s'adresse ce livre ?

Ce livre s'adresse aux étudiantes et étudiants avancés de bachelor et de master en informatique, en mathématiques et dans des disciplines voisines, ainsi qu'aux praticiennes et praticiens qui souhaitent acquérir une compréhension solide et mathématiquement fondée de la logique. Des connaissances de base en théorie élémentaire des ensembles, en mathématiques discrètes et en techniques de preuve (en particulier l'induction complète) sont supposées ; lorsque cela est nécessaire, ces bases sont brièvement rappelées.

Ce que ce livre ne vise pas à faire

Cet ouvrage n'est ni un recueil complet d'exercices de programmation pour des outils logiques spécialisés, ni une histoire exhaustive de toutes les écoles de la logique. Il se concentre plutôt sur les concepts formels, les algorithmes typiques et le lien avec la pratique informatique. Les aspects spécialisés (par exemple l'implémentation détaillée de solveurs SMT ou SAT) ne sont abordés que dans la mesure où ils servent à illustrer des énoncés théoriques.

0.1.6 Objectifs d'apprentissage et conseils d'utilisation

Après la lecture des six premiers chapitres, les lectrices et les lecteurs devraient être capables de :

— définir des langages formels et décrire leur sémantique dans des structures ;

— appliquer les techniques de preuve typiques de la logique (induction complète et structurelle, preuve par contradiction, construction directe) ;

— comprendre dans leurs grandes lignes des procédures algorithmiques telles que la résolution SAT et le model checking, ainsi que traiter manuellement des cas simples ;

— reconnaître les limites des systèmes formels (indécidabilité, classification de complexité) et en tirer des conséquences pour l'informatique.

Remarques sur la notation

Nous nous efforçons d'employer une notation cohérente ; les conventions centrales sont les suivantes :

— $\mathbb{N}, \mathbb{Z}, \mathbb{R}$ désignent les ensembles de nombres usuels ;

— les variables propositionnelles ou de formules sont généralement notées p, q, r, φ, ψ, les individus par x, y, z ;

— pour les opérations ensemblistes, nous utilisons $\cup, \cap, \setminus, \times$;

— les quantificateurs apparaissent sous la forme $\forall x$ ou $\exists x$.

Suite du parcours

Nous commencerons par une introduction concise, mais complète, aux notations utilisées, puis nous rappellerons les notions nécessaires de théorie des ensembles et d'induction. Nous aborderons ensuite la logique propositionnelle et progresserons pas à pas vers les thèmes plus avancés.

Les pages qui suivent proposent de brèves présentations de chaque chapitre. Elles indiquent les contenus abordés ainsi que les compétences que les différentes sections visent à développer.

0.1.7 Brèves descriptions des chapitres

Les paragraphes suivants offrent une orientation synthétique sur les thèmes traités dans chaque chapitre et sur les acquis attendus de la lectrice ou du lecteur.

Chapitre 0 – Introduction et fondements. Ce chapitre fixe les bases notationnelles et conceptuelles centrales : théorie des ensembles et des relations, fonctions, principes d'induction, construction des langages formels ainsi que notions fondamentales de sémantique (modèles, satisfaisabilité, validité, conséquence logique). Il sert de référence pour les chapitres suivants.

Chapitre 1 – Logique propositionnelle. La logique propositionnelle constitue l'entrée dans le raisonnement

formel : nous introduisons les variables propositionnelles et les connecteurs (« et », « ou », « non », « implique »), définissons l'évaluation sémantique à l'aide de tables de vérité et étudions les formes normales syntaxiques (FNC, FND). Les principaux thèmes sont les procédures de preuve (notamment la résolution et la déduction naturelle), la satisfaisabilité (SAT) et les premières notions de complexité.

Chapitre 2 – Logique des prédicats. La logique des prédicats étend la logique propositionnelle au moyen de quantificateurs et de prédicats, ce qui permet d'exprimer des relations entre individus. Nous étudions les signatures, les structures, l'interprétation des termes et des formules, ainsi que des résultats métathéoriques centraux (complétude, incomplétude, théorème de Löwenheim–Skolem, questions de décidabilité) et leurs conséquences pour l'informatique, notamment les limites de l'automatisation.

Chapitre 3 – Logiques modales (y compris la logique temporelle). Les logiques modales permettent de qualifier les énoncés au moyen d'opérateurs tels que « possible », « nécessaire », « toujours » ou « un jour ». Nous introduisons la sémantique de Kripke, présentons des systèmes modaux classiques (K, T, S4, S5) et décrivons des instances particulières comme les logiques temporelles (LTL, CTL), ainsi que leurs applications au model checking et à la spécification de systèmes.

Chapitre 4 – Autres logiques non classiques et programmation par ensembles-réponses. Nous considérons ici une sélection de logiques non classiques qui constituent des extensions ou des alternatives typiques à la logique classique : logique intuitionniste, logique paraconsistante, logiques multivaluées et logique floue, logiques par défaut et logiques probabilistes. Nous présentons aussi la programmation par ensembles-réponses (Answer Set Programming, ASP) comme une technique pratique de représentation des connaissances et d'inférence non monotone.

Chapitre 5 – Synthèse et perspectives. Le dernier chapitre récapitule les résultats centraux, met en évidence les liens entre les domaines étudiés et ouvre brièvement sur le raisonnement automatique, les thèmes de recherche actuels et les applications avancées en informatique, en intelligence artificielle et en mathématiques.

Les notions importantes qui reviennent tout au long de cet ouvrage sont les suivantes :

— **Formule** – une expression syntaxiquement bien formée d'un langage logique.

— **Structure ou modèle** – une interprétation qui attribue des valeurs aux symboles du langage.

— **Satisfaisabilité** – l'existence d'un modèle qui rend une formule vraie.

— **Conséquence logique** – relation selon laquelle une formule découle logiquement d'un ensemble de

formules.

— **Vérification de modèles** – procédure algorithmique permettant de déterminer si une structure est un modèle d'une formule.

