Chapitre 4

Logique propositionnelle

4.1 Introduction

Les objectifs de la logique sont de
- Traiter formellement les notions de vérité et fausseté
- Formaliser et justifier le raisonnement logique intuitif, la déduction logique
- Permettre le raisonnement (humain ou automatique) dans des cas non intuitifs, complexes

L'élément de base de la logique des propositions est la ... proposition. Il s'agit d'un énoncé qui peut être vrai ou faux. Par exemple

> Jean mange une pomme
>
> Albertine pense qu'il pleuvra demain
>
> Tous les nombres impairs sont des nombres premiers
>
> Si le moteur n'est pas en marche et la phares sont allumés et la porte avant gauche est ouverte alors l'alarme sonne

Dans le langage formel de la logique ces énoncés seront représentés par des variables (p, q, r, etc.). Remarquons que certaines phrases de la langue courante ne sont pas de propositions, elles ne sont ni vraies ni fausses :

> Sortez immédiatement !
>
> Demain il pleuvra.
>
> Ce livre est-il bon ?

Aspects déductifs et sémantiques de la logique

On peut appréhender une logique soit sous son aspect déductif (ou syntaxique) soit sous son aspect sémantique.

Dans l'approche syntaxique on s'intéresse à la notion de *preuve formelle*. À partir d'axiomes et d'hypothèses on déduit de nouveaux faits en appliquant un ensemble de règles d'inférence.

Par exemple, la règle dite du *modus ponens* nous permet de déduire à partir des deux énoncés

S'il pleut alors il y a des nuages.

Il pleut.

l'énoncé

Il y a des nuages.

Dans l'approche sémantique on s'intéresse à évaluer la vérité ou fausseté de certains énoncés dans certaines situations, à voir si un énoncé est une conséquence logique d'un autre, à vérifier la consistance d'un ensemble d'énoncés. Par exemple, est-il possible que les deux énoncés

Paul ne mange pas de chocolat

Paul mange du pain ou du chocolat

soient vrais simultanément ?

Logique et informatique

La logique apparaît en de nombreuses circonstances en systèmes d'information.

- Spécification formelle des systèmes : invariants, conditions de fonctionnement, pré et post conditions ; détecter les incohérences, prouver des propriétés.
- Vérification du logiciel : preuves de programmes, preuves des systèmes parallèles par model checking
- Logique des données : interprétation logique des langages d'accès aux bases de données, expression des contraintes d'intégrité.
- Systèmes basés sur la connaissance : systèmes experts, déduction automatique, programmation logique.
- Logique de la connaissance : définition formelle des concepts, définition de théories (le temps, l'espace, les matériaux, la physique, la gestion, ...), modélisation logique

En outre, les fondements théoriques de l'informatique et de la logique formelle sont fortement liés. Notre compréhension profonde de la logique, en particulier les fameux théorèmes d'incomplétude de Gödel, repose sur la notion de calcul (que ce soit par les fonctions récursives ou les machines de Turing). Inversement, c'est un problème de logique (la satisfiabilité) qui apparait dans le théorème de Cook qui est essentiel en théorie de la complexité algorithmique.

4.2 Syntaxe des formules logiques

Un vocabulaire logique est composé
- des symboles (connecteurs) : $\wedge, \vee, \Rightarrow, \Leftrightarrow, \neg, (,)$
- des symboles de variables propositionnelles : $a, b, c, \ldots, p, q, r, s$, etc.

Pour construire une formule correcte on applique les règles de grammaire

$$\begin{array}{rcl} \text{Formule} & \rightarrow & \text{Variable} \\ & | & \text{Formule} \wedge \text{Formule} \\ & | & \text{Formule} \vee \text{Formule} \\ & | & \text{Formule} \Rightarrow \text{Formule} \\ & | & \text{Formule} \Leftrightarrow \text{Formule} \\ & | & \neg\ \text{Formule} \\ & | & (\text{Formule}\ ) \\ \text{Variable} & \rightarrow & \text{a} \mid \text{b} \mid \ldots \mid \text{x} \mid \text{y} \mid \text{z} \end{array}$$

Exemple 4.1. Quelques formules

$$p$$

$$q \Rightarrow p$$

$$\neg\neg p$$

$$\neg(p \vee (r \wedge \neg s) \vee t)$$

$$(q \Rightarrow p) \Leftrightarrow (s \Rightarrow p)$$

La grammaire ci-dessus est ambigüe, il y a, par exemple, deux manière d'analyser

$$p \wedge q \vee r$$

On utilise des parenthèses pour lever toute ambiguïté :

$$(p \wedge q) \vee r$$

n'a qu'un seul arbre syntaxique.

Il existe d'autres notations pour les connecteurs logique, la table ci-dessous donne les équivalences

$\wedge$	$\&$
$\Rightarrow$	$\rightarrow$ ou $\supset$
$\Leftrightarrow$	$\leftrightarrow$
$\neg x$	$\sim x$ ou $\bar{x}$

4.3 Sémantique

Etant donné un langage $\mathcal{L}$ on veut attribuer un sens à chaque chaine w de $\mathcal{L}$.
Les sens est une valeur prise dans un ensemble D appelé domaine d'interprétation.
Dans le cas de la logique $D = \{\mathbf{v}, \mathbf{f}\}$ (vrai, faux)

Principe de l'interprétation des formules

Objectif : attribuer une valeur logique vrai (v) ou faux (f) à chaque formule

Définition 4.1. Une interprétation d'un langage logique est une fonction I qui associe à chaque formule une valeur de vérité prise dans l'ensemble $\{\mathbf{v}, \mathbf{f}\}$

Pour définir une interprétation I on va

1. Fixer une interprétation $I(x)$ de chaque variable propositionnelle x
2. Appliquer les règles d'interprétation pour interpréter les formules.

