Chapitre 8 : Logique

Table des matières

1 Introduction

La logique est un domaine mathématique dont l'objet d'étude est l'ensemble des propriétés que l'on peut exprimer, leur valeur de vérité et la notion de démonstration. La logique propose un cadre formel pour manipuler ces notions, où l'on distingue la *syntaxe*, *i.e* la manière d'exprimer les propriétés, de la *sémantique*, *i.e* des interprétations que l'on peut donner à la syntaxe.

Les démonstrations, vues en tant qu'objets mathématiques, ne seront étudiées qu'en MPI.

2 Syntaxe des formules logiques

2.1 Logique propositionnelle

2.1.1 Introduction

On décompose les propriétés que l'on cherche à exprimer pour faire ressortir leur structure logique. Les énoncés que l'on ne peut pas décomposer sont appelés *atomes*. En logique propositionnelle, on abstrait les atomes en des variables dites propositionnelles.

2.1.2 Définition (formules de la logique propositionnelle)

Soit V un ensemble de variables propositionnelles.

on définit inductivement l'ensemble $\mathcal{F}$ des formules de la logique propositionnelle par

$$\frac{x \in V}{x \in \mathcal{F}} \qquad \frac{\varphi \in \mathcal{F}}{\neg\varphi \in \mathcal{F}}$$

$$\frac{\varphi_1 \in \mathcal{F}\ \varphi_2 \in \mathcal{F}}{\varphi_1 \vee \varphi_2 \in \mathcal{F}} \qquad \frac{\varphi_1 \in \mathcal{F}\ \varphi_2 \in \mathcal{F}}{\varphi_1 \wedge \varphi_2 \in \mathcal{F}} \qquad \frac{\varphi_1 \in \mathcal{F}\ \varphi_2 \in \mathcal{F}}{\varphi_1 \to \varphi_2 \in \mathcal{F}}$$

On appelle $\neg$ la *négation*, $\vee$ la *disjonction*, $\wedge$ la *conjonction*, et $\to$ l'*implication*.

On appelle aussi *équivalence* le symbole $\leftrightarrow$ défini par

$$\varphi_1 \leftrightarrow \varphi_2 = (\varphi_1 \to \varphi_2) \wedge (\varphi_2 \to \varphi_1)$$

2.1.3 Remarque

Malgré leur nom, les symboles de la définition précédente n'ont aucune signification.

Le sens des symboles tient au domaine de la sémantique. On parle ici de syntaxe abstraite.

On peut utiliser une autre représentation pour les syntaxes abstraites, appelée *grammaire*, qui s'écrit comme suit :

$$\varphi ::= x \mid \neg\varphi \mid \varphi_1 \vee \varphi_2 \mid \varphi_1 \wedge \varphi_2 \mid \varphi_1 \to \varphi_2$$

où x parcourt V.

L'étude précise des grammaires relève du programme de MPI.

2.1.4 Vocabulaire

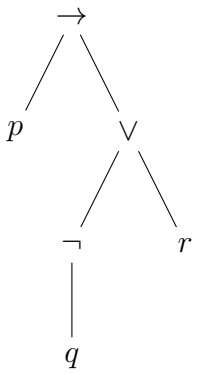
Les symboles $\neg, \vee, \wedge, \rightarrow, \leftrightarrow$ sont appelés connecteurs logiques. Ce sont eux qui définissent la structure logique des énoncés. Le nombre d'arguments d'un connecteur logique est appelé son *arité*.

2.1.5 Représentation des formules

On représente généralement les formules de manière linéaire en ajoutant des parenthèses pour désambigüiser la lecture. Par exemple, $p \vee q \wedge r$ peut se lire $p \vee (q \wedge r)$ ou $(p \vee q) \wedge r$. Par convention, la négation est prioritaire sur les autres connecteurs, donc $\neg p \vee q$ se lit $(\neg p) \vee q$ et pas $\neg(p \vee q)$.

En raison de leur définition inductive, les formules ont une représentation arborescente naturelle : chaque règle d'inférence utilisée définit un symbole de tête / connecteur principal qui est la racine du sous-arbre correspondant.

Exemple : $p \rightarrow (\neg q \vee r)$ est représenté par :



Remarque : on obtient un arbre (binaire) dont les nœuds internes sont les connecteurs logiques et les nœuds externes sont les atomes. L'arité d'un connecteur est l'arité du nœud correspondant.

