INFZ21, Logiques du raisonnement valide

AUDIBERT Laurent [1]

4 mai 2004

1. DELIC - Université de Provence - 29 Avenue Robert SCHUMAN - 13621 Aix-en-Provence Cedex 1 - laurent.audibert@up.univ-aix.fr

Le but de la logique est de calculer des conclusions sûres. Le langage naturel s'avère trop imprécis et riche pour permettre des développements simples et rigoureux.

La logique est un outil pour parler et raisonner dans un domaine déterminé. Différents domaines ont différentes logiques. C'est un problème à la fois philosophique et mathématique que de savoir si une logique donnée est adéquate pour un domaine particulier.

Les logiques classiques (la logique des propositions et la logique des prédicats) ont été créées pour raisonner sur des objets mathématiques. Comme ces objets sont conceptuellement assez simple, ces logiques le sont aussi. Tout comme en mathématiques, les systèmes déductifs de la logique classique se limitent à la formalisation du raisonnement valide.

Un système formel de déduction de la logique classique est composé d'un ensemble de schémas d'axiomes et de règles d'inférence. Un tel système permet d'inférer des conclusions à partir de prémisses et définit donc une relation de déduction entre formules, notée $\vdash$. Cette relation possède les propriétés suivantes :

- **propriété de réflexivité** : $\{p_1, \ldots, p_n, q\} \vdash q$;
- **propriété de monotonie** : si $\{p_1, \ldots, p_n\} \vdash q$ alors $\{p_1, \ldots, p_n, r\} \vdash q$;
- **propriété de transitivité** : si $\{p_1, \ldots, p_n\} \vdash r$ et $\{p_1, \ldots, p_n, r\} \vdash q$ alors $\{p_1, \ldots, p_n\} \vdash q$.

Dans les expressions ci-dessus, $p_1, \ldots, p_n, q$ et r désignent des formules du langage logique considéré. Ces propriétés fondamentales traduisent formellement certaines exigences du raisonnement valide :

- Inférer une conclusion identique à l'une des prémisses est une opération valide ;
- Un résultat acquis n'est pas remis en cause par des résultats ultérieurs ;
- Des résultats intermédiaires peuvent être utilisés pour établir la validité d'une conclusion.

Table des matières

Chapitre 1

Logique des propositions

1.1 Introduction

Le but initial de la logique est de formaliser certaines relations qui peuvent apparaître de bon sens. Si je dis la phrase p : « il fait beau et je suis en vacances », cette phrase sera vraie si les deux propositions q : « il fait beau » et r : « je suis en vacances » sont vraies toutes les deux. Si l'une des deux est fausse, alors la phrase p sera fausse. Nous utilisons donc une interprétation intuitive du connecteur *et* qui nous permet de déterminer la valeur de vérité d'une phrase contenant deux sous-propositions reliées par *et* si nous connaissons la valeur de vérité de chacune des propositions. De même, nous utilisons un mécanisme semblable pour des phrases contenant les connecteurs *ou*, *donc*, . . . , c'est le principe de compositionalité.

Logique des propositions

Le but du calcul propositionnel est de donner un fondement formel à un ensemble restreint d'énoncés du langage. Nous utiliserons comme élément de base des propositions élémentaires (*i.e.* des énoncés déclaratifs) : « il fait beau », « je suis en vacances », etc. Ces propositions élémentaires seront soit vraies, soit fausses (principe du tiers-exclu). Pour former des phrases, à partir de ces propositions élémentaires, nous n'utiliserons que quatre connecteurs, *et*, *ou*, *donc*, *équivalent à*, ainsi que la négation (si la phrase « il fait beau » est vraie, alors « il ne fait pas beau » est fausse).

Pas de convention : divergence des interprétations

Plaçons-nous dans le contexte suivant : « Il aime les fraises et la chantilly ».
Dites si les propositions suivantes sont vraies ou fausses.

1. « Il aime les fraises et il aime la chantilly ».
2. « Il aime les fraises et il n'aime pas la chantilly ».
3. « Il aime les fraises ou il aime la chantilly ».
4. « Il aime les fraises ou il n'aime pas la chantilly ».
5. « Il aime les fraises donc il aime la chantilly ».
6. « Il aime les fraises donc il n'aime pas la chantilly ».
7. « Il n'aime pas les fraises donc il aime la chantilly ».

8. « Il n'aime pas les fraises donc il n'aime pas la chantilly ».

Il est probable que beaucoup ne seront pas d'accord sur toutes les réponses.

Convention d'interprétation des connecteurs

$P1$	$P2$	$P1$ *équivalent à* $P2$	$P1$ *donc* $P2$	$P1$ *et* $P2$	$P1$ *ou* $P2$
vrai	vrai	vrai	vrai	vrai	vrai
vrai	faux	faux	faux	faux	vrai
faux	vrai	faux	vrai	faux	vrai
faux	faux	vrai	vrai	faux	faux

TAB. 1.1 – *Interprétation des quatres connecteurs, P1 et P2 sont deux propositions.*

Proposition	*non Proposition*
vrai	faux
faux	vrai

TAB. 1.2 – *Convention d'interprétation de la négation.*

Les conventions d'interprétation des connecteurs dyadiques [1], introduits dans la première section de l'introduction, sont définies par le tableau 1.1. On peut lire, par exemple, que ($P1$ *équivalent à* $P2$) est vrai lorsque $P1$ et $P2$ prennent les mêmes valeurs. Naturellement, nous interpréterons la négation comme indiqué dans le tableau 1.2

Une interprétation qui a posé de nombreux problèmes est l'interprétation de l'implication (*donc*), et plus particulièrement des lignes 3 et 4, qui nous dit que si $P1$ est faux alors ($P1$ *donc* $P2$) est vrai quel que soit $P2$.

L'acception courante du terme voudrait probablement « si $P1$ alors $P2$ » soit vrai lorsque $P1$ et $P2$ sont vrais tous les deux et soit faux lorsque $P1$ est faux. La valeur vrai sur les lignes 3 et 4 de ($P1$ *donc* $P2$) deviendrait faux. Mais alors nous remarquons que *donc* est devenu synonyme de *et*. Ce n'est donc pas de cette manière-là qu'il nous faut résoudre ce « paradoxe » dit *paradoxe de l'implication matérielle*.

Il faut cependant remarquer que dans bien des situations, l'acception de « si ... alors ... » dans le langage courant recouvre bien la définition que nous donnons ici. Par exemple, soit la proposition « s'il fait beau, j'irai me promener ». S'il fait effectivement beau, j'irai me promener et ma proposition sera vrai. Par contre, s'il pleut, que j'aille ou que je n'aille pas me promener ne rend pas la proposition fausse. L'implication matérielle est donc bien reliée à une notion intuitive réelle.