Règles d'interprétation d'une formule :

Expression	$I(Expression)$
$P \wedge Q$	$\mathbf{v}$ si $I(P) = \mathbf{v}$ et $I(Q) = \mathbf{v}$, $\mathbf{f}$ si l'un des deux est faux
$P \vee Q$	$\mathbf{v}$ si $I(P) = \mathbf{v}$ ou $I(Q) = \mathbf{v}$ $\mathbf{f}$ si tous les deux sont faux
$P \Rightarrow Q$	$\mathbf{v}$ si $I(Q) = \mathbf{v}$ quand $I(P) = \mathbf{v}$, donc $\mathbf{f}$ ssi $I(P) = \mathbf{v}$ et $I(Q) = \mathbf{f}$
$P \Leftrightarrow Q$	$\mathbf{v}$ si $I(P) = I(Q)$, $\mathbf{f}$ s'ils sont différents
$\neg P$	$\mathbf{v}$ si $I(P) = \mathbf{f}$, $\mathbf{f}$ sinon

Exemple 4.2. On définit l'interprétation I des variables par

$$I(p) = \mathbf{v}; I(q) = \mathbf{f}; I(r) = \mathbf{v}$$

On a alors

$$\begin{aligned} I(p \wedge q) &= \mathbf{f} \\ I(p \vee \neg p) &= \mathbf{v} \\ I(p \wedge (\neg r \vee q)) &= \mathbf{f} \\ I(p \Rightarrow q) &= \mathbf{f} \\ I(q \Rightarrow p) &= \mathbf{v} \end{aligned}$$

Exemple 4.3. Une autre interprétation

Si

$$J(p) = \mathbf{f}; J(q) = \mathbf{v}; J(r) = \mathbf{f}$$

les formules de l'exemple précédent s'interpréteront comme

$$\begin{aligned} J(p \wedge q) &= \mathbf{f} \\ J(p \vee \neg p) &= \mathbf{v} \\ J(p \wedge (\neg r \vee q)) &= \mathbf{f} \\ J(p \Rightarrow q) &= \mathbf{v} \\ J(q \Rightarrow p) &= \mathbf{f} \end{aligned}$$

Modèles et satisfiabilité

On considère qu'une interprétation I est un "monde possible".
Pour une formule Φ donnée existe-t-il un monde dans lequel Φ est vraie ?

Définition 4.2. Un *modèle* d'une formule Φ est une interprétation M telle que

$$M(\Phi) = \mathbf{v}$$

Un modèle d'un ensemble de formules $F = \{\Phi_1, \ldots, \Phi_k\}$ est une interprétation M telle que

$$M(\Phi_1) = \cdots = M(\Phi_k) = \mathbf{v}$$

Il peut exister zéro, un ou plusieurs modèles pour une formule ou un ensemble de formules.

Définition 4.3. S'il existe au moins un modèle de F on dit que F est *satisfiable* (ou consistant ou non contradictoire).sinon F est *inconsistante* .

Exemple 4.4. L'ensemble $F = \{p \wedge q, q \vee r\}$ est satisfiable

Il y a deux modèles :

1. <+-> $I_1 : I_1(p) = \mathbf{v}, I_1(q) = \mathbf{v}, I_1(r) = \mathbf{v}$
2. <+-> $I_2 : I_2(p) = \mathbf{v}, I_2(q) = \mathbf{v}, I_2(r) = \mathbf{f}$

Exemple 4.5. L'ensemble $F = \{\neg p \vee q, r \vee \neg q, p, \neg r\}$ est inconsistant

Dans tout modèle I on devrait avoir
- $I(p) = \mathbf{v}$
- $I(q) = \mathbf{v}$ car $I(\neg p \vee q) = \mathbf{v}$ et $I(p) = \mathbf{v}$
- $I(r) = \mathbf{v}$ car $I(r \vee \neg q) = \mathbf{v}$ et $I(q) = \mathbf{v}$
- $I(r) = \mathbf{f}$ car $I(\neg r) = \mathbf{v}$

d'où contradiction

Définition 4.4. Une *tautologies* est une formule vraie quelle que soit l'interprétation. Par exemple :

$$P \vee \neg P$$
$$P \Rightarrow P$$
$$((P \Rightarrow Q) \wedge P) \Rightarrow Q$$
$$(P \vee Q) \Leftrightarrow \neg(\neg P \wedge \neg Q)$$

etc.

Les tautologies sont des vérités universelles qui ne dépendent pas de l'état du monde.

Complexité de la recherche de modèles

Etant donné une formule avec n variables, il y a 2^n interprétations possibles. Par exemple, une formule à 100 variables possède 1267650600228229401496703205376 interprétations.

Si on essaye toutes les interprétations possibles, la complexité en temps sera donc exponentielle et les temps effectifs de calculs seront énormes (infaisables). Si on teste 1 milliard d'interprétations par seconde on en aura pour 40 196 936 841 331 années avec 100 variables.

Or on ne connait pas, à l'heure actuelle, d'algorithme qui fournisse à coup sûr une réponse en temps non exponentiel. Et ce problème est aussi difficile que de nombreux autres pour lesquelles on ne connait pas d'algorithme non exponentiel (on dit que ce problème est NP-complet).