L'écriture linéaire de la formule correspond au parcours en profondeur infixe, en adoptant une représentation préfixe pour les nœuds d'arité 1.

On pourrait envisager d'utiliser les parcours préfixes et postfixe pour représenter les formules.

2.1.6 Vocabulaire

– Une *sous-formule* d'une formule φ donnée est la formule associée à un sous-arbre de l'arbre représentant φ.

On définit inductivement l'ensemble $SF(\varphi)$ des sous-formules de φ par :

$$\forall x \in V,\ SF(x) = \{x\} \qquad SF(\neg\varphi) = \{\neg\varphi\} \cup SF(\varphi)$$

$$\forall \circ \in \{\vee, \wedge, \rightarrow\},\ SF(\varphi_1 \circ \varphi_2) = \{\varphi_1 \circ \varphi_2\} \cup SF(\varphi_1) \cup SF(\varphi_2)$$

Exemple :

$$SF(p \to (\neg q \vee r)) = \{p \to (\neg q \vee r), \neg q \vee r, p, \neg q, r, q\}$$

– La *taille* $|\varphi|$ d'une formule φ est le nombre de connecteurs logiques de la formule, *i.e* le nombre de nœuds internes de l'arbre A_φ associé à φ.

Proposition :

$$\frac{|A_\varphi| - 1}{2} \leqslant |\varphi| \leqslant |A_\varphi| - 1$$

□ Démonstration :

L'arbre A_φ contient au moins une feuille donc $|\varphi| \leqslant |A_\varphi| - 1$
(cas d'égalité pour φ de taille n : $\varphi = \underbrace{\neg\neg\cdots\neg}_{n \text{ fois}} p$)

A_φ est un arbre binaire ayant $|\varphi|$ nœuds internes, donc il a au plus $|\varphi| + 1$ feuilles (Chap 6, 1.1.10), donc $|A_\varphi| \leqslant |\varphi| + |\varphi| + 1 = 2|\varphi| + 1$ ■

– La *hauteur* d'une formule est la hauteur de l'arbre associé, *i.e* le nombre maximal de connecteurs à traverser pour atteindre une variable propositionnelle.
Exemple : la hauteur de $p \to (\neg q \vee r)$ est 3.

2.2 Logique du premier ordre

2.2.1 Définition (*langage du premier ordre*)

Un *langage du premier ordre* est défini par une signature Σ, composée de :

– symboles de fonction, chacun muni d'une arité $k \in \mathbb{N}$. Les symboles de fonction d'arité 0 sont appelés symboles de constante ;

– symboles de prédicat ou de relation, chacun muni d'une arité $k \in \mathbb{N}$. Les symboles de prédicat d'arité 0 sont appelés symboles de constante propositionnelle.

2.2.2 Exemple : le langage de la théorie des ensembles

Le langage de la théorie des ensembles est défini par la signature suivante :

– symboles de fonction : $\varnothing$ (arité 0), $\{\cdot\}$ (arité 1), $\cdot \cup \cdot$ et $\cdot \cap \cdot$ (arité 2), $\cdot^c$ (arité 1) ;

– symboles de prédicat : $\cdot = \cdot$, $\cdot \subseteq \cdot$, $\cdot \in \cdot$ (arité 2).

2.2.3 Définition (*termes* et formules de la logique du premier ordre)

Soit Σ une signature et V un ensemble de variables.
Σ et V définissent des ensembles de termes et de formules, *via* les grammaires suivantes :

– Termes :

$$t ::= x \mid f(t_1, \ldots, t_k)$$

où

$$\begin{cases} x \text{ parcourt } V \\ (f, k) \text{ parcourt les symboles de la fonction et leur arité} \end{cases}$$

– Formules :

$$\varphi ::= p(t_1, \ldots, t_k) \mid \neg\varphi \mid \varphi_1 \vee \varphi_2 \mid \varphi_1 \wedge \varphi_2 \mid \varphi_1 \to \varphi_2 \mid \forall x\ \varphi \mid \exists x\ \varphi$$

où :

$$\begin{cases} x \text{ parcourt } V \\ (p, k) \text{ parcourt les symboles de prédicat et leur arité} \end{cases}$$