Cet ouvrage vise à proposer une introduction précise et orientée vers les applications aux principales logiques. Nous accordons une attention particulière à la rigueur formelle, aux exemples motivants ainsi qu'aux aspects algorithmiques, notamment la calculabilité et la complexité.

0.2 Fondements

0.2.1 Notation et théorie des ensembles

Nous utilisons la notation usuelle de la théorie des ensembles. Si X est un ensemble et si x est un objet, alors $x \in X$ signifie que x est un élément de X. Pour deux ensembles X et Y, nous écrivons $X \subseteq Y$ lorsque X est un sous-ensemble de Y.

Les ensembles des nombres naturels, des nombres entiers et des nombres réels sont notés respectivement $\mathbb{N}$, $\mathbb{Z}$ et $\mathbb{R}$.

Le produit cartésien de deux ensembles X et Y est noté $X \times Y$. L'ensemble des parties d'un ensemble X est noté $\mathcal{P}(X)$ ou 2^X.

Ensembles, opérations et relations

Les ensembles peuvent être décrits implicitement par une règle de construction. Par exemple,

$$D \;=\; \left\{ x \in \mathbb{N} \;\middle|\; \begin{array}{r} \exists y \in \mathbb{N} : y \mid x \\ \text{et } 1 < y < x \end{array} \right\}$$

est l'ensemble de tous les nombres naturels composés.

Parmi les opérations ensemblistes importantes figurent l'union $\cup$, l'intersection $\cap$ et la différence $\setminus$. Pour deux ensembles X, X', on a :

$$\begin{aligned} X \cup X' &= \{\, x \mid x \in X \text{ ou } x \in X' \,\}, \\ X \cap X' &= \{\, x \mid x \in X \text{ et } x \in X' \,\}, \\ X \setminus X' &= \{\, x \mid x \in X \text{ et } x \notin X' \,\}. \end{aligned}$$

Le produit cartésien de X et de X' est

$$X \times X' \;=\; \{\, (x, x') \mid x \in X,\; x' \in X' \,\}.$$

Plus généralement, nous écrivons $X^n = X \times \cdots \times X$ (n fois) pour le produit cartésien n-uple.

Une relation (binaire) R entre les ensembles X et X' est un sous-ensemble $R \subseteq X \times X'$. Au lieu de $(x, x') \in R$, nous écrivons parfois aussi xRx'. Une relation n-aire sur X est un sous-ensemble de X^n.

Si $R \subseteq X \times X$ (donc si $X' = X$), nous définissons :

— R est *réflexive* ssi $(x, x) \in R$ pour tout $x \in X$.

— R est *symétrique* ssi $(x, y) \in R$ implique aussi $(y, x) \in R$.

— R est *transitive* ssi $(x, y) \in R$ et $(y, z) \in R$ impliquent aussi $(x, z) \in R$.

Si R est réflexive, symétrique et transitive, alors R est appelée une *relation d'équivalence*.

Une fonction f est une application $f : X \to Y$. Pour $x \in X$, nous notons $f(x)$ l'image de x. Toute fonction peut aussi être considérée comme une relation particulière $f \subseteq X \times Y$ satisfaisant la propriété de fonctionnalité : si $(x, y) \in f$, alors il n'existe aucun $y' \in Y$ tel que $y' \neq y$ et $(x, y') \in f$.

Soit $f : X \to Y$.

— f est *injective* (une application un-à-un) si, pour tous $x, x' \in X$, l'égalité $f(x) = f(x')$ entraîne toujours $x = x'$.

— f est *surjective* si, pour tout $y \in Y$, il existe un $x \in X$ tel que $f(x) = y$.

— f est *bijective* si elle est à la fois injective et surjective.

L'image réciproque f^{-1} est généralement définie comme une application vers l'ensemble des parties :

$$f^{-1} : Y \to \mathcal{P}(X), \qquad f^{-1}(y) = \{\, x \in X \mid f(x) = y \,\}.$$

Si f est une bijection, alors f^{-1} est une application $f^{-1} : Y \to X$, et pour chaque $y \in Y$, $f^{-1}(y)$ est l'unique $x \in X$ tel que $f(x) = y$.

Si $f : X \to Y$ et $g : Y \to Z$ sont des applications, alors $g \circ f : X \to Z$ désigne leur composition, définie par $(g \circ f)(x) = g(f(x))$ pour tout $x \in X$.

Notation de croissance. Pour des fonctions $f, g : \mathbb{N} \to \mathbb{N}$, on écrit $g \in O(f)$ s'il existe des constantes $c > 0$ et $n_0 \in \mathbb{N}$ telles que

$$\forall n \geq n_0 : \quad g(n) \leq c \cdot f(n).$$

Intuitivement, $g \in O(f)$ signifie que g ne croît pas asymptotiquement plus vite que f.

0.2.2 Structures et sémantique

Une structure $\mathcal{A}$ pour une signature logique se compose d'un ensemble de base A et d'une interprétation des symboles de la signature. Une formule φ est vraie dans une structure $\mathcal{A}$ lorsque $\mathcal{A} \models \varphi$.

Les notions importantes sont les suivantes :

— Satisfaisabilité : il existe une structure $\mathcal{A}$ telle que $\mathcal{A} \models \varphi$.

— Validité : pour toute structure $\mathcal{A}$, on a $\mathcal{A} \models \varphi$.

— Conséquence logique : $\Phi \models \psi$.

0.2.3 Principes d'induction

Les preuves par induction constituent des techniques de démonstration fondamentales en logique et en informatique théorique. Nous distinguons ici deux formes courantes : l'*induction complète* sur les nombres naturels et l'*induction structurelle* sur des objets définis syntaxiquement (termes, formules, arbres). Ces deux méthodes reposent sur l'idée de structures bien ordonnées ou bien

fondées, mais elles diffèrent par la formulation de l'étape d'induction.