Cependant, il existe des algorithmes, basés sur celui de Davis-Putnam-Logemann-Loveland, qui peuvent trouver une solution rapidement dans de nombreux cas, mais pas toujours (voir 4.4).

4.4 Equivalences et formes normales

Equivalence

Deux formules E et F sont équivalentes si pour toute interprétation elles prennent les même valeurs de vérité.

p.ex.

$$p \wedge q \equiv \neg(p \Rightarrow \neg q)$$

On peut voir que : E et F sont équivalentes si la formule

$$E \Leftrightarrow F$$

est une tautologie (est toujours vrai)

Les équivalences permettent des manipulations syntaxiques qui préservent la sémantique.

Quelques équivalences utiles

Utiles pour simplifier les formules ou les restructurer.

A, B, C, ...sont des variables ou des formules quelconques

double négation :

$$\neg(\neg A) \equiv A$$

associativité

$$(A \wedge B) \wedge C \equiv A \wedge (B \wedge C); (A \vee B) \vee C \equiv A \vee (B \vee C)$$

commutativité

$$A \wedge B \equiv B \wedge A; A \vee B \equiv B \vee A$$

idempotence

$$A \wedge A \equiv A; A \vee A \equiv A$$

distributivité-ou

$$A \vee (B \wedge C) \equiv (A \vee B) \wedge (A \vee C)$$

distributivité-et :

$$A \wedge (B \vee C) \equiv (A \wedge B) \vee (A \wedge C)$$

De Morgan

$$\neg(A \wedge B) \equiv \neg A \vee \neg B; \neg(A \vee B) \equiv \neg A \wedge \neg B$$

implication-ou

$$A \Rightarrow B \equiv \neg A \vee B$$

double implication

$$A \Leftrightarrow B \equiv (A \Rightarrow B) \wedge (B \Rightarrow A)$$

donc $\Rightarrow$ et $\Leftrightarrow$ ne sont pas essentiels, toute formule peut être réécrite sans utiliser ces deux connecteurs.

tiers exclu

$$A \wedge \neg A \equiv faux$$

$$A \vee \neg A \equiv vrai$$

Formes normales

Toute formule peut se réécrire sous la forme

$$C_1 \wedge C_2 \wedge \cdots \wedge C_n$$

où chaque C_i (appelé clause) est elle-même de la forme

$$p_1 \vee p_2 \vee \cdots \vee p_m \vee \neg q_1 \vee \neg q_2 \vee \cdots \vee \neg q_r$$

où les p_j et q_k sont des atomes.
Cette forme est appelée forme normale conjonctive .
Il y a un algorithme pour mettre en FNC
La FNC est nécessaire pour pouvoir appliquer certains algorithmes.

Exemple 4.6. On veut mettre la formule $((b \vee c) \Rightarrow a) \vee d$ sous forme normale conjonctive

1. Appliquer les équivalence double-implique et implique-ou pour supprimer les $\Rightarrow$ et $\Leftrightarrow$.

$$\equiv (\neg(b \vee c) \vee a) \vee d$$

2. Appliquer De Morgan pour «descendre» les négation près des variables.

$$\equiv ((\neg b \wedge \neg c) \vee a) \vee d$$

3. Appliquer la distributivité pour descendre les $\vee$ et remonter les $\wedge$

$$\equiv ((\neg b \vee a) \wedge (\neg c \vee a)) \vee d$$

$$\equiv ((\neg b \vee a) \vee d) \wedge ((\neg c \vee a)) \vee d)$$

4. Appliquer l'associativité des $\vee$ et $\wedge$

$$\equiv (\neg b \vee a \vee d) \wedge (\neg c \vee a \vee d)$$

Il existe aussi une forme normale *disjonctive* :

$$D_1 \vee D_2 \vee \cdots \vee D_m$$

où chaque D_i est lui même de la forme

$$r_1 \wedge r_2 \wedge ... \wedge r_u \wedge \neg s_1 \wedge \neg s_2 \wedge ... \wedge \neg s_v$$

où les r_j et s_k sont des atomes.

Utilisation de la FNC pour la satisfiabilité

Algorithme de Davis-Putnam-Logemann-Loveland
On considère une formule Φ qui a été mise en FNC.
L'algorithme consiste essentiellement à

1. repérer les clauses littérales (une seule variable), fixer la valeur de la variable correspondante, mettre à jour les autres clauses
2. choisir un littéral k (au hasard ou à l'aide d'une heuristique)
3. appliquer l'algorithme sur $\Phi \wedge k$ et si on n'a pas trouvé de modèle, l'appliquer sur $\Phi \wedge \neg(k)$

Si à un moment donné la formule n'est composée que de littéraux et qu'elle est consistante, on a trouvé un modèle. Si par contre on a généré une clause vide on est sûr de ne pas trouver de modèle.
[fragile]

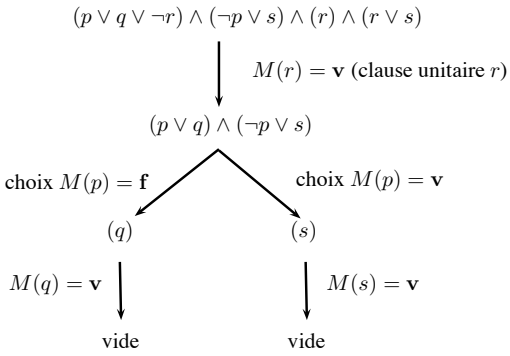
Algorithme DPLL

entrée : une formule F en FNC, un modèle partiel M
sortie : *faux* si on n'a pas trouvé de modèle , *vrai* si on a trouvé + le modèle
si F est vide
    retourner *vrai* et M
si F contient une clause vide :
    retourner *faux*
pour toute clause unitaire (x) ou $(\neg x)$de F :
    fixer $M(x) = \mathbf{v}$(resp. $\mathbf{f}$)
    $F = \textsc{PropagationValeur}(x, F)$
K := choisir un littéral de F
retourner DPLL$(F \wedge K)$ ou sinon DPLL$(F \wedge \neg K\ )$