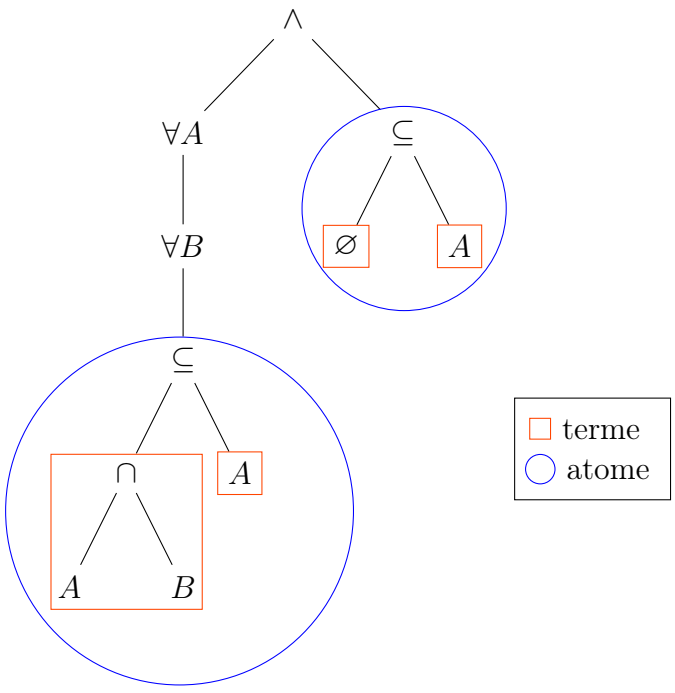
On appelle $\forall$ le *quantificateur universel*, et $\exists$ le *quantificateur existentiel*.
Les atomes de cette logique sont les formules de la forme $p(t_1, \ldots, t_k)$.

$$\exists! x,\ P(x) \ \equiv\ \exists x\ (P(x) \wedge \forall y (P(y) \to y = x))$$

On parle de logique du premier ordre car on ne peut quantifier que sur des variables représentant des termes. Si l'on peut quantifier sur des variables représentant des formules, on parle de logique du second ordre.

2.2.4 Exemple

En théorie des ensembles, la formule $(\forall A, \forall B, A \cap B \subseteq A) \wedge \varnothing \subseteq A$ est représentée de manière arborescente par :



2.2.5 Remarque

• La logique du premier ordre est aussi appelée calcul des prédicats.

• Dans une formule du premier ordre, les variables peuvent être "capturées" par un quantificateur ou indépendantes de toute quantification.
On peut voir une formule comme une propriété des variables indépendantes de toute quantification et donc remplacer ces variables par des termes concrets.

• Le calcul propositionnel est un cas particulier du calcul des prédicats où il n'y a que des constantes propositionnelles (les quantificateurs et les termes deviennent inutiles).

2.2.6 Définition (variables libres / liées)

Les *variables libres* d'une formule φ sont les variables qui ne sont pas "capturées" par un quantificateur. On les définit inductivement par

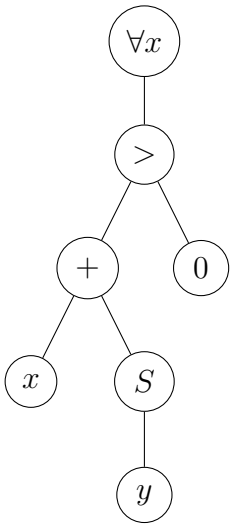
$$FV(p(t_1,\dots,t_n)) = \bigcup_{i=1}^{n} \text{Vars}(t_i) \quad \text{où} \quad \begin{cases} \text{Vars}(x) = \{x\} \\ \text{Vars}(f(t_1,\dots,t_n)) = \bigcup_{i=1}^{n} \text{Vars}(t_i) \end{cases}$$

$$FV(\neg\varphi) = FV(\varphi)$$

$$\forall \circ \in \{\vee, \wedge, \to\},\ FV(\varphi_1 \circ \varphi_2) = FV(\varphi_1) \cup FV(\varphi_2)$$

$$\forall Q \in \{\exists, \forall\},\ FV(Qx\varphi) = FV(\varphi) \setminus \{x\}$$

Exemple :
$FV(\forall x,\ x + Sy > 0) = \{y\}$



$FV((\forall x\ x + y = 1) \wedge (\forall y\ x + y = 1)) = \{x, y\}$
Une variable est dite *liée* si elle n'est pas libre. Un même nom de variable peut avoir des occurrences libres et des occurrences liées. Dans une formule de la forme $Q\ x\ \varphi$ où $Q \in \{\forall, \exists\}$, on dit que φ est la portée de la liaison pour x. Une variable est donc libre si elle admet une occurence hors de la portée de toutes les liaisons pour cette variable

et une variable est liée si toutes ses occurrences sont dans la portée d'une liaison pour cette variable.