Induction complète

L'induction complète (également appelée induction simple ou forte, selon la formulation) concerne des énoncés portant sur les nombres naturels. Pour démontrer une assertion $A(n)$ pour tout $n \in \mathbb{N}$, on établit :

1. (*initialisation*) $A(1)$ est vraie (ou $A(0)$, selon la convention choisie).
2. (*étape d'induction*) Pour un $n > 1$ arbitraire, l'hypothèse selon laquelle $A(k)$ vaut pour tout $k < n$ entraîne que $A(n)$ vaut aussi.

Le principe d'induction complète découle alors du fait qu'il ne peut pas exister de plus petit contre-exemple.

Exemple 0.1. *(Formule de sommation de Gauss) Pour tout $n \in \mathbb{N}$, on a*

$$\sum_{i=1}^{n} i \;=\; \frac{n(n+1)}{2}.$$

Démonstration. Initialisation : Pour $n = 1$, on a $\sum_{i=1}^{1} i = 1$ et $\frac{1\cdot(1+1)}{2} = 1$. L'égalité est donc vraie pour $n = 1$.

Étape d'induction : Soit $n > 1$ et supposons que la formule soit vraie pour $n - 1$, c'est-à-dire

$$\sum_{i=1}^{n-1} i = \frac{(n-1)n}{2}.$$

On obtient alors, pour n :

$$\begin{aligned}\sum_{i=1}^{n} i &= \Big(\sum_{i=1}^{n-1} i\Big) + n \\ &= \frac{(n-1)n}{2} + n \\ &= \frac{n(n-1) + 2n}{2} \\ &= \frac{n(n+1)}{2}.\end{aligned}$$

L'étape d'induction est ainsi établie, et l'énoncé est démontré pour tout n. □

Induction structurelle

L'induction structurelle convient aux énoncés portant sur des objets définis récursivement ou de forme arborescente, tels que les termes ou les formules. On prend comme guide la définition récursive du type d'objet considéré : on montre d'abord la propriété pour les objets élémentaires (atomiques), puis on montre qu'elle est préservée lors de la construction d'objets composés, conformément aux règles de formation.

Formellement, soit $\mathcal{F}$ l'ensemble des formules d'un langage, défini par les règles

$$\varphi ::= p \mid \neg\varphi \mid (\varphi \wedge \psi) \mid (\varphi \vee \psi)$$

où p désigne une formule atomique. Pour démontrer une propriété $P(\varphi)$ pour toute formule $\varphi \in \mathcal{F}$, on établit :

— $P(p)$ pour toute formule atomique p.

— De $P(\varphi)$, on déduit $P(\neg\varphi)$.

— De $P(\varphi)$ et $P(\psi)$, on déduit $P((\varphi \wedge \psi))$ (et de même pour les autres opérateurs binaires).

Exemple 0.2. *Toute formule $\varphi \in \mathcal{F}$ ne possède qu'un nombre fini de sous-formules.*

Démonstration. Nous démontrons, par induction structurelle, l'énoncé *« φ possède un nombre fini de sous-formules »* pour toute formule $\varphi \in \mathcal{F}$.

Base : soit $\varphi = p$ une formule atomique. Alors l'ensemble de ses sous-formules est $\{\ p\ \}$, donc il est fini.

Étape d'induction : supposons que les ensembles des sous-formules de φ et de ψ soient finis. Les sous-formules de $\neg\varphi$ sont alors données par $\{\neg\varphi\} \cup \{\text{sous-formules de } \varphi\}$; il s'agit donc d'une union finie d'ensembles finis, et cet ensemble est fini. De même, pour la formule composée $(\varphi \wedge \psi)$, l'ensemble des sous-formules est

$$\{\ (\varphi \wedge \psi)\ \} \cup \{\text{sous-formules de } \varphi\} \cup \{\text{sous-formules de } \psi\}$$

une union finie d'ensembles finis ; il est donc lui aussi fini.

Par induction structurelle, toute formule $\varphi \in \mathcal{F}$ possède donc seulement un nombre fini de sous-formules. □

Comparaison, remarques et cas d'application typiques

— **Quelle induction choisir ?** On utilise l'induction complète lorsque l'énoncé est formulé à propos de nombres naturels (ou directement sur $\mathbb{N}$), par exemple en combinatoire, pour des sommes ou pour des suites récursives. On utilise l'induction structurelle lorsque l'énoncé porte sur des objets récursivement définis et arborescents, comme des termes, des formules, des arbres de preuve ou des arbres de syntaxe abstraite.

— **Forme de l'étape d'induction.** Dans l'induction complète, l'étape d'induction se rapporte typiquement au prédécesseur immédiat $n - 1$, ou bien à tous les $k < n$ dans le cas de l'induction forte. Dans l'induction structurelle, on montre que la propriété est conservée par les règles de construction.

— **Points communs.** Les deux méthodes sont des applications du principe selon lequel il n'existe pas de suites infinies strictement descendantes dans une relation bien fondée. Formellement, on peut les considérer comme des cas particuliers de l'induction bien fondée (well-founded induction).

— **Conseils pratiques.** Formule les énoncés d'induction de telle sorte que l'hypothèse d'induction contienne exactement les informations nécessaires. En induction structurelle, il est souvent utile de choisir des énoncés d'induction assez forts (par

exemple des énoncés portant sur l'ensemble de toutes les sous-structures), afin que les cas composés se traitent plus facilement.

0.3 Structure et style des textes mathématiques

Les textes mathématiques de ce livre suivent une forme claire et reconnaissable : les définitions et les notations fixent des concepts et des conventions avec précision, les exemples éclairent l'intuition, les résultats (lemmes, propositions, théorèmes, corollaires) formulent les énoncés centraux, et les preuves les justifient de manière rigoureuse. Cette structure aide la lectrice ou le lecteur à distinguer rapidement l'énoncé formel, la preuve et le matériau illustratif.

Définitions et notations

— **Définitions.** Les définitions formalisent de nouveaux concepts. Elles doivent être concises, complètes et choisies de telle sorte que les propriétés nécessaires puissent être vérifiées directement.

— **Notations.** Les notations regroupent des écritures abrégées (par exemple $\mathcal{P}(X)$, $X \times Y$, $O(\cdot)$). Elles doivent être utilisées de façon cohérente et rassemblées à un endroit bien visible (au début d'un chapitre ou dans un encadré de notation).