Convention : convergence des interprétations

Nous avons dit que le but de la logique était de calculer des conclusions sûres. Une convention d'interprétation de nos quatres connecteurs et de notre négation vient d'être définie. Il nous reste maintenant à vérifier que ces conventions nous permettent de faire converger les réponses de l'exercice précédent.

1. Dyadique signifie s'appliquant à deux sous-formules.

Plaçons-nous dans le contexte suivant : « Il aime les fraises et la chantilly ».
Dites si les propositions suivantes sont vraies ou fausses.

1. « Il aime les fraises et il aime la chantilly ».
2. « Il aime les fraises et il n'aime pas la chantilly ».
3. « Il aime les fraises ou il aime la chantilly ».
4. « Il aime les fraises ou il n'aime pas la chantilly ».
5. « Il aime les fraises donc il aime la chantilly ».
6. « Il aime les fraises donc il n'aime pas la chantilly ».
7. « Il n'aime pas les fraises donc il aime la chantilly ».
8. « Il n'aime pas les fraises donc il n'aime pas la chantilly ».

Maintenant, tout le monde devrait être d'accord sur chacune des réponses.

1.2 Langage de la logique des propositions : (ie. syntaxe)

1.2.1 Définitions

Définition 1.1 **-Alphabet-** *L'Alphabet de la logique des propositions (langage que l'on notera $\mathcal{L}$) est constitué des symboles suivants :*

- *les formules* $\{ A, B, C, \ldots, \varphi, \psi, \ldots, p, q, r, s, t, \ldots \}$,
- *les connecteurs* $\{ \leftrightarrow, \rightarrow, \wedge, \vee, \neg \}$,
- *les délimiteurs* $\{ (,) \}$

Définition 1.2 **-Atomes-** *Nous appellerons atomes ou* ***variables propositionnelles*** *ou* ***propositions élémentaires*** *des énoncés dont nous ne connaissons pas (et ne chercherons pas à connaître) la structure interne, et qui gardent leur identité tout au long du calcul propositionnel qui nous occupe.*
L'ensemble des variables propositionnelles est noté $v(\mathcal{L})$.

On désigne généralement des propositions élémentaires par des lettres minuscules de l'alphabet (habituellement p, q, r, ...).

Définition 1.3 **-Connecteurs-** *Nous appellerons connecteurs propositionnels les symboles suivants :*

équivalence $\leftrightarrow$ *;*
implication $\rightarrow$ *;*
conjonction $\wedge$ *;*
disjonction $\vee$ *;*
négation $\neg$.

Définition 1.4 **-Formules-** *Nous dénoterons les formules (ou formules bien formées : fbf) par des lettres majuscules de l'alphabet latin ou grecque (A, B, ... ou φ, ψ). L'ensemble des formules, noté $F(\mathcal{L})$, est défini par :*

- *les atomes sont des formules ($v(\mathcal{L}) \subseteq F(\mathcal{L})$) ;*

– *si φ et ψ sont des formules, alors $(\varphi \leftrightarrow \psi)$, $(\varphi \rightarrow \psi)$, $(\varphi \wedge \psi)$, $(\varphi \vee \psi)$ et $(\neg\varphi)$ sont des formules.*

Toute formule est obtenue par l'application des règles précédentes un nombre fini de fois.

L'ensemble des formules $F(\mathcal{L})$ est infini, même si l'ensemble des propositions élémentaires $v(\mathcal{L})$ est fini.

En appliquant strictement la définition ci-dessus de « Formules », nous dirions que $p \leftrightarrow q \rightarrow r$ n'est pas une formule bien formée. En effet, le parenthèsage est absent. On ne sait pas s'il s'agit de la formule $((p \leftrightarrow q) \rightarrow r)$ ou s'il s'agit de $(p \leftrightarrow (q \rightarrow r))$. Les parenthèses sont un moyen de lever l'ambiguïté. Il en existe un autre qui consiste à donner à chaque opérateur un ordre de priorité. **Les connecteurs sont traditionnellement classés de la façon suivante (par priorité décroissante des connecteurs) : $\neg$, $\wedge$, $\vee$, $\rightarrow$, $\leftrightarrow$. Dans le cas où deux connecteurs ont même priorité, et en l'absence de parenthèses, l'associativité se fait de gauche à droite**. On peut ainsi se permettre d'omettre les parenthèses. Par exemple, la formule $p \rightarrow q \leftrightarrow \neg r$ doit se lire $((p \rightarrow q) \leftrightarrow (\neg r))$. A l'avenir, lorsque nous parlerons de formules bien formées, nous inclurons également les formules dont le parenthèsage est partiellement ou complètement implicite. L'ensemble des formules bien formées ainsi défini forme le langage de la logique propositionnelle.

Définition 1.5 -Littéral- *Un littéral est un atome (littéral positif) ou la négation d'un atome (littéral négatif).*

Définition 1.6 -Clause- *Une clause est une disjonction de littéraux ($p_1 \vee p_2 \vee \ldots \vee p_n$), les littéraux pouvant être positifs ou négatifs.*

1.2.2 Travaux dirigés (la syntaxe)

Formules bien formées

Les « formules » suivantes sont-elle des formules bien formées?

1. $a \vee b \wedge c$
2. $\neg a \vee b \wedge c$
3. $a \vee \neg b \wedge c$
4. $a\neg \vee b \wedge c$
5. (a)
6. $(a)b$
7. $\neg b(a)$
8. $a \vee \neg(b \wedge c)$
9. $a\neg(\vee b \wedge c)$
10. $a \rightarrow b$
11. $a \leftarrow a$
12. $a \rightarrow b \leftrightarrow c$
13. $a \rightarrow \neg(b \leftrightarrow c)$
14. $a\neg \rightarrow b$
15. $a \vee (b \leftrightarrow c)\neg c \rightarrow d$
16. $a \vee (b \leftrightarrow c) \rightarrow \neg c \rightarrow d$

Parenthèsage

Explicitez le parenthèsage implicite des formules suivantes :

1. $a \to b \to c$
2. $a \vee b \wedge c$
3. $a \vee b \wedge c \leftrightarrow d \to \neg\ e \vee f \wedge g$

Simplifiez au maximum le parenthèsage des formules suivantes :