Propagation des valeurs

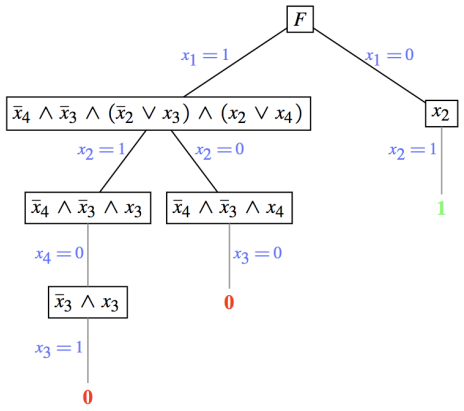
Si on a fixé $M(x) = \mathbf{v}$
    supprimer toutes les clauses ou x apparaît positivement
    enlever $\neg x$ de toutes les clauses où il apparaît
Si on a fixé $M(x) = \mathbf{f}$
    supprimer toutes les clauses ou $\neg x$ apparaît
    enlever x de toutes les clauses où il apparaît positivement

Exemple 1



Exemple 2 [1]

$$F = (x_1 \vee x_2) \wedge (\neg x_1 \vee \neg x_4) \wedge (\neg x_1 \vee \neg x_3) \wedge (\neg x_1 \vee \neg x_2 \vee x_3) \wedge (\neg x_1 \vee x_2 \vee x_4)$$



Déroulement de l'algorithme avec essais-erreurs ($\overline{x}$ signifie $\neg x$)

4.5 Algèbre de Boole

Opération. Une autre notation (algébrique)

1. Frederic Koriche. Satisfiabilité Propositionnelle. Université Montpellier II. http ://www.lirmm.fr/ koriche/W-Docs/Logique-Cours1.pdf

Formule logique	Algèbre de Boole
$A \vee B$	$A + B$
$A \wedge B$	$A \cdot B$
$\neg A$	$\overline{A}$
f	0
v	1

Donc
$p \Rightarrow q$ devient $\overline{p} + q$

Quelques formules booléennes

$$0 + 0 = 0, 0 + 1 = 1 + 0 = 1, 1 + 1 = 1$$

$$0 \cdot 0 = 0, 0 \cdot 1 = 1 \cdot 0 = 0, 1 \cdot 1 = 1$$

clause : $x_1 + ... + x_n + \overline{y_1} + ... + \overline{y_n}$
forme normale conjonctive : $C_1 \cdot C_2 \cdot \cdots \cdot C_k$
monome : $x_1 \cdot \cdots \cdot x_n \cdot \overline{y_1} \cdot \cdots \cdot \overline{y_n}$
forme normale disjonctive : $M_1 + M_2 + \cdots + M_r$

4.6 Conséquence logique

Intuitivement, dire que A est une conséquence logique de B c'est dire que

A est nécessairement vraie lorsque B est vrai

La notion d'interprétation nous permet de formaliser cette intuition en disant que

Si B est vraie pour une interprétation I alors A est vraie dans I

Définition 4.5. ϕ est une *conséquence logique* de $\{f_1, f_2, \ldots, f_n\}$ si tout modèle de $\{f_1, f_2, \ldots, f_n\}$ est forcément un modèle de ϕ

C-à-d si pour toute interprétation I telle que $I(f_1) = I(f_2) = \cdots = I(f_n) = \mathbf{v}$ on a $I(\phi) = \mathbf{v}$,
Notation $\{f_1, f_2, \ldots, f_n\} \models \phi$

Exemple 4.7. On part de l'ensemble de formules $F = \{p \Rightarrow q, q \Rightarrow r\}$.

Vérifions que $p \Rightarrow r$ est une conséquence logique de F et que $r \vee p$ ne l'est pas.
Il y a quatre modèles de F, pour chaque modèle on évalue nos deux formules :

Modèles de F	p	q	r	$p \Rightarrow r$	$\neg r \vee p$
I_1	v	v	v	v	v
I_2	f	v	v	v	f
I_3	f	f	v	v	f
I_4	f	f	f	v	v

$p \Rightarrow r$ étant vraie dans chaque modèle de F on en déduit. $F \models p \Rightarrow r$. Par contre $r \vee p$ est fausse pour certains modèles donc $F \not\models r \vee p$.

4.7 Déduction

L'approche purement sémantique, basée sur la recherche de modèles n'est en général pas praticable. Si l'on veut vérifier que $E \models f$ il faut

1. trouver tous les modèles de E
2. pour chaque modèle M de E, vérifier que $M(f) = v$

Si E utilise n variables atomiques, le nombre de modèles potentiels est 2^n . Le temps de vérification croit donc exponentiellement avec le nombre de variables.

D'autre part, cette approche n'aide pas à trouver des formules qui sont des conséquences logiques de E.

L'approche syntaxique (déductive) a pour objet de calculer les conséquences logiques par l'application de règles d'inférence. Pour cela on construit des systèmes formels d'inférence composés d'axiomes (formules) et de règles d'inférence.