Exemples et illustrations

Les exemples servent à rendre les notions plus concrètes et à montrer des cas typiques ou, au contraire, des cas limites. Ils doivent rester brefs, illustrer l'usage des nouvelles définitions et, lorsque c'est possible, renvoyer à des exercices ou à des applications ultérieures.

Résultats : lemme, proposition, théorème, corollaire

— **Lemme.** Un résultat auxiliaire utilisé principalement dans la preuve d'un autre résultat.

— **Proposition.** Un résultat utile, généralement moins central qu'un théorème.

— **Théorème.** Un résultat central et important.

— **Corollaire.** Un énoncé qui découle immédiatement d'un résultat précédent.

Tout résultat doit comporter une formulation précise, les hypothèses pertinentes lorsqu'elles sont nécessaires, ainsi qu'une preuve claire.

Preuves

Les preuves font partie de l'argumentation mathématique usuelle. Elles doivent, autant que possible, être structurées :

1. une brève indication initiale de ce qui doit être démontré ;

2. si nécessaire, une division en cas ou en lemmes auxiliaires ;
3. à la fin, un signe de clôture explicite, le plus souvent noté □.

Remarques, exercices et indications bibliographiques

— **Remarques.** Elles contiennent souvent des informations complémentaires, de l'intuition ou des généralisations. Elles sont plus courtes et moins formelles.

— **Exercices.** Placés à la fin d'un chapitre, ils consolident la compréhension.

— **Indications bibliographiques.** Elles renvoient à des sources complémentaires et doivent être bibliographiées avec précision.

Bref exemple modèle

Définition 0.3. *Une application $f : X \to Y$ est dite* injective *si, de $f(x) = f(x')$, il s'ensuit toujours que $x = x'$.*

Théorème 0.4. *Soit $f : X \to Y$ une application injective et soit $g : Y \to Z$ une application quelconque. Alors $g \circ f$ est injective si et seulement si g est injective sur l'image de f.*

Démonstration. L'énoncé découle d'une vérification directe de la définition : si $g \circ f$ est injective et si $g(f(x)) =$

$g(f(x'))$, alors $f(x) = f(x')$; comme f est injective, il s'ensuit que $x = x'$. La réciproque est tout aussi directe. □

Chapitre 1

Logique propositionnelle

1.1 Introduction et vue d'ensemble

La logique propositionnelle est le système formel fondamental qui permet de décrire avec précision des énoncés simples ainsi que leurs connexions logiques. Elle repose sur des propositions atomiques, dont chacune reçoit exactement l'une des deux valeurs de vérité possibles : vrai ou faux. À partir de ces éléments de base, des formules composées sont construites au moyen de connecteurs tels que *et*, *ou* et *non* ; elles permettent d'exprimer des relations logiques plus complexes.

Malgré sa syntaxe élémentaire, la logique propositionnelle possède une portée considérable. Elle constitue une base importante pour de nombreux autres domaines

de la logique et joue en même temps un rôle central en informatique, par exemple dans la démonstration automatique, l'analyse des circuits logiques, la vérification formelle ou les méthodes de l'intelligence artificielle. Son intérêt tient notamment au fait qu'elle est entièrement formalisée : aussi bien la formation des formules que leur signification peuvent être définies de manière exacte.

Ce chapitre introduit d'abord les fondements syntaxiques et sémantiques. Nous commencerons par les signatures, les formules et les interprétations, puis nous préciserons quand une formule est vraie sous une interprétation donnée et ce que signifie, pour un ensemble de formules, le fait de posséder un modèle. Nous étudierons ensuite la notion de conséquence logique, c'est-à-dire la question de savoir quand une formule découle d'une autre formule ou d'un ensemble de formules. Sur cette base, nous examinerons l'équivalence logique des formules et présenterons des règles de transformation ainsi que des formes normales, particulièrement importantes pour le traitement algorithmique.

Dans la suite du chapitre, nous nous tournerons vers des méthodes qui ne décrivent pas seulement le raisonnement logique de manière sémantique, mais l'exécutent aussi de façon syntaxique. Il s'agit notamment de différents calculs formels ainsi que du calcul de résolution, qui constitue un outil central de la démonstration automatique. Enfin, nous examinerons de plus près l'aspect algorithmique de la logique propositionnelle : nous étudierons en particulier le problème de la satisfaisabilité,

expliquerons pourquoi il est difficile d'un point de vue théorique, et présenterons le codage de Tseitin ainsi que l'algorithme DPLL, deux méthodes importantes permettant de tester efficacement la satisfaisabilité dans de nombreux cas pratiques.

Le chapitre est structuré comme suit : dans la sous-section 1.1, nous introduisons la syntaxe et la sémantique de la logique propositionnelle. Nous y traiterons d'abord les signatures, puis les formules, les interprétations et les modèles. La sous-section 1.2 porte sur les propriétés des modèles, la notion de conséquence et l'indépendance de la signature. La sous-section 1.3 est consacrée à l'équivalence ; elle comprend aussi bien la définition de l'équivalence logique que les principales transformations équivalentes et la représentation des propositions sous forme normale. Dans la sous-section 1.4, plusieurs calculs destinés à la déduction automatique seront présentés, en particulier le calcul de résolution. Enfin, la sous-section 1.5 expliquera la réduction aux problèmes de satisfaisabilité, le codage de Tseitin et l'algorithme DPLL comme méthodes centrales de résolution du problème de satisfaisabilité.

Ce chapitre relie ainsi les bases conceptuelles de la logique propositionnelle à ses principales procédures algorithmiques. Il fournit la base nécessaire à tout ce qui sera ensuite consacré au raisonnement formel, aux formes normales et aux méthodes de démonstration automatique.

1.2 Syntaxe et sémantique

1.2.1 Introduction

Comme toute logique, la logique propositionnelle est définie par deux composantes :

— *Syntaxe :* la description des expressions bien formées (formules), c'est-à-dire des suites de symboles qui sont admises comme objets valides, et

— *Sémantique :* l'attribution de significations (valeurs de vérité) à ces expressions.