1. (a)
2. $((a \vee b))$
3. $((a) \wedge (b))$
4. $(\neg(a \vee b))$
5. $\neg(((a) \wedge b))$
6. $a \vee (b \wedge c)$
7. $(a \vee b) \wedge c$
8. $((a \wedge b) \to c)$
9. $(a \wedge (b \to c))$
10. $((a \vee b) \wedge c) \leftrightarrow e$
11. $(((a \vee b) \wedge c) \leftrightarrow e) \to f$
12. $((a \wedge b) \vee c) \leftrightarrow (e \to f)$
13. $(((a \to b) \to c) \to d)$
14. $(a \wedge (b \wedge c))$
15. $(a \to (b \to c))$

Formalisation d'un énoncé (premier pas vers la sémantique)

Traduisez les énoncés suivants en formule logique :

1. Quand il fait beau, Jean est heureux ;
   il fait soleil ;
   donc, Jean est heureux.
2. Quand il fait beau, Jean est heureux ;
   or, il fait mauvais ;
   donc Jean est malheureux.
3. Quand il fait beau, Jean est heureux ;
   or, Jean est malheureux ;
   donc il fait mauvais.

Remarque : la qualité et la correction de la traduction ne sont pas du ressort de la logique.

1.3 Théorie des modèles (ie. sémantique)

1.3.1 Introduction

La théorie des modèles a pour but d'établir un mécanisme sémantique d'évaluation des formules. Par sémantique, nous entendons que nous allons donner un sens à nos atomes, puis donner un sens à une formule à partir de la valeur de vérité des atomes qui la composent et des connecteurs qui relient ces atomes. La théorie des modèles est une formalisation de la notion intuitive que nous avons de la vérité ou de la fausseté d'une phrase en fonction des propositions qui la composent.

En calcul propositionnel, ce résultat est obtenu en donnant à chaque atome une *valeur de vérité* et en associant à chaque connecteur une *table de vérité* qui, en fonction de la valeur de vérité des arguments du connecteur, donne la valeur de vérité de la formule constituée (nous avons déjà introduit de telles tables, cf. section 1.1).

Notre hypothèse [2] , en calcul propositionnel, est donc que chaque atome ne peut prendre que deux valeurs : *vrai* ($\mathbb{T}$) ou *faux* ($\mathbb{F}$), et que la valeur de vérité d'une composition de formules est entièrement déterminée par la valeur de chacun de ses arguments.

1.3.2 Définitions

Définition 1.7 -Valuation- *On appelle valuation, ou $\mathcal{L}$-Modèle, d'un ensemble de variables propositionnelles $v \subseteq v(\mathcal{L})$, une fonction m de $v(\mathcal{L})$ dans $\{\mathbb{T},\mathbb{F}\}$ (ie. $m : v(\mathcal{L}) \rightarrow \{\mathbb{T},\mathbb{F}\}$).*

L'ensemble des valuations ($\mathcal{L}$-modèles) est noté $M(\mathcal{L})$, ou plus simplement $M_{\mathcal{L}}$.

Définition 1.8 -Sémantique des connecteurs- *Soient A et B deux formules, la sémantique des connecteurs est donnée par les tables de vérité représentées dans les tableaux 1.3 et 1.4.*

A	B	$A \leftrightarrow B$	$A \rightarrow B$	$A \wedge B$	$A \vee B$
$\mathbb{T}$	$\mathbb{T}$	$\mathbb{T}$	$\mathbb{T}$	$\mathbb{T}$	$\mathbb{T}$
$\mathbb{T}$	$\mathbb{F}$	$\mathbb{F}$	$\mathbb{F}$	$\mathbb{F}$	$\mathbb{T}$
$\mathbb{F}$	$\mathbb{T}$	$\mathbb{F}$	$\mathbb{T}$	$\mathbb{F}$	$\mathbb{T}$
$\mathbb{F}$	$\mathbb{F}$	$\mathbb{T}$	$\mathbb{T}$	$\mathbb{F}$	$\mathbb{F}$

TAB. 1.3 – *Table de vérité des connecteurs dyadiques.*

A	$\neg A$
$\mathbb{T}$	$\mathbb{F}$
$\mathbb{F}$	$\mathbb{T}$

TAB. 1.4 – *Table de vérité du connecteur monadique.*

Définition 1.9 -Interprétation d'une formule- *Une interprétation d'une formule dans laquelle apparaissent les variables propositionnelles $v_1,\ldots,v_n$ est une valuation de $\{v_1,\ldots,v_n\}$.*

2. Ce n'est pas le cas pour toutes les logiques, et notamment pour la logique modale.

Remarque : une formule composée de n atomes propositionnels admet 2^n interprétations.

Exemple : soit la formule $F = (a \wedge b) \vee \neg b \rightarrow \neg a$, $\{m(a) = \mathbb{T}, m(b) = \mathbb{F}\}$ est une interprétation de F.

Définition 1.10 -Modèle d'une formule- *Une interprétation I est un modèle d'une formule φ si elle est vrai (si elle vaut $\mathbb{T}$).*

1.3.3 Validité et consistance

Définition 1.11 -Formule valide- *Une formule valide, ou* ***tautologie****, est une formule φ vraie quelles que soient les valeurs de vérité des atomes qui la composent (*i.e. *vraie dans toute interprétation). On la note $\models \varphi$.*

Définition 1.12 -Formule insatisfiable- *Une formule insatisfiable, ou* ***sémantiquement inconsistante****, ou encore* ***antitautologie****, est une formule fausse dans toute interprétation.*

Définition 1.13 -Formules satisfiables- *Une formule satisfiable ou* ***sémantiquement consistante*** *est une formule vraie dans au moins une interprétation.*

Définition 1.14 -Formules invalides- *Une formule invalide est fausse dans au moins une interprétation.*

Définition 1.15 -Formules contingentes- *Une formule contingente est vraie dans certaines interprétations et fausse dans d'autres.*

Définition 1.16 -Conséquence valide- *Soient deux formules A et B. Nous dirons que A est la conséquence valide de B ($A \models B$) si tout modèle de A est un modèle de B.*

Remarques

Soit une formule bien formée A, on a :
- A valide entraîne A consistante ;
- A inconsistante entraîne A invalide ;
- A est valide si et seulement si $\neg A$ est inconsistante.

Trois cas se présentent pour A :
- soit A est valide (donc consistante) ;
- soit A est invalide mais consistante (*i.e.* contingente) ;
- soit A est inconsistante (donc invalide).