Dans le cas de la logique on veut des systèmes qui produisent des formules qui sont des conséquences logiques des formules de départ. Le critère de consistance du système est donc le fait de produire uniquement des conséquences logiques des axiomes.

Il existe plusieurs systèmes d'inférence de ce type pour la logique des proposition : système de Hilbert, système de déduction naturelle, principe de résolution de Robinson, etc.

Système d'inférence naturelle

Dans ce système il n'y a pas d'axiomes, uniquement des règles d'inférence.

On notera

$$\frac{f_1, \ldots, f_n}{g}$$

le fait qu'on peut produire g à partir de $f_1, \ldots, f_n$ (inférence directe).

La notation $[f \rightarrow \ldots \rightarrow g]$ représente le fait qu'on peut produire g à partir de f par une succession d'inférences.

Remarquons tout d'abord que dans tout système d'inférence on peut utiliser toutes les équivalences connues pour produire de nouvelles formules. Car si

$$f \equiv g$$

on a forcément

$$f \models g$$

Par exemple, de $\neg(p \vee q)$ on peut déduire $\neg p \wedge \neg q$.

Règles pour $\wedge$

Introduction

$$\frac{f, g}{f \wedge g}$$

Élimination

$$\frac{f \wedge g}{f}$$

$$\frac{f \wedge g}{g}$$

Règles d'élimination de $\Rightarrow$

Modus ponens

$$\frac{f \Rightarrow g, f}{g}$$

Modus tollens

$$\frac{f \Rightarrow g, \neg g}{\neg f}$$

Exemple 4.8. Application des premières règles du système naturel.

On part de trois formules :

1. $p \wedge q$
2. $q \Rightarrow r$
3. $q \Rightarrow s$

On applique les règles :

4. q $\wedge$-élimination sur 1.
5. r modus ponens 2 et 4
6. s modus ponens 3 et 4
7. $r \wedge s$ $\wedge$-introduction 5 et 6

Négation

Introduction

$$\frac{f}{\neg\neg f}$$

Élimination

$$\frac{\neg\neg f}{f}$$

Règle d'introduction de $\Rightarrow$

Principe de déduction

$$\frac{[f \rightarrow \ldots \rightarrow g]}{f \Rightarrow g}$$

Si on peut produire g (conclusion) à partir de f (hypothèse) alors on peut produire $f \Rightarrow g$. Autrement dit : si on peut prouver g à partir de f alors on a démontré le théorème $f \Rightarrow g$

Exemple 4.9. Application du principe de déduction

[	(début déduction)
p	hypothèse
$\neg\neg p$	double négation
$\neg\neg\neg\neg p$	double négation (conclusion)
]	(fin déduction)
$p \Rightarrow \neg\neg\neg\neg p$	principe de déduction, introduction de $\Rightarrow$

Au cours d'une preuve on peut appliquer le principe de déduction de manière imbriquée On peut «importer» n'importe quelle formule déjà produite à l'intérieur d'une dérivation.

Exemple 4.10. Déductions imbriquées

[	début
p	hypothèse
[	début, 2
q	hypothès
p	importation
$p \wedge q$	introduction $\wedge$
]	fin 2
$q \Rightarrow (p \wedge q)$	résultat déduction 2
]	
$p \Rightarrow (q \Rightarrow (p \wedge q))$	résultat déduction

[	
$(p \Rightarrow q) \wedge (\neg p \Rightarrow q)$	hypothèse
$p \Rightarrow q$	élimination $\wedge$
$\neg q \Rightarrow \neg p$	contraposition
$\neg p \Rightarrow q$	dissociation
$\neg q \Rightarrow \neg\neg p$	contraposition
[	
$\neg q$	hypothèse
$\neg q \Rightarrow \neg\neg p$	importation
$\neg\neg p$	modus ponens
$\neg q \Rightarrow \neg p$	importation
$\neg p$	modus ponens
$\neg\neg p \wedge \neg p$	introduction $\wedge$
$\neg(\neg p \vee p)$	De Morgan
]	
$\neg q \Rightarrow \neg(\neg p \vee p)$	introduction $\Rightarrow$
$(\neg p \vee p) \Rightarrow q$	contraposition
[	
p	hypothèse
]	
$p \Rightarrow p$	introduction $\Rightarrow$
$\neg p \vee p$	échange ou-implique
q	modus ponens
]	
$((p \Rightarrow q) \wedge (\neg p \Rightarrow q)) \Rightarrow q$	intro $\Rightarrow$

TABLE 4.1 – Une preuve de $((p \Rightarrow q) \wedge (\neg p \Rightarrow q)) \Rightarrow q$

Exemple 4.11. Le tableau 4.1 montre une preuve plus complexe.

Règles pour le $\vee$

Introductions

$$\frac{f}{f \vee g}$$

$$\frac{f}{g \vee f}$$

avec g une formule quelconque

Élimination

$$\frac{f \vee g, [f \rightarrow \ldots \rightarrow h], [g \rightarrow \ldots \rightarrow h]}{h}$$

Règles pour la contradiction $\perp$

Introduction

$$\frac{f, \neg f}{\perp}$$

où $\perp$ est la formule dont l'interprétation est toujours *faux*, donc la formule impossible à satisfaire.

Élimination

$$\frac{[f \rightarrow \ldots \rightarrow \perp]}{\neg f}$$

$$\frac{[\neg f \rightarrow \ldots \rightarrow \perp]}{f}$$

C'est le principe des preuves par l'absurde.