Dans ce qui suit, nous introduisons d'abord les signatures et les formules, puis nous définissons la sémantique au sens de la théorie des modèles.

1.2.2 Signatures

Définition 1.1. *Une* signature propositionnelle Σ *est un ensemble (fini) de variables propositionnelles, également appelées atomes ou propositions.*

Exemple 1.2. *Pour modéliser un système d'alarme domestique intelligent, on peut choisir par exemple*

$$\Sigma_{Alarm} = \{Smoke,\ Heat,\ Motion,\ Alarm,\ Sprinkler\},$$

où les significations visées des variables sont par exemple les suivantes :

— *Smoke : de la fumée est détectée*
— *Heat : une température élevée est détectée*

— *Motion : un mouvement est enregistré dans la maison*
— *Alarm : l'alarme est déclenchée*
— *Sprinkler : les sprinklers sont activés*

Une formule typique de $L(\Sigma_{Alarm})$ *serait par exemple*

$$(Smoke \vee Heat) \rightarrow Alarm,$$

qui signifie : « Si de la fumée ou une température élevée est détectée, alors l'alarme est déclenchée ». ■

1.2.3 Formules

Définition 1.3. *Soit* Σ *une signature propositionnelle. L'ensemble* $L(\Sigma)$ *des* formules *sur* Σ *est défini inductivement comme suit :*

1. *toute variable propositionnelle* $p \in \Sigma$ *est une formule (formule atomique) ;*
2. *si* φ *est une formule, alors* $\neg\varphi$ *est une formule ;*
3. *si* φ *et* ψ *sont des formules, alors* $(\varphi \wedge \psi)$ *et* $(\varphi \vee \psi)$ *sont des formules.*

De manière équivalente (BNF) :

$$\varphi ::= p \mid \neg\varphi \mid (\varphi \wedge \varphi) \mid (\varphi \vee \varphi) \qquad (p \in \Sigma).$$

Notation 1.4. *Nous utiliserons les abréviations et conventions suivantes :*

— *Connecteurs :* $\neg$ *(négation),* $\wedge$ *(conjonction),* $\vee$ *(disjonction).*

— *L'implication et l'équivalence sont définies par* $\varphi \to \psi := \neg\varphi \vee \psi$ *et* $\varphi \leftrightarrow \psi := (\varphi \to \psi) \wedge (\psi \to \varphi)$.

— *Convention de parenthésage :* $\neg$ *lie plus fortement que* $\wedge$*, qui lie plus fortement que* $\vee$*, qui lie plus fortement que* $\to$*, qui lie plus fortement que* $\leftrightarrow$.

— *Un* littéral *est soit un atome* p*, soit sa négation* $\neg p$.

Exemple 1.5. *Avec* Σ_{Alarm} *de l'exemple précédent, les expressions suivantes sont par exemple des formules de* $L(\Sigma_{Alarm})$ *:*

$$(Smoke \vee Heat) \to Alarm,$$
$$\neg Smoke \wedge (Alarm \vee Sprinkler),$$
$$Motion \to Alarm.$$

L'ensemble des littéraux est

$$\begin{aligned}\mathrm{Lit}(\Sigma_{Alarm}) = \{&Smoke, \neg Smoke,\ Heat, \neg Heat,\\ &Motion, \neg Motion,\ Alarm, \neg Alarm,\\ &Sprinkler, \neg Sprinkler\}.\end{aligned}$$

■

1.2.4 Sémantique : interprétations, satisfaction et modèles

Définition 1.6. *Soit* Σ *une signature. Une* interprétation *(ou valuation)* ω *sur* Σ *est une fonction* $\omega : \Sigma \to \{true, false\}$*. Nous appelons* $\Omega(\Sigma)$ *l'ensemble de toutes les interprétations.*

Définition 1.7. *La relation de satisfaction* $\models_{\mathrm{LP}} \subseteq \Omega(\Sigma) \times L(\Sigma)$ *est définie inductivement :*

$$\begin{aligned} \omega \models_{\mathrm{LP}} p \quad & \textit{ssi} \quad \omega(p) = \textit{true}, \\ \omega \models_{\mathrm{LP}} \neg\varphi \quad & \textit{ssi} \quad \textit{non } (\omega \models_{\mathrm{LP}} \varphi), \\ \omega \models_{\mathrm{LP}} \varphi \wedge \psi \quad & \textit{ssi} \quad \omega \models_{\mathrm{LP}} \varphi \textit{ et } \omega \models_{\mathrm{LP}} \psi, \\ \omega \models_{\mathrm{LP}} \varphi \vee \psi \quad & \textit{ssi} \quad \omega \models_{\mathrm{LP}} \varphi \textit{ ou } \omega \models_{\mathrm{LP}} \psi. \end{aligned}$$

Si $\omega \models_{\mathrm{LP}} \varphi$*, alors* ω *est appelé un* modèle *de* φ*.*

Définition 1.8. *Pour une formule* φ*, nous définissons*

$$\mathrm{Mod}_{\mathrm{LP}}(\varphi) \;=\; \{\, \omega \in \Omega(\Sigma) \mid \omega \models_{\mathrm{LP}} \varphi \,\},$$

l'ensemble de tous les modèles de φ*. De plus,* φ *est* satisfaisable *ssi* $\mathrm{Mod}_{\mathrm{LP}}(\varphi) \neq \varnothing$*, et* φ *est* valide *(une tautologie) ssi* $\mathrm{Mod}_{\mathrm{LP}}(\varphi) = \Omega(\Sigma)$*.*

Explication. Si φ est une formule et si ω est un modèle de φ ($\omega \in \mathrm{Mod}_{\mathrm{LP}}(\varphi)$), on peut comprendre ω, de manière intuitive, comme un *monde possible* ou comme une *situation* dans laquelle l'énoncé φ est effectivement vrai. L'interprétation ω détermine, pour chaque énoncé atomique $p \in \Sigma$, si p est vrai ou faux dans ce monde ; elle explique ainsi *complètement* pourquoi φ est vraie, au moyen de l'attribution de valeurs de vérité aux atomes dont φ est composée.