Une formule dont toutes les variables sont liées est dite *close*.

2.2.7 Remarque

Le nom des variables liées n'est pas important.

Exemple : $\forall x,\ x = x$ et $\forall y,\ y = y$ expriment la même propriété.

On identifiera donc les formules au renommage près de leurs variables liées.

On appelle cela *l'α-équivalence*.

Lors du renommage de variables liées, il faut faire attention au phénomène de *capture* de variables, par exemple $\forall y,\ x + y = 1$ n'est pas la même formule que $\forall x,\ x + x = 1$.

2.2.8 Définition (*substitution*)

Soit φ une formule, x une variable, et t un terme.

La substitution de t à x dans φ, notée $\varphi[x := t]$ est définie inductivement par :

$$p(t_1, \ldots, t_n)[x := t] = p(t_1[x := t], \ldots, t_n[x := t])$$

où

$$\begin{cases} x[x := t] = t \\ y[x := t] = y \ \forall y \in V \setminus \{x\} \\ f(t_1, \ldots, t_n)[x := t] = f(t_1[x := t], \ldots, t_n[x := t]) \end{cases}$$

$(\neg\varphi)[x := t] = \neg(\varphi[x := t])$

$\forall \circ \in \{\vee, \wedge, \to\},\ (\varphi_1 \circ \varphi_2)[x := t] = \varphi_1[x := t] \circ \varphi_2[x := t]$

$\forall Q \in \{\forall, \exists\},\ (Q\ x\ \varphi)[x := t] = Q\ x\ \varphi$

$\forall Q \in \{\exists, \forall\},\ (Q\ y\ \varphi)[x := t] = Q\ y\ (\varphi[x := t])$ si $y \neq x$ et $y \notin \mathrm{Vars}(t)$

Exemple :

$(\forall x,\ x = x)[x := 2 + 2] = \forall x,\ x = x$ car $(\forall x,\ x = x) \underset{\alpha}{\equiv} (\forall y,\ y = y)$

$((\forall x,\ x + y = 1) \wedge (\forall y,\ x + y = 1))[x := 2] = (\forall x,\ x + y = 1) \wedge (\forall y,\ 2 + y = 1)$

$(\forall y,\ y = y + x)[x := 1 + y] \neq \forall y(y = y + 1 + y)$

Mais plutôt $\forall z,\ z = z + 1 + y$: on renomme les occurrences liées dans φ des variables de t avant de les substituer.

2.2.9 Remarque

Le principe de l'α-équivalence et les restrictions de la substitution sont liées aux questions de sémantique : l'α-équivalence et la substitution doivent en quelque sorte conserver la signification logique des formules.

3 Sémantique de la logique propositionnelle

3.1 Vocabulaire

3.1.1 Introduction

Définir une sémantique revient à donner du sens aux symboles utilisés dans la syntaxe abstraite. On doit donc choisir un ensemble de valeurs qui servent d'interprétations aux termes construits à l'aide de la syntaxe et on doit décrire l'effet des symboles sur cet ensemble de valeurs.

Exemple : on considère des termes arithmétiques définis par :

$$t ::= x \mid c \mid t_1 + t_2 \mid t_1 - t_2 \mid t_1 \times t_2$$

où x parcourt un ensemble V de variables et c parcourt $\mathbb{N}$, l'ensemble des constantes.

On peut définir une sémantique en choisissant $\mathbb{N}$ pour l'ensemble des valeurs, $c + 1$ comme interprétation de c, la fonction min comme interprétation de $+$, la fonction max pour $-$, et l'addition pour $\times$.

On peut bien-sûr donner une autre sémantique à ces termes, plus en cohérence avec les règles de l'arithmétique.

Problème : l'interprétation des variables : elle dépend d'un contexte qui donne une valeur à chaque variable.

Les sémantiques sont donc paramétrées par un environnement.

3.1.2 Définition (*valuation*)

Une *valuation* est une fonction de l'ensemble $\mathcal{V}$ des variables dans l'ensemble des valeurs choisi pour définir la sémantique. On parle aussi d'environnement, ou, dans le cas de la logique propositionnelle, de *distribution* de vérité. L'ensemble des valeurs de vérité est noté $\{V, F\}$ où V est la valeur vraie et F la valeur fausse.

