Préface

La logique consiste à préciser ce qu'est un *énoncé* et plus précisément comment on assemble des briques de base du langage pour former des énoncés complexes. Dès l'antiquité, avec Aristote, on a compris que cet assemblage ne dépendait pas du domaine du discours et ne nécessitait que quelques connecteurs logiques et quelques quantificateurs. Dans son syllogisme le plus connu : «tous les hommes sont mortels, Socrate est un homme donc Socrate est mortel», on a déjà la conjonction (la virgule), l'implication («donc») et la quantification universelle («tous»). Et même si la question ne se posait sans doute pas à cette époque, on sait aujourd'hui qu'il suffit de rajouter la négation pour avoir un jeu complet permettant de construire tous les énoncés.

La logique consiste aussi à préciser comment on établit la *vérité* d'un énoncé à partir des axiomes d'un domaine. Là encore, cela ne dépend pas du domaine et peut s'étendre dans une certaine mesure à la physique, la philosophie ou même la politique, et dès l'époque d'Aristote, un certain nombre de *règles de démonstration* était connues, par exemple pour établir la vérité du syllogisme sus-cité.

Depuis le début du 20e siècle, avec Hilbert, Frege, Gentzen, Gödel, et bien d'autres, on connaît *la logique du premier ordre* qui permet d'exprimer tous les énoncés. On connaît aussi plusieurs notions de démonstration, toutes équivalentes, bien que très différentes (systèmes à la Hilbert, méthode des tableaux, déduction naturelle, etc.). On sait même que pour tout domaine contenant l'arithmétique, il y aura toujours des énoncés vrais dont on ne peut établir la vérité, c-à-d des énoncés vrais mais *indémontrables* (c'est le premier théorème d'incomplétude de Gödel).

La logique du premier ordre, présentée dans cet ouvrage, définit formellement l'ensemble des énoncés comme des arbres, la vérité dans un *modèle* et la prouvabilité par l'existence d'un enchaînement de règles permettant de passer des axiomes à l'énoncé visé.

Quant à l'informatique, elle est aussi née dès l'antiquité, avec les premiers algorithmes tel que l'algorithme d'Euclide pour le calcul du PGCD (plus grand commun diviseur) de deux entiers. Les premiers langages de programmation comme le λ-calcul de Church, le système T de Gödel ou les machines de Turing sont apparus dans les années 1930, avant les premiers ordinateurs. La «thèse de Church» affirme que tous ces langages sont équivalents.

À cette époque, il s'agissait de comprendre la différence entre recherche de preuves et calcul mécanique. Hilbert, dans son fameux programme, se demandait si il n'existait pas un moyen mécanique d'établir la vérité des énoncés. Et c'est justement le résultat

de Gödel et la thèse de Church qui ont établi une distinction définitive entre énoncés prouvables et énoncés calculables.

Avec ce point de vue la logique et l'informatique sont intimement liées. En effet, la première définit les énoncés *prouvables*, c-à-d les énoncés pour lesquels on peut produire une démonstration. Tandis que la seconde définit les énoncés *calculables*, c-à-d les énoncés pour lesquels on peut définir un programme qui distingue les énoncés vrais des énoncés faux. Ce lien profond justifie à lui seul que la logique soit un des piliers de base d'un cursus d'informatique.

Par ailleurs, avec l'augmentation de la puissance des ordinateurs et la découverte de nouveaux algorithmes, la preuve automatique a fait d'immenses progrès et se trouve de plus en plus présente en particulier pour vérifier des circuits logiques et des programmes informatiques. On voit ici un double lien entre l'informatique et la logique : l'utilisation de l'informatique pour écrire des algorithmes recherchant des preuves d'énoncés et l'utilisation de ces mêmes algorithmes pour démontrer la correction de matériels ou logiciels informatiques. Cet ouvrage donne les bases pour appréhender ces deux aspects dans les derniers chapitres en abordant la résolution, une méthode classique de recherche de preuves, et en évoquant le *model checking* pour exprimer et vérifier des propriétés de logiciels.

De plus, il existe un lien profond entre preuves et programmes ! En effet, on peut voir une preuve d'une implication $A \Rightarrow B$ comme un programme calculant une preuve de B à partir d'une preuve de A. Cette correspondance dite de *Curry-Howard* n'est pas juste une curiosité théorique. Elle est présente et mise en œuvre dans tous les langages fonctionnels typés de la famille ML comme OCaml ou Haskell. Utiliser au maximum cette correspondance est l'une des voies prometteuses que l'on explore pour produire des logiciels prouvés soit en enrichissant les langages de la famille ML, soit en extrayant directement des programmes à partir de preuves avec des logiciels d'aide à la construction de preuves comme CoQ.