Le terme *modèle* souligne précisément ce rôle explicatif : un modèle est un témoin concret montrant qu'une formule est satisfaisable. Par conséquent, l'ensemble $\Omega(\Sigma)$

est l'ensemble de tous les mondes possibles (c'est-à-dire de toutes les valuations possibles), et $\mathrm{Mod}_{\mathrm{LP}}(\varphi)$ est constitué des mondes qui rendent φ vraie.

Deux notions brèves et utiles :

— *Satisfaisabilité :* La formule φ est satisfaisable exactement lorsque $\mathrm{Mod}_{\mathrm{LP}}(\varphi) \neq \varnothing$. Un $\omega \in \mathrm{Mod}_{\mathrm{LP}}(\varphi)$ concret est aussi appelé *modèle* ou *valuation satisfaisante* de φ.

— *Contre-modèle :* Pour montrer qu'une formule φ n'est *pas* valide (n'est pas tautologique), il suffit d'indiquer un $\omega \in \Omega(\Sigma)$ tel que $\omega \notin \mathrm{Mod}_{\mathrm{LP}}(\varphi)$; un tel ω est appelé *contre-modèle*.

Remarque. Dans les applications (par exemple en vérification de modèles), trouver un modèle revient souvent à montrer qu'un comportement souhaité du système peut se produire, tandis que fournir un contre-modèle montre qu'une propriété souhaitée est violée sous certaines valuations.

Exemple 1.9. *Considérons* Σ_{Alarme} *et l'interprétation* ω *telle que* $\omega(Fumée) = vrai$, $\omega(Chaleur) = faux$, $\omega(Mouvement) = vrai$, $\omega(Alarme) = vrai$, $\omega(Arroseur) = faux$. *Alors on a par exemple* $\omega \models_{\mathrm{LP}} Fumée$, $\omega \models_{\mathrm{LP}} \neg Chaleur$, $\omega \models_{\mathrm{LP}} (Fumée \vee Chaleur) \rightarrow Alarme$ *(car* $\omega(Fumée) = vrai$ *et* $\omega(Alarme) = vrai$*),* $\omega \models_{\mathrm{LP}} Mouvement \rightarrow Alarme$ *(puisque* $\omega(Mouvement) = vrai$ *et* $\omega(Alarme) = vrai$*) ainsi que* $\omega \models_{\mathrm{LP}} \neg Arroseur$. ■

1.2.5 Tables de vérité

Les connecteurs peuvent être décrits au moyen de tables de vérité ; voici une présentation compacte (1 = vrai, 0 = faux) :

φ	ψ	$\varphi \wedge \psi$	$\varphi \vee \psi$	$\neg\varphi$
0	0	0	0	1
0	1	0	1	1
1	0	0	1	0
1	1	1	1	0

1.2.6 Remarques

— Si $|\Sigma| = n$, alors $|\Omega(\Sigma)| = 2^n$ interprétations.

— L'induction définitoire de la sémantique (satisfaction) est équivalente à la donnée directe d'une fonction de vérité $[\![\cdot]\!]_\omega$.

— La sémantique modèle-théorique constitue le fondement de notions comme la *conséquence logique* $\Phi \models \psi$ (toute interprétation qui satisfait toutes les formules de Φ satisfait aussi ψ).

1.2.7 Fonctions de vérité

Définition 1.10. *Soient Σ une signature propositionnelle et $\omega \in \Omega(\Sigma)$ une interprétation. La* fonction de vérité $[\![\cdot]\!]_\omega : L(\Sigma) \to \{\text{true}, \text{false}\}$ *est définie inductivement comme*

suit :

$$[\![p]\!]_\omega = \omega(p) \quad (p \in \Sigma),$$

$$[\![\neg\varphi]\!]_\omega = \begin{cases} \text{true}, & \textit{si } [\![\varphi]\!]_\omega = \text{false}, \\ \text{false}, & \textit{sinon}, \end{cases}$$

$$[\![\varphi \wedge \psi]\!]_\omega = \begin{cases} \text{true}, & \textit{si } [\![\varphi]\!]_\omega = \text{true } \textit{et } [\![\psi]\!]_\omega = \text{true}, \\ \text{false}, & \textit{sinon}, \end{cases}$$

$$[\![\varphi \vee \psi]\!]_\omega = \begin{cases} \text{false}, & \textit{si } [\![\varphi]\!]_\omega = \text{false } \textit{et } [\![\psi]\!]_\omega = \text{false}, \\ \text{true}, & \textit{sinon.} \end{cases}$$

Proposition 1.11. *Soient Σ une signature, $\omega \in \Omega(\Sigma)$ une interprétation et $\varphi \in L(\Sigma)$. Alors on a*

$$\omega \models_{\text{LP}} \varphi \quad \textit{ssi} \quad [\![\varphi]\!]_\omega = \text{true}.$$

Démonstration. Soient Σ une signature propositionnelle, $\omega \in \Omega(\Sigma)$ une interprétation et $\varphi \in L(\Sigma)$ une formule. Nous montrons l'équivalence

$$\omega \models_{\text{LP}} \varphi \quad \text{ssi} \quad [\![\varphi]\!]_\omega = \text{true}$$

par induction structurelle sur la formule φ.

Initialisation. Soit $\varphi = p$, où $p \in \Sigma$ est un atome. Par définition de la relation de satisfaction, $\omega \models_{\text{LP}} p$ vaut exactement lorsque $\omega(p) = \text{true}$. Par définition de la fonction de vérité, on a $[\![p]\!]_\omega = \omega(p)$. Il s'ensuit immédiatement que $\omega \models_{\text{LP}} p \iff [\![p]\!]_\omega = \text{true}$.

Hérédité. Soit maintenant φ une formule, et supposons que l'énoncé soit déjà établi pour toutes les sous-formules propres de φ (hypothèse d'induction). Nous distinguons les différents cas de construction.