1.3.4 Équivalence des formules bien formées

Définition 1.17 -Formules équivalentes- *Deux formules sont équivalentes quand elles ont même valeur dans toutes interprétation (notation : $A \equiv B$).*

Théorème 1.1 -Règle de substitution uniforme- *Soit la formule φ contenant les atomes p_1,p_2,...,p_n. Soit la formule φ^* obtenue en substituant aux atomes p_1,p_2,...,p_n les formules ψ_1,ψ_2,...,ψ_n. Alors si $\models \varphi$, on a $\models \varphi^*$.*

Propriété 1.1 -Formules équivalentes- *Soient A, B et C trois formules bien formées.*

Implication
- $A \rightarrow B \equiv \neg A \vee B$
- $\neg(A \rightarrow B) \equiv A \wedge \neg B$
- $A \leftrightarrow B \equiv (A \rightarrow B) \wedge (B \rightarrow A)$

Idempotence
- $A \wedge A \equiv A$
- $A \vee A \equiv A$

Commutativité
- $A \vee B \equiv B \vee A$
- $A \wedge B \equiv B \wedge A$

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

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

Élément neutre
- $A \vee \mathbb{F} \equiv A$
- $A \wedge \mathbb{T} \equiv A$
- $A \vee \mathbb{T} \equiv \mathbb{T}$
- $A \wedge \mathbb{F} \equiv \mathbb{F}$

Complémentarité
- $A \vee \neg A \equiv \mathbb{T}$
- $A \wedge \neg A \equiv \mathbb{F}$

Involution
- $\neg\neg A \equiv A$

De Morgan
- $\neg(A \vee B) \equiv \neg A \wedge \neg B$
- $\neg(A \wedge B) \equiv \neg A \vee \neg B$

Absorption
- $A \vee (\neg A \wedge B) \equiv A \vee B$
- $A \wedge (\neg A \vee B) \equiv A \wedge B$
- $A \vee (A \wedge B) \equiv A$
- $A \wedge (A \vee B) \equiv A$

1.3.5 Travaux dirigés (validité, consistance, formules équivalentes)

Équivalence de formules

A l'aide d'une table de vérité, vérifiez les deux équivalences de Morgan.

Valeur de formules

1. A l'aide d'une table de vérité, étudiez la validité et la consistance des formules que vous avez trouvées pour l'exercice « Formalisation d'un énoncé » du TD 1.2.2.
2. A l'aide d'une table de vérité, étudiez la validité et la consistance de la formule suivante :
$$F = (P \wedge Q \rightarrow \neg R) \wedge \neg P \wedge R \rightarrow \neg Q$$

Traduction

Traduire en formule logique les énoncés suivants, puis étudier leur validité et leur consistance :

1. Quand on fait de l'alpinisme, on est montagnard ou sportif ;
un montagnard est un sportif ;
donc, quand on n'est pas sportif, on ne fait pas d'alpinisme.
2. Il ne dit rien s'il travaille ou s'il est seul ;
il se repose mais il se tait ;
donc il est seul.
3. Jean ne sort que s'il fait beau ;
or, il pleut ;
donc, Jean reste chez lui.

Démonstration

Montrer que les deux premières équivalences de l'absorption ($A \vee (\neg A \wedge B) \equiv A \vee B$ et $A \wedge (\neg A \vee B) \equiv A \wedge B$) sont justes en utilisant, entre autres, la distributivité.

Simplification

En utilisant des équivalences de formules, tenter de trouver une forme plus simple pour les formules suivantes :

1. $(a \rightarrow a) \vee (a \rightarrow b)$
2. $a \wedge (\neg a \vee c)$
3. $\neg a \wedge (a \rightarrow b)$
4. $a \vee (\neg c \vee (b \rightarrow c))$
5. $a \wedge \neg b \vee b \wedge \neg a \rightarrow a$

1.3.6 Travaux pratiques : tables de vérité sous Excel

Introduction

Nous allons travailler avec le logiciel Microsoft Excel que vous devez déjà avoir manipulé (ce TP ne constitue pas une initiation aux manipulations de base sous Microsoft Excel).

Dans ce logiciel, $\mathbb{T}$ se notera $VRAI$ et $\mathbb{F}$ se notera $FAUX$. $\neg A$ s'écrira $NON(A)$, $A \vee B$ s'écrira $OU(A;B)$ et $A \wedge B$ s'écrira $ET(A;B)$. Toutes les formules doivent commencer par le signe « = ».

Bien entendu, les variables A et B doivent être remplacées par des coordonnées de cellule, par exemple $ET(B3;C3)$ si les cellules $B3$ et $C3$ contiennent la valeur logique des variables A et B dont on cherche à évaluer la conjonction.

Tous les autres opérateurs n'existent pas et doivent être obtenus à l'aide des formules équivalentes constituées des trois opérateurs existants.

Travail pratique 1

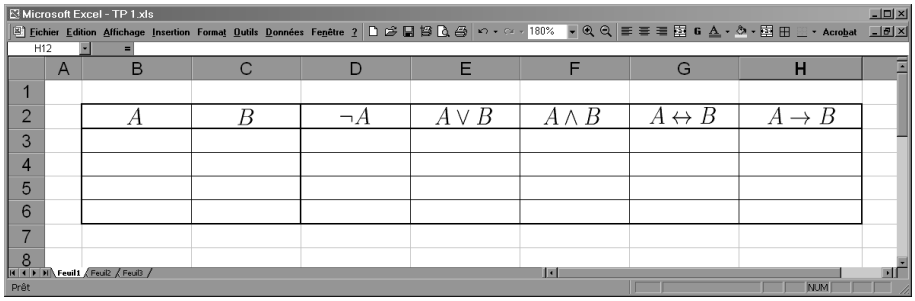

FIG. 1.1 – *Capture d'écran du TP 1*

On se propose d'établir la table de vérité des 5 connecteurs logiques que nous avons vus à l'aide du logiciel Microsoft Excel. Voir la figure 1.1 pour une capture d'écran de la feuille de calcul correspondant au TP1.

1. Remplir les colonnes B et C du tableau pour obtenir toutes les valuations possibles de l'ensemble de variables propositionnelles $\{A,B\}$.
2. Saisir, dans les cellules de la ligne 3 du tableau, les formules correspondant à nos 5 connecteurs. Par exemple, dans la cellule $D3$ on saisira $= NON(B3)$.
3. Faire un copier coller vers les trois lignes au dessous.

Travail pratique 2

On se propose d'établir la table de vérité de la formule

$$F = (P \wedge Q \rightarrow \neg R) \wedge \neg P \wedge R \rightarrow \neg Q$$

à l'aide du logiciel Microsoft Excel. Voir la figure 1.2 pour une capture d'écran de la feuille de calcul correspondant au TP2.