Enfin, les logiciels que l'on met en œuvre pour ou avec la logique sont relativement différents des logiciels informatiques plus classiques sur les graphes ou les tableaux. Ceci et les liens profonds entre logique et informatique que nous avons évoqués plus haut rendent cet ouvrage indispensable pour compléter une culture générale en informatique. Le livre «Logique pour l'informatique» que vous vous apprêtez à lire est donc bien un incontournable du domaine.

Christophe RAFFALLI
Ancien Maître de Conférences
Capitaine d'Ou La La

Table des matières

Introduction

Depuis son antique origine chez Aristote, la logique cherche à déterminer les lois générales du raisonnement. Pour mener à bien cette entreprise, elle considère deux aspects du langage par quoi le raisonnement est exprimé :

— quels sont les énoncés dont on peut dire qu'ils sont «vrais» ou qu'ils sont «faux» ?

— quels sont les enchaînements de tels énoncés dont on peut affirmer que la vérité des uns a pour conséquence la vérité des autres ?

Au fur et à mesure de son développement, la logique s'est rapprochée des méthodes des mathématiques. Dans cette intention, le philosophe et mathématicien G. W. Leibniz a tenté, au XVII^{e} siècle, de définir une langue formelle (la *lingua caracteristica*) dans laquelle on énonce les problèmes à trancher et de lui adjoindre une procédure de calcul (le *calculus ratiocinator)* permettant de trancher effectivement la question. Au XIX^{e} siècle, G. Boole [Boo92] donna une tournure mathématique à ce projet de calcul logique en exhibant comment l'on pouvait réduire certains de ses aspects à un simple calcul sur un domaine à deux valeurs ({«vrai», «faux»} ; {0, 1}) que l'on appelle depuis *valeurs booléennes*. Poursuivant cet effort de «mathématisation» de la logique, G. Frege [Fre99] donna, au tournant des XIX^{e} et XX^{e} siècles, une nouvelle analyse des énoncés élémentaires que considère la logique basée sur les notions de «variable» et de «fonction» tels que l'usage s'en était développé en mathématique. Il développa une langue logique purement formelle : l'idéographie (*Begriffsschrift*). Avec G. Frege, les énoncés sont devenus des *formules*. Reste une dernière étape à franchir, un formalisme purement symbolique pour l'écriture des démonstrations. Les premières tentatives en ce sens s'inspiraient de la méthode axiomatique (un ensemble fini de schémas de formules admises comme «vraies») complétée par des règles de déduction. G. Gentzen [Gen55] paracheva en quelque sorte l'entreprise en élaborant un système de *preuve* où tout est règle de déduction. La langue logique étant devenue parfaitement abstraite, ses formules n'y ont aucun sens par elles-mêmes. Leur sémantique doit être apportée par un élément externe : un domaine d'*interprétation* pour les symboles de variable et les relations que peuvent entretenir entre eux les éléments de ce domaine. Les formules, une fois interprétées, ont une valeur de vérité. Cette notion fut assez longtemps utilisée intuitivement jusqu'à ce que A. Tarski [Tar72] la précise plus formellement.

La «logique» que nous présentons et étudions dans ce livre est l'héritière de ces travaux. C'est la *logique des prédicats du premier ordre*.

Les quatre premiers chapitres de ce livre donnent la définition de cette logique sous ses trois aspects : le langage des formules, le système de preuve et l'interprétation des formules. La matière de ces chapitres est issue d'un cours de logique dispensé à des étudiants de licence d'informatique. Chacun d'eux se termine par un ensemble d'exercices dont les solutions sont données en annexe à la fin de cet ouvrage. Le niveau d'exposition de ces quatre premiers chapitres mais aussi des suivants reste adapté à ce public et ne réclame aucun prérequis.

Le premier chapitre est consacré à la présentation du langage de la logique des prédicats du premier ordre. C'est essentiellement celui défini par G. Frege. Techniquement, l'ensemble des formules, c-à-d l'ensemble des suites de caractères considérées comme des formules, y est défini par un ensemble de règles d'assemblage syntaxique de symboles : symboles de variable, constante et fonction, symboles de prédicat, symboles des connecteurs et quantificateurs logiques. Nous terminons ce chapitre par la définition d'une opération de transformation purement syntaxique des formules : la substitution.

Le deuxième chapitre est consacré au système de preuve de la *déduction naturelle* restreint aux formules sans variable de la logique des prédicats (le «fragment propositionnel» du langage). Nous faisons dans ce chapitre un sort particulier au «raisonnement par l'absurde» dont l'usage ou non distingue une logique dite «classique» d'une logique dite «intuitionniste».