— $\varphi = \neg\psi$.
$\Rightarrow$: Supposons que $\omega \models_{\mathrm{LP}} \neg\psi$. Alors, par définition, $\omega \not\models_{\mathrm{LP}} \psi$. D'après l'hypothèse d'induction, il vient $[\![\psi]\!]_\omega \neq \text{true}$, donc $[\![\psi]\!]_\omega = \text{false}$. Par définition de la fonction de vérité, on obtient alors $[\![\neg\psi]\!]_\omega = \text{true}$.
$\Leftarrow$: Supposons que $[\![\neg\psi]\!]_\omega = \text{true}$. Alors $[\![\psi]\!]_\omega = \text{false}$. Par l'hypothèse d'induction, on a $\omega \not\models_{\mathrm{LP}} \psi$, et donc $\omega \models_{\mathrm{LP}} \neg\psi$.

— $\varphi = \psi \wedge \xi$.
$\Rightarrow$: Supposons que $\omega \models_{\mathrm{LP}} \psi \wedge \xi$. Alors $\omega \models_{\mathrm{LP}} \psi$ et $\omega \models_{\mathrm{LP}} \xi$. D'après l'hypothèse d'induction, on a donc $[\![\psi]\!]_\omega = \text{true}$ et $[\![\xi]\!]_\omega = \text{true}$. Ainsi, $[\![\psi \wedge \xi]\!]_\omega = \text{true}$.
$\Leftarrow$: Supposons que $[\![\psi \wedge \xi]\!]_\omega = \text{true}$. Alors $[\![\psi]\!]_\omega = \text{true}$ et $[\![\xi]\!]_\omega = \text{true}$. Par l'hypothèse d'induction, il s'ensuit que $\omega \models_{\mathrm{LP}} \psi$ et $\omega \models_{\mathrm{LP}} \xi$, donc $\omega \models_{\mathrm{LP}} \psi \wedge \xi$.

— $\varphi = \psi \vee \xi$.
$\Rightarrow$: Supposons que $\omega \models_{\mathrm{LP}} \psi \vee \xi$. Alors $\omega \models_{\mathrm{LP}} \psi$ ou $\omega \models_{\mathrm{LP}} \xi$. Supposons sans perte de généralité que $\omega \models_{\mathrm{LP}} \psi$. D'après l'hypothèse d'induction, $[\![\psi]\!]_\omega = \text{true}$, de sorte que $[\![\psi \vee \xi]\!]_\omega = \text{true}$. L'autre cas est analogue.
$\Leftarrow$: Supposons que $[\![\psi \vee \xi]\!]_\omega = \text{true}$. Alors $[\![\psi]\!]_\omega = \text{true}$ ou $[\![\xi]\!]_\omega = \text{true}$. Par l'hypothèse d'induction, il s'ensuit que $\omega \models_{\mathrm{LP}} \psi$ ou $\omega \models_{\mathrm{LP}} \xi$, et donc

$\omega \models_{\mathrm{LP}} \psi \vee \xi$.

Tous les cas de construction ayant été traités, l'assertion vaut pour toute formule φ. □

Les connecteurs peuvent également être présentés de manière compacte au moyen de tables de vérité. Commençons par la variante utilisant les désignations true/false :

φ	ψ	$\varphi \wedge \psi$	$\varphi \vee \psi$	$\neg\varphi$
false	false	false	false	true
false	true	false	true	true
true	false	false	true	false
true	true	true	true	false

Voici ensuite la représentation compacte en 0/1 (0 = false, 1 = true) :

φ	ψ	$\varphi \wedge \psi$	$\varphi \vee \psi$	$\neg\varphi$
0	0	0	0	1
0	1	0	1	1
1	0	0	1	0
1	1	1	1	0

Remarque. La définition ci-dessus de la fonction de vérité est équivalente à la définition inductive de la relation de satisfaction $\models_{\mathrm{LP}}$. Dans les preuves, l'une ou l'autre formulation peut être plus commode selon le contexte : la relation de satisfaction met l'accent sur la perspective des modèles ($\omega \models \varphi$), tandis que la fonction

de vérité indique explicitement la valeur d'une formule dans une interprétation donnée ($[\![\varphi]\!]_\omega \in \{\text{true}, \text{false}\}$).

Perspective

Dans la section suivante, nous étudierons les formes normales syntaxiques, les règles concrètes de transformation (par exemple FNC, FND) ainsi que les premières méthodes de preuve (résolution, déduction naturelle), qui sont à la fois importantes sur le plan théorique et utiles en pratique (par exemple pour les solveurs SAT).

1.3 Exercices avec solutions

Exercice 1. Soit la signature propositionnelle $\Sigma = \{P, Q, R, S, T, U\}$. On considère les expressions

$$\begin{aligned} \varphi_1 &= S \wedge \neg(P \leftrightarrow \neg R), \\ \omega_2 &: P \mapsto \text{true},\ Q \mapsto \text{false},\ R \mapsto \text{true}, \\ &\quad S \mapsto \text{false},\ T \mapsto \text{true},\ U \mapsto \text{false}. \end{aligned}$$

et

$$\varphi_3 \ = \ \neg T \wedge \neg(\neg T \vee R).$$

Décidez, pour chacun des énoncés suivants, s'il est **vrai** ou **faux** et donnez à chaque fois une brève justification :

1. $\neg P \vee R \in L(\Sigma)$.
2. La formule indiquée

$$\overline{\varphi_1} \ = \ \neg S \vee (\neg P \vee \neg R) \wedge (R \vee P) \vee \neg Q$$

est le complément, c'est-à-dire la négation, de φ_1.

3. $[\![\neg T \vee (\neg R \leftrightarrow P)]\!]_{\omega_2} = \text{false}$.
4. $|\operatorname{Mod}_{\mathrm{LP}}(\varphi_3)| = 0$.

Solutions et explications.