1. Remplir les colonnes B et C et D du tableau pour obtenir toutes les valuations possibles de l'ensemble de variables propositionnelles $\{P,Q,R\}$. On ne rentrera manuellement que les valeurs de la première ligne. On cherchera ensuite un moyen pour systématiser et automatiser le remplissage des autres lignes.
2. Écrire les formules et faire le copier/coller pour remplir le reste du tableau.
3. Trouver les formules logiques qui permettent de remplir automatiquement les cellules $G12$ à $G16$.

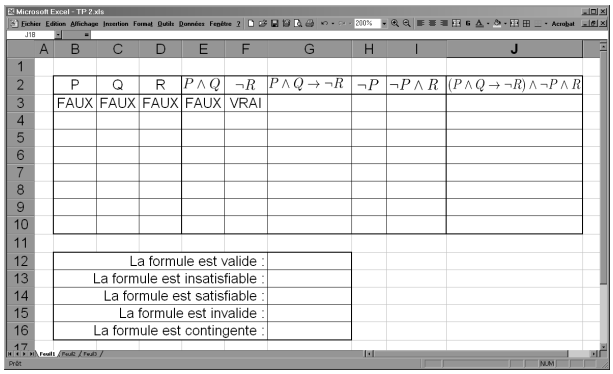

FIG. 1.2 – *Capture d'écran du TP 2*

Travail pratique 3

On se propose d'étudier la validité et la consistance de différentes formules. Voir la figure 1.3 pour une capture d'écran de la feuille de calcul correspondant au TP3.

Le déroulement de ce TP est analogue à celui du TP2. Les colonnes intermédiaires pour faciliter l'évaluation des formules sont absentes, il faudra donc saisir des formules plus complexes dans le tableau.

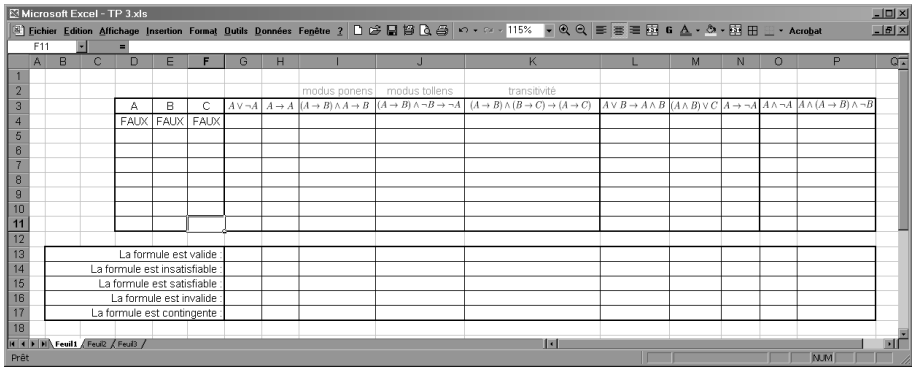

FIG. 1.3 – *Capture d'écran du TP 3*

1.4 Calcul propositionnel et résolution

Le but de ce chapitre est d'exposer des méthodes algorithmiques permettant de déterminer la satisfiabilité d'une formule, d'un ensemble de clauses et de faire de la démonstration automatique.

1.4.1 Principe de déduction

Introduction

Nous avons déjà vu une méthode générale pour établir la validité et la consistance d'une formule : la méthode des tables de vérité. Cependant, cette méthode réclame, pour une formule contenant n atomes distincts, le calcul d'une table comprenant 2^n lignes. Il faut donc essayer de trouver des méthodes plus efficaces.

Le principe de déduction ramène toute preuve de déduction à une preuve d'inconsistance. Cela nous permettra de nous concentrer uniquement sur les mécanismes de preuve d'inconsistance.

Il faut cependant signaler un résultat théorique important : le problème de satisfiabilité d'une formule propositionnelle quelconque est NP-complet. Cela signifie qu'il n'existe presque certainement aucun algorithme efficace permettant de calculer une affectation qui satisfait une formule, ou de vérifier sa validité.

Définitions

On a pu voir (définition 2.16 page 30) qu'une formule est insatisfiable si et seulement si elle n'a pas de modèle. Cette notion s'étend simplement à un ensemble de formules :

Définition 1.18 -Ensemble de formules insatisfiables- *Un ensemble de formules est insatisfiable, ou (sémantiquement) inconsistant, si et seulement si il n'existe aucune interprétation M telle que chaque formule A de E soit satisfaite par M.*

Il découle de la définition précédente et de la notion de conséquence valide (définition 1.16 page 10) que :

Théorème 1.2 -Principe de déduction- *Une formule A est la conséquence valide d'un ensemble de formules E ssi $E \cup \{\neg A\}$ est insatisfiable.*

En effet, toute interprétation est :

- soit c'est un modèle de E, et c'est alors un modèle de A, et donc certainement pas un modèle de $E \cup \{\neg A\}$;
- soit ce n'est pas un modèle de E, et ce n'est donc pas, a fortiori, un modèle de $E \cup \{A\}$.

Réciproquement, si $E \cup \{\neg A\}$ est insatisfiable, cela implique que toute interprétation qui satisfait E ne satisfait pas $\neg A$, et satisfait donc A.

Les arbres sémantiques

Cette méthode n'est pas plus efficace que la méthode classique des tables de vérité. Elle consiste à construire pour une formule F contenant les atomes de $S = \{p_1, \ldots, p_n\}$, un arbre binaire d'après les règles suivantes :

- chaque arc est étiqueté par un littéral p ou $\neg p$ avec $p \in S$. La branche p correspond à une affectation de $\mathbb{T}$ à p alors que la branche $\neg p$ correspond à l'affectation de $\mathbb{F}$ à p ;
- les littéraux étiquetant les deux arcs issus d'un même nœud sont opposés ;
- aucune branche ne comporte plus d'une occurrence de chaque atome.

Un arbre sémantique est dit *complet* si chaque branche contient une fois chaque atome. Il est dit *partiel* dans le cas contraire. De la même manière que la table de vérité de F contient 2^n lignes, un arbre sémantique complet comprend 2^n feuilles.

Algorithme de Quine

L'Algorithme de Quine est une amélioration des arbres sémantiques. A chaque nœud de l'arbre binaire, on réalise une évaluation partielle en prenant en compte tous les atomes dont la valeur est déterminée. Si cette évaluation permet de conclure directement, on ne poursuit pas la construction à partir de ce nœud.

Algorithme de réduction

Basée sur un mécanisme de preuve par l'absurde, cette méthode peut être avantageusement utilisée quand la formule comprend de nombreuses implications. Elle consiste à supposer que la formule initiale est fausse, puis à en déduire les sous-valeurs logiques de chacune des deux sous-formules placées de chaque côté de l'implication et à réitérer le processus jusqu'au bout.