Complétude et Consistance des règles de déduction

Idéalement un système formel doit satisfaire les deux critères de consistance et de complétude. Ces deux notions font le lien entre syntaxe et sémantique.

On se rappelle qu'un théorème est une formule qu'on peut produire par application des règles à partir des axiomes. La « théorèmité » d'une formule est donc une notion syntaxique. Par contre la notion de conséquence logique est sémantique, elle est définie en dehors de tout système d'inférence formel. C'est cette notion qui nous servira à vérifier la consistance et la complétude d'un système d'inférence.

Définition 4.6. Un système d'inférence est *consistant* si tous les théorèmes sont des conséquences logiques des formules des axiomes (tout ce qui est démontrable est « vrai »). Ou de manière équivalence, si on peut produire g à partir de l'ensemble de formules F , noté $F \vdash g$, alors g est une conséquence de F ($F \models g$)

Un système d'inférence est *complet* s'il permet de produire *toutes* les conséquence logiques des axiomes, c'est-à-dire si toue conséquence des axiomes est un théorème (tout ce qui est vrai est démontrable). Ou, de manière équivalente, $F \models g$ entraîne $F \vdash g$

Le système d'inférence naturel est consistant et complet.

4.8 Principe de résolution (Robinson)

Cette règle d'inférence a été inventée par Robinson en 1965. Elle est surtout utilisée dans les systèmes de preuve automatique (démonstrateurs de théorème, systèmes de programmation logique). C'est l'unique règle du système, il n'y a donc pas de problème de choix de la bonne règle à appliquer. Par contre on peut l'appliquer de multiples manières.

Cette règle travaille uniquement sur des clauses. Il faut donc commencer par tout mettre en forme normale conjonctive.

$$C_1 \wedge C_2 \wedge ... \wedge C_n$$

où chaque clauseC_i est de la forme

$$L_1 \vee \cdots \vee L_m.$$

Les L_j sont des *littéraux* de la forme

$$V \text{ ou } \neg V$$

où V est une variable.

La règle elle-même s'énonce comme suit :

Si on a deux clauses

$$C_1 = (p \vee L_1 \vee L_2 \vee ... \vee L_m),$$

$$C_2 = (\neg p \vee M_1 \vee M_2 \vee ... \vee M_n)$$

on peut déduie le résolvant de C_1 et C_2 qui est la clause

$$L_1 \vee L_2 \vee ... \vee L_m \vee M_1 \vee M_2 \vee ... \vee M_n$$

En d'autre termes, si un littéral et son complément apparaissent dans deux clauses on peut déduire une nouvelle clause à partir de celles-ci.

Exemple 4.12. Application du principe de résolution

$$C_1 = (p \vee q \vee \neg r \vee s)$$

$$C_2 = (q \vee \neg p \vee t)$$

Résolution sur p et $\neg p$.

Résolvant :

$$(q \vee \neg r \vee s \vee q \vee t),$$

Exemple 4.13. On veut prouver

$$\{(p \vee t) \Rightarrow q, r \Rightarrow (t \vee s), (q \wedge t) \Rightarrow u, p, \neg s, r\} \models u$$

On met les formules sous forme de clauses :

1. $(p \vee t) \Rightarrow q \equiv \neg(p \vee t) \vee q \equiv (\neg p \wedge \neg t) \vee q \equiv (\neg p \vee q) \wedge (\neg t \vee q)$ ce qui donne deux clauses : $(\neg p \vee q)$ et $(\neg t \vee q)$
2. $r \Rightarrow (t \vee s) \equiv \neg r \vee (t \vee s) \equiv \neg r \vee t \vee s$
3. $(q \wedge t) \Rightarrow u \equiv \neg(q \wedge t) \vee u \equiv (\neg q \vee \neg t) \vee u \equiv \neg q \vee \neg t \vee u$

On applique ensuite le principe de résolution à partir des six clauses obtenues :

1. $\neg p \vee q$
2. $\neg t \vee q$
3. $\neg r \vee t \vee s$
4. $\neg q \vee \neg t \vee u$
5. p
6. $\neg s$
7. r
8. q (résolution 1 et 5 sur p)
9. $t \vee s$ (résolution 3 et 7 sur r
10. t (résolution 6 et 9 sur s)
11. $\neg t \vee u$ (résolution 4 et 8 sur q)
12. u (résolution 10 et 11 sur t)

Il faut faire attention à ne pas prendre de raccourcis. On ne peut pas appliquer le principe sur deux variables en même temps. Si on a

$$C_1 = (p \vee \neg q \vee \neg r \vee s)$$

$$C_2 = (q \vee \neg p \vee t)$$

on peut déduire

$$\neg q \vee \neg r \vee s \vee q \vee t \text{ (résolution sur p)}$$

ou

$$p \vee \neg r \vee s \vee \neg p \vee t \text{ (résolution sur q)}$$

mais pas

$$\neg r \vee s \vee t \text{ ! ("résolution" sur p et q)}$$

L'utilisation pratique du principe de résolution se base sur le théorème suivant :

Théorème 4.1. *Le résolvant C est satisfiable si et seulement si les clauses parentes C_1 et C_2 le sont (simultanément)*

À partir de ce théorème on peut créer une procédure de résolution pour vérifier si un ensemble S de clauses est satisfiable.