1. **Vrai.** $\neg P \vee R$ est une expression bien formée, construite à partir d'atomes de Σ au moyen des connecteurs $\neg$ et $\vee$; c'est donc une formule de $L(\Sigma)$.
2. **Faux.** La formule proposée n'est pas la négation de φ_1. Premièrement, elle contient l'atome Q, qui n'apparaît pas dans φ_1 ; deuxièmement, d'après les lois de De Morgan, la structure syntaxique d'une négation de $\varphi_1 = S \wedge \neg(P \leftrightarrow \neg R)$ est équivalente à

$$\neg S \vee \neg\neg(P \leftrightarrow \neg R) \;=\; \neg S \vee (P \leftrightarrow \neg R),$$

   et non à la formule plus compliquée indiquée, qui contient $\neg Q$ ainsi que le terme supplémentaire $(\neg P \vee \neg R) \wedge (R \vee P)$.
3. **Vrai.** Calculons la formule $\neg T \vee (\neg R \leftrightarrow P)$ sous l'interprétation ω_2. Sous ω_2, on a $T = \text{true}$, donc $\neg T = \text{false}$. De plus, $R = \text{true}$, donc $\neg R = \text{false}$. Comme $P = \text{true}$, l'équivalence $\neg R \leftrightarrow P$ relie false et true, et vaut donc false. Ainsi, $\neg T \vee (\neg R \leftrightarrow P)$ vaut false $\vee$ false $=$ false. L'énoncé est donc correct.
4. **Vrai.** Simplifions $\varphi_3 = \neg T \wedge \neg(\neg T \vee R)$. La sous-formule $\neg(\neg T \vee R)$ est équivalente à $\neg\neg T \wedge \neg R$, donc à $T \wedge \neg R$. Par conséquent, $\varphi_3 \equiv \neg T \wedge (T \wedge \neg R)$.

Cette formule est manifestement insatisfaisable, car elle exigerait simultanément $\neg T$ et T. Ainsi, φ_3 n'a *aucun* modèle, et donc $|\operatorname{Mod}_{\mathrm{LP}}(\varphi_3)| = 0$.

Retour bref :

— (1) La formule appartient bien au langage construit sur Σ — correct.

— (2) La formule de négation proposée est fausse : elle contient un atome étranger et ne possède pas la bonne structure (elle n'est pas équivalente à la négation de φ_1).

— (3) Vrai — l'évaluation donne false.

— (4) Vrai — φ_3 est contradictoire, donc insatisfaisable.

Exercice approfondi. Faites varier la signature Σ (par exemple en ne gardant que trois atomes) et déterminez, pour toutes les $2^{|\Sigma|}$ interprétations, quelles formules de l'exercice 1 sont vraies. Comparez ensuite le nombre de modèles des différentes formules.

Solution modèle (exemple avec $\Sigma' = \{P, Q, R\}$)

Nous choisissons $\Sigma' = \{P, Q, R\}$ et définissons les formules analogues à celles de l'exercice (adaptation de l'exercice initial à une signature plus petite) :

$$\begin{aligned}
\varphi_1 &:= Q \wedge \neg(P \leftrightarrow \neg R),\\
\alpha &:= \neg P \vee R,\\
\beta &:= \neg Q \vee \big((\neg P \vee \neg R) \wedge (R \vee P)\big) \quad \text{(candidat pour } \neg\varphi_1),\\
\gamma &:= \neg R \vee (\neg P \leftrightarrow Q),\\
\delta &:= \neg P \wedge \neg(\neg P \vee R).
\end{aligned}$$

La formule $(\neg P \vee \neg R) \wedge (R \vee P)$ est équivalente à $(P \leftrightarrow \neg R)$. Ainsi, β représente exactement $\neg Q \vee (P \leftrightarrow \neg R)$, c'est-à-dire formellement la négation de $\varphi_1 = Q \wedge \neg(P \leftrightarrow \neg R)$.

La table de vérité complète (T = true, F = false) :

P	Q	R	φ_1	α	β	γ	δ
F	F	F	F	T	T	T	F
F	F	T	F	T	T	F	F
F	T	F	T	T	F	T	F
F	T	T	F	T	T	T	F
T	F	F	F	F	T	T	F
T	F	T	F	T	T	T	F
T	T	F	F	F	T	T	F
T	T	T	T	T	F	F	F

À partir de cette table, nous lisons le nombre de modèles, c'est-à-dire le nombre d'interprétations pour lesquelles la formule est vraie :

— $|\operatorname{Mod}(\varphi_1)| = 2$ (vraie pour les valuations $(P, Q, R) = (F, T, F)$ et (T, T, T))

— $|\operatorname{Mod}(\alpha)| = 6$

— $|\operatorname{Mod}(\beta)| = 6$

— $|\operatorname{Mod}(\gamma)| = 6$

— $|\operatorname{Mod}(\delta)| = 0$ (insatisfaisable)

Interprétation et conclusions

— $\alpha = \neg P \vee R$ n'est pas tautologique (elle n'est pas vraie dans toutes les interprétations), mais elle est satisfaite dans 6 cas sur 8 : elle est donc souvent vraie, sans être universellement valide.

— Dans cette formulation précise, β est effectivement logiquement équivalente à $\neg\varphi_1$. On le voit algébriquement :

$$\begin{aligned}\neg\varphi_1 &= \neg\big(Q \wedge \neg(P \leftrightarrow \neg R)\big)\\ &= \neg Q \vee \neg\neg(P \leftrightarrow \neg R)\\ &= \neg Q \vee (P \leftrightarrow \neg R)\end{aligned}$$

et $P \leftrightarrow \neg R$ est équivalente à $(\neg P \vee \neg R) \wedge (R \vee P)$. Par conséquent, β et $\neg\varphi_1$ ont les mêmes modèles (6 interprétations).

— γ est elle aussi vraie dans 6 interprétations ; son ensemble de modèles diffère, dans sa composition exacte, de ceux de α et de β, mais les cardinalités sont identiques (6).

— δ est contradictoire : après transformation, $\delta = \neg P \wedge (\ P \wedge \neg R\ )$, ce qui exige simultanément P et $\neg P$. Elle n'a donc aucun modèle.