Voici un exemple. Soit la formule : $((p \wedge q) \rightarrow r) \rightarrow (p \rightarrow (q \rightarrow r))$. On va supposer qu'il existe une assignation de p, q, r qui rend cette formule fausse. On en déduit alors :

1. $(p \wedge q) \rightarrow r$ est assignée à $\mathbb{T}$;
2. $p \rightarrow (q \rightarrow r)$ est assignée à $\mathbb{F}$

A partir de ces éléments, on peut appliquer à nouveau le processus sur (2) ; on obtient alors que p est $\mathbb{T}$, et enfin que q est $\mathbb{T}$ et r est $\mathbb{F}$. Or cette affectation est en contradiction avec (1). Donc la formule est valide.

1.4.2 Travaux dirigés (algorithmes de Quine et de réduction)

Algorithme de Quine

En utilisant l'algorithme de Quine, étudiez la validité et la consistance de la formule suivante :

$$F = (P \wedge Q \rightarrow \neg R) \wedge \neg P \wedge R \rightarrow \neg Q$$

Algorithme de réduction

En utilisant l'algorithme de réduction, étudiez la validité de la formule suivante :

$$F = (A \to B) \wedge (B \to C) \to (A \to C)$$

1.4.3 Formes normales

Les méthodes précédentes (table de vérité, arbre sémantique, algorithme de Quine, algorithme de réduction) sont inefficaces dans la pratique. En exploitant une forme simplifiée d'expression de formules, certaines méthodes peuvent être plus efficaces en moyenne.

Définition 1.19 -Forme conjonctive normale- *Une forme normal conjonctive est une conjonction de clauses.*

La forme conjonctive normale d'une formule n'est pas unique.

On considère souvent une forme conjonctive normale $C_1 \wedge \ldots \wedge C_n$ comme un ensemble de clauses et on la note alors : $\{C_1, \ldots, C_n\}$. On utilise ainsi souvent une terminologie ensembliste lorsque l'on parle de forme conjonctive normale (on dit, par exemple, qu'une clause *appartient* à une forme conjonctive normale).

Théorème 1.3 -Théorème de normalisation- *Toute formule admet une forme conjonctive normale qui lui est logiquement équivalente.*

Algorithme de normalisation - A partir d'une formule quelconque, voici comment construire la forme conjonctive normale associée.

1. On remplace toutes les occurrences de $(A \leftrightarrow B)$ par $(A \to B) \wedge (B \to A)$. Ceci supprime toutes les occurrences du connecteur $\leftrightarrow$.
2. On remplace toutes les occurrences de $(A \to B)$ par $(\neg A \vee B)$. Ceci supprime toutes les occurrences du connecteur $\to$.
3. On applique récursivement les règles de réécriture suivantes :
   $\neg(A \wedge B) \rightsquigarrow (\neg A \vee \neg B)$;
   $\neg(A \vee B) \rightsquigarrow (\neg A \wedge \neg B)$.
   On ne trouve plus maintenant d'occurrence de $\neg$ que devant des atomes, ou devant d'autres occurrences de $\neg$.
4. on supprime toutes les doubles occurrences de la négation : $\neg\neg A \rightsquigarrow A$. Il ne reste alors plus que des occurrences uniques de $\neg$ placées directement devant les atomes.
5. On applique la règle de distributivité : $A \vee (B \wedge C) \rightsquigarrow (A \vee B) \wedge (A \vee C)$. (On peut également avoir à appliquer la commutativité de $\vee$ si l'expression est « dans le mauvais sens »).

La formule obtenue est une forme normale équivalente à la formule de départ.

Une clause d'une forme normale contenant deux littéraux p et $\neg p$ est valide et peut être supprimée de la forme normale sans que son sens soit modifié.

Définition 1.20 -Forme conjonctive normale pure- *Une forme normale ne contenant aucune clause valide est dite forme conjonctive normale pure.*

Remarques :

- si une clause Ci, élément d'une forme normale, contient une clause C_j, élément de la même forme normale, la clause C_i peut être supprimée de la forme normale sans que son sens ne soit changé ;
- une forme normale vide est valide ;
- une forme normale contenant la clause vide est inconsistante ;
- les formules valides ont une forme normale pure vide ; le test de validité d'une forme normale est trivial ;
- on réduit classiquement deux clauses opposées $(A \wedge \neg A)$ en $\varnothing$, et n'importe quelle conjonction de clauses avec $\varnothing$, par exemple $A \wedge \varnothing$, à $\varnothing$. La clause $\varnothing$ est la seule clause inconsistante.

1.4.4 Travaux dirigés (normalisation)

Formes conjonctives normales

Les formules suivantes sont-elles en forme conjonctive normale ? Combien contiennent-elles de clauses ?

1. ;
2. $\varnothing$;
3. $a \vee b$;
4. $\neg(a \vee b) \vee c$;
5. $a \wedge b \wedge \neg c$;
6. $a \vee b \wedge c \vee d$;
7. $(a \vee b) \wedge (\neg a \vee c) \vee d$;
8. $(a \vee b) \wedge (\neg a \vee c \vee d)$;
9. $(a \vee b) \wedge (\neg a \vee c \wedge d)$;
10. $(a \vee b) \wedge (\neg a \vee c) \wedge d$;
11. $(\neg a \vee b) \wedge \neg(\neg a \vee c) \wedge (d \vee e)$;
12. $(\neg a \vee b) \wedge a \wedge \neg c \wedge (d \vee e)$.

Normalisation

Trouver une forme normale conjonctive des formules suivantes :

1. $p \wedge q \rightarrow p \vee q$;
2. $p \vee q \rightarrow p \wedge q$;
3. $p \leftrightarrow (p \rightarrow r)$;
4. $p \leftrightarrow (q \rightarrow r)$.

1.4.5 Principe de résolution

Le principe de résolution est formé d'une unique règle d'inférence. Sa grande simplicité de mise en œuvre en fait une méthode très utilisée.

Nous avons vu qu'un ensemble de clauses est inconsistant si et seulement si $\varnothing$ est une conséquence valide de cet ensemble. L'algorithme de résolution va construire des conséquences valides de notre ensemble N de départ jusqu'à obtention de la clause vide.