1. On définit $S_0 := S$
2. On calcule une séquence $S_1, S_2, \ldots$ avec la règle
   (a) soient C_1 et C_2 deux clauses de S_i dont le résolvant est C
   (b) on définit $S_{i+1} := S_i \cup \{C\}$
   (c) si au cours du processus on trouve un $C = []$ alors S_0 était inconsistant
   (d) si $S_{i+1} = S_i$ pour tout choix de C_1 et C_2, alors S_0 était satisfiable

Une dérivation de [] à partir de S s'appelle une réfutation de S.

La technique de *preuve par* réfutation consiste à prouver que A est une conséquence logique de S en réfutant
$S_0 = S \cup \{\neg A\}$.

Exemple 4.14. Soit $S = \{p, \neg p \vee q, \neg p \vee \neg q \vee r\}$. On veut montrer que $S \models r$.

1. $S_0 = \{p, \neg p \vee q, \neg p \vee \neg q \vee r, \neg r\}$
2. $S_1 = \{p, \neg p \vee q, \neg p \vee \neg q \vee r, \neg r, \neg p \vee \neg q\}$
3. $S_2 = \{p, \neg p \vee q, \neg p \vee \neg q \vee r, \neg r, \neg p \vee \neg q, \neg p\}$
4. $S_3 = \{p, \neg p \vee q, \neg p \vee \neg q \vee r, \neg r, \neg p \vee \neg q, \neg p, []\}$

donc $S \models r$

4.9 Principes de modélisation logique

On se rappelle qu'une interprétation d'un langage logique est une fonction I qui associe à chaque formule une valeur de vérité prise dans l'ensemble $\{\mathbf{v}, \mathbf{f}\}$. Pour définir une interprétation I on doit commencer par fixer une interprétation $I(x)$ de chaque variable propositionnelle x puis on applique les règles d'interprétation pour interpréter les formules formules.

Lorsqu'on modélise un système, c'est l'état de ce système, qui fournit l'interprétation des variables. Plus précisément, on représente certaines propriétés du système, qui peuvent êtres vraies ou fausses, par des variables logiques (s'il y a n variables on peut représenter 2^n états du système)

Exemple 4.15. On s'intéresse au système de sécurité d'une automobile. Ce système peut être caractérisé par les variables suivantes :

variable	propriété représentée
m	le moteur est en marche
p1	la porte 1 est ouverte
p2	la porte 2 est ouverte
p3	la porte 3 est ouverte
p4	la porte 4 est ouverte
c	la clé de contact est présente
r	la voiture roule
ap	l'alerte porte est en marche
ac	l'alerte clé de contact est en marche

Parmi tous les états possibles du système (valeurs des variables) seul un certain nombre correspond à des états considérés comme corrects (ou admissibles ou souhaitables). Pour distinguer les états corrects des autres on utilise la notion de modèle logique, tel que nous l'avons vu précédemment.

Un modèle d'un ensemble F de formules est une interprétation M qui rend vraies toutes les formules de F. On dira que le système est dans un état correct si l'interprétation des variables qu'il fournit (à un instant donné) est un modèle d'un ensemble Fde formules fixés. Les formules de F constituent des invariants du systèmes (ce qui doit toujours rester vrai)

Exemple 4.16. Quelques invariant du système « Sécurité Automobile »

Lorsque la voiture roule toutes les portes doivent être fermées, sinon l'alerte porte doit être enclenchée.

$$f_1 = (r \wedge (p1 \vee p2 \vee p3 \vee p4)) \Rightarrow ap$$

Si le moteur est arrêté, que la porte 1 est ouverte et la clé de contact présente, l'alerte clé doit être enclenchée

$$f_2 = (\neg m \wedge p1 \wedge c) \Rightarrow ac$$

L'alerte clé ne doit s'enclencher que dans cette circonstance

$$f_3 = ac \Rightarrow (\neg m \wedge p1 \wedge c)$$

Les invariants fournissent donc une description compacte des états possibles du système. L'ensemble des invariants doit être consistant sinon cela signifierait qu'il n'y a aucun état du système qui est correct.

Dynamique du système

L'ensemble des invariant donne également des indications sur la dynamique du système. En effet, pour que le système reste dans un état correct toute opération ou évènement qui modifie une ou plusieurs variables n'est admissible que si elle fait passer d'un modèle à un autre (les invariants doivent effectivement rester vrais en toute circonstance)

Exemple 4.17. Considérons les trois états I_1, I_2, I_3 ci-dessous.

variable	I_1	I_2	I_3
m	v	v	v
p1	f	f	v
p2	f	f	f
p3	f	f	f
p4	f	**v**	**v**
c	v	v	v
r	v	v	v
ap	f	f	**v**
ac	f	f	f

I_1 et I_3 sont des modèles pour les invariants f_1, f_2, f_3 alors que I_2 n'en est pas un. Par conséquent l'opération qui passe de I_1 à I_2 (ouvrir la porte 4 pendant que la voiture roule) est interdite. Par contre le passage de I_1 à I_3 (simultanément ouvrir la porte et mettre en marche l'alarme) est possible.

Utilisation de la déduction

On peut employer un système d'inférence pour trouver quelles sont les conséquences logiques des invariants. En terme de modélisation, le fait qu'un formule g soit une conséquence des invariants signifie que g sera toujours vraie dans tous les (bons) états du système. Ou inversement, que $\neg g$ n'est vrai dans aucun état acceptable.