3.1.3 Définition (valeur de vérité d'une formule)

Soit φ une formule et v une valuation.

On définit inductivement l'interprétation de φ pour v, notée $[\![\varphi]\!]_v$, par :

$$\forall x \text{ variable propositionnelle, } [\![x]\!]_v = v(x)$$

$$[\![\neg\varphi]\!]_v = \begin{cases} V \text{ si } [\![\varphi]\!]_v = F \\ F \text{ sinon} \end{cases}$$

$$[\![\varphi_1 \vee \varphi_2]\!]_v = \begin{cases} F \text{ si } [\![\varphi_1]\!]_v = [\![\varphi_2]\!]_v = F \\ V \text{ sinon} \end{cases}$$

$$[\![\varphi_1 \wedge \varphi_2]\!]_v = \begin{cases} V \text{ si } [\![\varphi_1]\!]_v = [\![\varphi_2]\!]_v = V \\ F \text{ sinon} \end{cases}$$

$$[\![\varphi_1 \to \varphi_2]\!]_v = \begin{cases} F \text{ si } [\![\varphi_1]\!]_v = V \text{ et } [\![\varphi_2]\!]_v = F \\ V \text{ sinon} \end{cases}$$

On dit v est un modèle de $\varphi \Leftrightarrow [\![\varphi]\!]_v = V$

3.1.4 Définition (*tautologie* / *antilogie* / *satisfiabilité*)

Soit φ une formule.
On dit que φ est
- une *tautologie* $\Leftrightarrow$ toute valuation est un modèle de φ. On note alors $\vDash^{\varphi}$;
- une *antilogie* $\Leftrightarrow$ elle n'admet aucun modèle ;
- *satisfiable* $\Leftrightarrow$ elle admet un modèle.

3.1.5 Remarque

On ajoute parfois à la syntaxe une tautologie notée $\top$ et une antilogie notée $\bot$.
On peut toutefois les encoder :

$$\top = x \vee \neg x \qquad \bot = x \wedge \neg x$$

La tautologie $\varphi \vee \neg\varphi$ est appelée loi du tiers exclu.
Exo Montrer que les formules suivantes sont des tautologie :
$p \to (q \to p), (p \to q) \vee (q \to r), (p \to (q \to r)) \to ((p \to q) \to (p \to r))$

Ex : Falso Quadlilet : $\bot \to \varphi$
$\neg\neg\varphi \to \varphi$ (non admis par tout le monde)
principe de démonstration par l'absurde : $\neg\neg\varphi = \neg\varphi \to \bot$

3.1.6 Définition (*table de vérité*)

La *table de vérité* de φ est la table indexée par les valuations des variables de φ et qui contient comme entrée correspondant à une valuation v la valeur $[\![\varphi]\!]_v$.
On représente la table de φ, de variables $x_1, \ldots, x_n$, en plaçant une colonne pour chaque x_i et une colonne pour φ.
Pour chaque valuation v, l'entrée correspondant à x_i est $v(x_i)$ et l'entrée correspondant à φ est $[\![\varphi]\!]_v$.

3.1.7 Exemple : table de vérité de $p \leftrightarrow q$

$p \leftrightarrow q = (p \to q) \wedge (q \to p)$

p	q	$p \to q$	$q \to p$	$p \leftrightarrow q$
F	F	V	V	V
F	V	V	F	F
V	F	F	V	F
V	V	V	V	V

Remarque : Ici on a considéré des colonnes supplémentaires pour des sous-formules pour simplifier le calcul.

3.1.8 Remarque

Construire une table de vérité est un algorithme simple pour déterminer si une formule est satisfiable / une tautologie / une antilogie.

Cependant, si φ a n variables distinctes, alors il y a 2^n lignes dans sa table de vérité.

De plus, étant donné v, déterminer $[\![\varphi]\!]_v$ se fait en temps $\mathcal{O}(|\varphi|)$ et φ peut avoir au plus $|\varphi| + 1$ variables distinctes.

Cela donne donc un algorithme de complexité $\mathcal{O}(|\varphi|\, 2^{|\varphi|})$

3.1.9 Proposition

Il y a $2^{(2^n)}$ tables de vérités distinctes pour des formules à n variables distinctes.

□ Démonstration :

Il y a 2^n lignes dans une table et pour chaque ligne on a le choix entre 2 valeurs de vérité ■