Théorème 1.4 -Principe de résolution- *Soit N une forme normale, C_1 et C_2 deux clauses de N. Soit p un atome tel que $p \in C_1$ et $\neg p \in C_2$. Soit la clause $R = C_1 \setminus \{p\} \cup C_2 \setminus \{\neg p\}$. Alors les formes normales N et $N \cup R$ sont logiquement équivalentes. La clause R est appelée résolvante de C_1 et $C2$.*

Exemple :

- $C_1 = \neg M \vee S$, $C_2 = M \vee S$, la résolvante de C_1 et C_2 est $R = S$;
- $C_1 = \neg M$, $C_2 = P \vee M \vee S$, la résolvante de C_1 et C_2 est $R = S \vee P$;
- $C_1 = A \vee B$, $C_2 = \neg B \vee C \vee D$, la résolvante de C_1 et C_2 est $R = A \vee C \vee D$;
- $C_1 = A \vee B$, $C_2 = A \vee C$, il n'y a pas de résolvante ;
- $C_1 = \neg B \vee A$, $C_2 = B \vee C \vee \neg A$ il y a deux résolvantes pour C_1 et C_2, $R_1 = A \vee C \vee \neg A$ et $R_2 = \neg B \vee B \vee C$; ces deux résolvantes sont des tautologies et n'apportent rien.

Le principe de résolution peut s'avérer très efficace dès lors que l'on sait choisir les bonnes clauses dans le bon ordre, mais il ne fournit pas, en général, de méthode plus efficace que les algorithmes vus précédemment.

Application

Nous voulons démontrer que : $\{p \to r, q \to r\} \vDash (p \vee q) \to r$.

- D'après le théorème de déduction, l'énoncé précédent se ramène à : $\{p \to r, q \to r, \neg((p \vee q) \to r)\} \vDash \varnothing$.
- Il nous reste donc à démontrer que $\{p \to r, q \to r, \neg((p \vee q) \to r)\}$ est inconsistant.
- En ramenant notre ensemble de clauses en forme normale, nous obtenons : $\{\neg p \vee r, \neg q \vee r, p \vee q, \neg r\}$.
- Nous allons appliquer le principe de résolution pour démontrer cette inconsistance. Nous allons numéroter nos clauses

$$\{\underbrace{\neg p \vee r}_{(1)}, \underbrace{\neg q \vee r}_{(2)}, \underbrace{p \vee q}_{(3)}, \underbrace{\neg r}_{(4)}\}.$$

  puis dérouler les conséquences valides.
- (5) $\neg p$ à partir de 1 et 4.
- (6) $\neg q$ à partir de 2 et 4.
- (7) q à partir de 5 et 3.
- $\varnothing$ à partir de 6 et 7.

Le résultat a été obtenu de façon rapide principalement parce que l'on a su choisir les bonnes clauses dans le bon ordre.

Remarque

Comme nous l'avons déjà dit, le test de validité d'une forme normale est trivial puisque les formules valides ont une forme normale pure vide. Mais alors à quoi sert le principe de résolution ? En fait, le principe de résolution est surtout utile en logique des prédicats. C'est la base du language PROLOG. Le principe de résolution n'est pas vraiment utile en logique des propositions.

1.4.6 Travaux dirigés (principe de résolution I)

Principe de résolution

Utiliser le principe de résolution pour étudier la validité des énoncés suivants :

1. $(A \to B) \wedge A \to B$;
2. $(A \to B) \wedge \neg B \to \neg A$
3. $(A \to B) \wedge (B \to C) \to (A \to C)$
4. $(A \to M \vee S) \wedge (M \to S) \to (\neg S \to \neg A)$
5. $(T \vee S \to P) \wedge (\neg T \wedge P) \to S$

Comparatif

Soit la formule suivante :

$$(p \to q) \wedge ((q \wedge \neg r) \to s) \wedge (\neg t \to (\neg r \wedge \neg s)) \to (p \to t)$$

Étudier la validité de la formule par les moyens suivants :

1. table de vérité partielle ;
2. algorithme de Quine (arbre sémantique partiel) ;
3. le principe de résolution.

Exercice complet

Traduire l'énoncé suivant dans le formalisme de la logique des propositions, puis étudier sa validité au moyen de l'algorithme de Quine puis du principe de résolution :

- quand un bateau fait naufrage, cela fait parler les journalistes ;
- les sponsors sont inquiets quand les journalistes parlent et qu'un bateau fait naufrage ;
- un bateau fait naufrage ;
- donc les sponsors sont inquiets.

1.4.7 Principe de résolution adapté aux clauses de Horn

La méthode de résolution est, dans le cas général, inefficace. Il existe cependant des cas particuliers où cette méthode est rapide.

Nous allons nous intéresser dans ce chapitre à un type particulier de forme normale, les formes normales composées de clause de Horn.

Définition 1.21 -Clause de Horn- *Une clause de Horn est une clause comportant au plus un littéral positif. Il existe donc trois types de clauses de Horn :*

- *celles comportant un littéral positif et au moins un littéral négatif, appelées* ***clauses de Horn strictes ;***
- *celles comportant un littéral positif et aucun littéral négatif, appelées* ***clauses de Horn positives;***
- *celles ne comportant que des littéraux négatifs, appelées* ***clauses de Horn négatives****(et dont fait partie la clause vide* $\varnothing$*)*

Algorithme de résolution adapté aux clauses de Horn

L'algorithme présenté ci-dessous constitue une amélioration de l'algorithme de résolution appliqué à une forme normale N ne contenant que des clauses de Horn.

1. si $\varnothing$ est dans N, l'ensemble est inconsistant et la résolution est terminée ;
2. sinon, choisir une clause C et une clause P telles que P soit une clause de Horn positive réduite à p et C une clause contenant $\neg p$;
3. calculer la résolvante R de P et C ;
4. reprendre l'algorithme en remplaçant N par $(N \setminus \{C\}) \cup R$.

Cet algorithme peut se terminer de deux façons :

1. nous avons abouti à $\varnothing$ et l'ensemble N est alors inconsistant ;
2. il est impossible de poursuivre car nous ne parvenons plus à trouver P et C vérifiant les conditions requises ; la formule est alors consistante et le modèle composé de chacune des clauses positives restantes (ou plus exactement le modèle composé de chacun des atomes de ces clauses), lorsque l'algorithme se termine, est un modèle de N (on appelle d'ailleurs ce modèle le modèle canonique de N).