Exemple 4.18. Dans le cas de l'automobile, on veut vérifier que les alarmes porte et clé ne peuvent jamais être en marche simultanément. On doit donc prouver $\neg(ap \wedge ac)$ à partir des invariants. Par réfutation on doit prouver que $\{f_1, f_2, f_3, ap \wedge ac\}$ n'est pas satisfiable. Si on applique la procédure de résolution (section 4.8) on s'aperçoit que c'est ensemble est en fait satisfiable. Donc il est possible que les deux alertes fonctionnent en même temps. Cela se produit, par exemple, si l'automobile roule, la porte 1 est ouverte et le moteur est arrêté, ce qui n'est pas interdit par les axiomes.

On peut également se poser la question des conséquences d'une action. Que se passe-t-il si on met le système dans un état où un ensemble F de formules devient vrai ? Pour obtenir la réponse il faut chercher les conséquences logiques de l'ensemble $\mathit{Invariants} \cup F$.

Exemple 4.19. On a un système de tuyaux et de vannes. L'état de chaque vanne est représenté par une variable qui est vrai si la vanne est ouverte, fausse si elle est fermée. Les invariants, correspondant à des règles de sécurité du système, sont :

si la vanne a est fermée, la vanne b doit être ouverte

1. $\neg a \Rightarrow b \equiv a \vee b$

   si b est ouverte, soit c soit d est ouverte

2. $b \Rightarrow (c \vee d) \equiv \neg b \vee c \vee d$

   si c est ouverte e est fermée

3. $c \Rightarrow \neg e \equiv \neg c \vee e$

   si d est ouverte f est ouverte

4. $d \Rightarrow f \equiv \neg d \vee f$

Quelles sont les conséquences de la fermeture des vannes a et f ?

On applique le principe de résolution à partir des invariants et de $\neg a$ et $\neg f$:

5.	$\neg a$	hypothèse
6.	$\neg f$	hypothèse
7.	$\neg d$	de 6 et 4 par résolution
8.	$\neg b \vee c$	de 7 et 2
9.	$a \vee c$	de 8 et 1
10.	c	de 5 et 9
11.	e	de 3 et 10
12.	b	de 1 et 5

Donc si a et f sont fermées, les vannes c, e et b doivent être ouvertes et la vanne d fermée pour respecter les invariants.

À partir de cet exemple on peut voir que ce type de modélisation et de calcul est particulièrement bien adapté à des situations où

- il y a beaucoup de variables (on peut même traiter des milliers de variables avec des outils informatiques)
- les variables sont binaires (vrai ou faux)
- les règles du domaine sont connues

Comme nous le verrons plus précisément avec la logique des prédicats, une des difficultés de la modélisation logique est de trouver : les axiomes du domaines. En effet, si l'on veut pouvoir faire des déductions intéressantes il est nécessaire d'ajouter des formules qui décrivent les règles implicites du domaine. Il s'agit de règles bien réelles mais qui sont rarement explicitées car elles sont « évidentes » pour tout être humain ou pour tout connaisseur du domaine en question. Dans l'exemple automobile on aurait du ajouter le fait que le moteur ne peut être en marche que si la clé de contact est présente.

Possibilités et limites de la modélisation propositionnelle La logique des propositions présente l'avantage de la décidabilité : il y a des algorithmes pour décider à coup sûr si un ensemble de formules est consistant ou si une formule est conséquence d'autres formules. Certains algorithmes sont même rapides dans biens des cas complexes. Cette décidabilité se paie par un pouvoir d'expression limité. En effet, en logique des propositions

il faut représenter sous forme de variables logiques (vrai/faux) chacune des propriétés de chacun des objets du systèmes, de même, chaque relation entre deux objets sera aussi représenté par une variable. Par exemple, si on a un petit système composé de 6 objets qui ont chacun 3 propriétés binaires il faudra $6 \times 3 = 18$ variables p_{11}, p_{12}, p_{13}, ..., p_{63}. Si, en plus, on veut représenter l'existence ou non d'une relation entre ces objets il faudra ajouter $5 + 4 + 3 + 2 + 1 = 15$ variables rel_{12}, rel_{13}, ..., rel_{16}, rel_{23}, ..., rel_{26}, ..., rel_{56}.

Vu cette prolifération de variables il devient extrêmement difficile d'exprimer des énoncés généraux tels que « il y a au moins un objet pour lequel la propriété 3 est vraie » : $p_{13} \vee p_{23} \cdots \vee p_{63}$ ou bien pire : « il y a au moins 2 objets qui ont la propriété 3 et qui sont en relation » :

$$(rel_{12} \wedge p_{13} \wedge p_{23}) \vee (rel_{13} \wedge p_{13} \wedge p_{33}) \vee \cdots \vee (rel_{56} \wedge p_{53} \wedge p_{63})$$

(une formule avec $15 \times 3 = 45$ variables. Bien que cela devienne très compliqué c'est encore possible. Par contre, si on ne sait pas combien il y a d'objets dans le système il deviennent carrément impossible d'exprimer de tels énoncés existentiels ou des énoncés universels comme « tout objet possède soit la propriété 1 soit la propriété 3 ».

Références

Cl. Benzaken. *Systèmes formels : introduction à la logique et à la théorie des langages* , Masson, Paris, 1991.

M. Huth, M. Ryan. *Logic in Computer Science*, Cambridge University Press, 2004.

Bundy, A. *The Computer Modelling of Mathematical Reasoning* , Academic Press, London, 1983.

Cori, R., Lascar, D. *Logique mathématique* , Masson, 1993.

D. Hofstadter. *Gödel, Escher, Bach : les brins d'une guirlande éternelle*, Dunod, 2008