Le troisième chapitre est consacré, toujours sur le fragment propositionnel, à la définition de l'interprétation sémantique (la «signification») des composants syntaxiques des formules : les symboles de constante et de fonction sont projetés sur des objets d'une «structure» ; les symboles de prédicat sur les propriétés de ces objets ou leurs relations ; les symboles logiques sur des fonctions booléennes. Sur cette base, on sait donner un sens à l'expression «formule vraie» ou à une déduction correcte appelée *conséquence sémantique*.

Le quatrième chapitre ajoute aux deux précédents l'usage des symboles de variable et des symboles de quantification au système de la déduction naturelle et à la sémantique des formules.

À l'issue de ces quatre premiers chapitres, la logique est devenue un objet parfaitement cerné dont on peut étudier les propriétés. Les deux chapitres suivants s'attachent à deux d'entre elles.

Le chapitre 5 contient les démonstrations de deux résultats mettant en relation preuves en déduction naturelle et conséquence sémantique. Il s'agit :

— du théorème de correction qui établit que les déductions prouvables en déduction naturelle sont sémantiquement fondées ;
— du théorème de complétude, réciproque de la correction, qui établit que toutes les déductions sémantiquement fondées sont prouvables en déduction naturelle.

Nous avons tenté de rendre les preuves de ces théorèmes aussi abordables que possible pour un lecteur qui n'aurait de connaissance du domaine que le contenu des quatre premiers chapitres de ce livre. La démonstration du théorème de correction, quoiqu'un peu fastidieuse, ne requiert rien d'autre que ce qui a été dit dans cet ouvrage. La démonstration du théorème de complétude, plus sophistiquée, fait appel à quelques notions externes à la logique que nous présentons aussi succinctement que possible.

Le chapitre 5 se conclut en situant ces résultats dans les enjeux de l'histoire de la logique.

Le chapitre 6 présente un autre résultat concernant la logique des prédicats du premier ordre : son indécidabilité. Pour énoncer et démontrer ce résultat, nous introduisons en première partie de ce chapitre une notion formelle de *calculabilité* qui est nécessaire pour énoncer ce qu'est un problème de décidabilité. Dans cette première partie, nous démontrons qu'un problème propre à la théorie de la calculabilté est indécidable : le problème dit «de l'arrêt». Dans une seconde partie de ce chapitre, nous en déduisons l'indécidabilité de la logique des prédicats. Les démonstrations contenues dans ce chapitre sont bien moins «fouillées» que celles du chapitre précédent, essentiellement parce que leur développement aurait nécessité beaucoup trop de place dans ce livre. Nous espérons toutefois avoir pu y être suffisamment simples et convaincants.

La formalisation de la logique ouvre la possibilité de sa mise en œuvre par des moyens informatiques. Les cinq derniers chapitres se tournent vers quelques uns de ces aspects.

Le chapitre 7 s'attache à la question de la recherche des conditions de vérité d'une formule du fragment propositionnel de la logique. Cette question est connue sous le nom de *problème de la satisfiabilité* des formules propositionnelles. Nous y donnons une définition plus précise de ce «fragment», sa syntaxe et sa sémantique. Nous étudions comment sa sémantique entretient un rapport étroit avec les fonctions booléennes. Enfin, nous isolons quelques formes particulières de formules propositionnelles qui jouent un rôle important tant du point de vue théorique que pratique : les formes normales disjonctives et conjonctives, ainsi que les clauses de Horn.

Le chapitre 8 est consacré à une méthode de preuve alternative à la déduction naturelle qui a donné lieu à d'importantes applications en informatique. Il s'agit de la méthode de *preuve par résolution*. Parmi ces importantes applications figure celle de programmation logique. Les réalisations concrètes de la méthode de résolution reposent sur un algorithme important : *l'unification*. Nous donnons une version abstraite de cet algorithme dans la dernière section de ce chapitre.

Le chapitre 9 présente une très brève et intuitive introduction à un développement remarquable de la théorie des preuves en déduction naturelle : la correspondance entre preuves et programmes (plus formellement, entre preuves et fonctions calculables).

Le chapitre 10 illustre comment, dans le domaine concret des bases de données, la logique des prédicats du premier ordre, sa syntaxe et sa sémantique sont au fondement de la conception des langages de requêtes permettant d'exploiter les données recueillies dans une base.

Enfin le chapitre 11 présente une réalisation des *méthodes formelles* de vérification de systèmes informatiques basée sur les concepts élaborés en logique formelle : la vérification de modèle (*model checking*).