Remarques

- Nous nous permettons ici de remplacer N par $(N \setminus \{C\}) \cup R$ alors que l'algorithme général demande d'utiliser $N \cup R$. Mais dans le cas présent, la clause C est une conséquence valide de R, et peut donc être omise (deuxième formule équivalente de l'absorption). Nous engendrons donc bien des conséquences valides de N.
- On remarque que la résolvante R de P et C n'est autre que $C \setminus \{\neg p\}$. On supprime donc à chaque étape un atome dans une clause de N. On est donc certain que l'algorithme aboutit. En ce qui concerne la complexité, il existe un algorithme linéaire résolvant le problème de la satisfiabilité d'un ensemble de clauses de Horn.

Exemple de résolution

Nous allons utiliser cette méthode pour démontrer l'inconsistance de : $\{\neg p \vee r, \neg r \vee s, p, \neg s\}$

1. Par sélection de p et $\neg p \vee r$, l'ensemble se réduit à : $\{r, \neg r \vee s, p, \neg s\}$.
2. Par sélection de r et $\neg r \vee s$, nous obtenons : $\{r, s, p, \neg s\}$.
3. Enfin, par sélection de s et $\neg s$: $\{r, s, p, \varnothing\}$.

L'ensemble de clauses est bien inconsistant.

Interprétation

Quel est le cadre d'application des clauses de Horn ? Ou encore, que peut-on représenter avec des clauses de Horn ?

- Les clauses de Horn positives, par exemple p, sont appelées faits. Il s'agit en effet de l'énoncé de la vérité logique d'un atome.
- Les clauses de Horn strictes $q \vee \neg p_1 \vee \ldots \vee \neg p_n$ sont équivalentes à $\{p_1, \ldots, p_n\} \models q$ et représentent des règles du type *si ... alors ...*. Elles permettent de déduire de nouveaux faits à partir de faits existants.
- les clauses négatives peuvent se comprendre comme des buts à atteindre. Considérons que nous souhaitions prouver $\{H_1, \ldots, H_n\} \models (p \wedge q \wedge r)$. La partie $(p \wedge q \wedge r)$ est le but de notre résolution. En appliquant une technique d'inconsistance nous sommes ramenés à $\{H_1, \ldots, H_n, (\neg p \vee \neg q \vee \neg r)\} \models \varnothing$. La clause $(\neg p \vee \neg q \vee \neg r)$ est une clause de Horn négative qui modélise donc bien le but à atteindre.

1.4.8 Travaux dirigés (principe de résolution II)

Principe de résolution sur des clauses de Horn

Utiliser le principe de résolution appliqué à des clauses de Horn pour étudier la validité des énoncés suivants :

1. $(A \rightarrow B) \wedge A \rightarrow B$;
2. $(A \rightarrow B) \wedge \neg B \rightarrow \neg A$
3. $(A \rightarrow B) \wedge (B \rightarrow C) \rightarrow (A \rightarrow C)$

Enquête logique

Trois personnes, *Adrien*, *Béatrice* et *Christophe*, accusées d'un meurtre, déclarent respectivement :

- Béatrice est coupable et Christophe est innocent (Adrien) ;
- si Adrien est coupable, alors Christophe l'est aussi (Béatrice) ;
- je suis innocent, mais au moins l'une des deux personnes est coupable (Christophe).

Utiliser le formalisme du calcul des propositions pour traduire les questions suivantes et donner la réponse :

1. Les trois déclarations sont-elles compatibles ?

2. L'un des témoignages peut-il se déduire des autres? Lequel?
3. Si tous sont innocents, lequel a menti?
4. Si tous disent la vérité, qui est coupable?
5. Si seuls les innocents disent la vérité, qui est coupable?

Chapitre 2

Logique des prédicats

2.1 Langage de la logique des prédicats

2.1.1 Introduction

Généralités

Dans le chapitre précédent, nous nous sommes intéressés aux propositions comme éléments de base de notre langage. Nous allons, dans ce chapitre, enrichir notre analyse de ces propositions en introduisant une nouvelle structure, celle de prédicat.

Un prédicat formalise une notion appelée en grammaire structure « sujet-prédicat ». Ainsi, considérons la proposition : « Tout homme est mortel ». La structure « est mortel » est un prédicat et « homme » est le sujet du prédicat.

Le calcul des prédicats permet de s'intéresser à des énoncés relatifs à un ensemble d'objets, comme « Tout homme est mortel » ou à certains éléments de ces ensembles, comme « il existe des hommes mauvais ». Les notions de « quel que soit » et de « il existe » sont fondamentales en calcul des prédicats.

Notons enfin que le calcul des prédicats est aussi appelé *logique du premier ordre*.

Limitation de la logique des propositions

Prenons le problème suivant :

Tout homme est mortel.
Socrate est un homme.
Donc, Socrate est mortel.

Nous avons déjà traduit des énoncé en logique des proposition. Supposons la traduction suivante :

- *Tout homme est mortel* est traduit par la proposition a.
- *Socrate est un homme* est traduit par la proposition b.
- *Donc, Socrate est mortel* est traduit par la proposition c.

Le problème s'écrit alors, en logique des propositions, $a \wedge b \rightarrow c$. Cette traduction est correcte. Mais elle est de piètre qualité. En effet, il n'échappera à personne que le raisonnement de départ était valide mais que sa traduction ne l'est pas (bien qu'elle soit satisfiable).

Légitimité de la logique des prédicats

Définition 2.1 -Prédicat- *Un prédicat est une propriété ou relation qui porte sur un ou plusieurs éléments d'un domaine $\mathcal{D}$. C'est une fonction de $\mathcal{D}$ dans $\{\mathbb{T},\mathbb{F}\}$.*

Le problème précédent peut être traduit de manière adéquate en logique des prédicats :

Hypothèse 1 : Quelque soit x appartenant au domaine $\mathcal{D}$, si x est un homme, alors x est mortel $(\forall x(H(x) \rightarrow M(x)))$;

Hypothèse 2 : Socrate à la propriété d'être un homme $(H(Socrate))$;

Conclusion : Socrate à la propriété d'être mortel $(M(Socrate))$.

Dans cet exemple, nous avons utilisé deux prédicats (H et M), une constante ($Socrate$), une variable (x) et le quantificateur universel ($\forall$). Comme nous le verrons plus tard, ce raisonnement, ainsi formalisé en logique des prédicats, sera valide.

2.1.2 Définition d'une formule

Alors que le calcul propositionnel s'appuyait seulement sur des atomes et des connecteurs, le calcul des prédicats s'appuie sur six classes de symboles :

- **les variables :** on les note en générale x,y,z,... ;
- **les symboles de constantes :** ils sont notés a,b,c,... ;
- **les symboles de fonction :** les symboles de fonctions d'arité $n > 0$ sont notés f,g,... ;
- **les symboles de prédicats :** les prédicats, d'arité positive ou nulle, sont notés p,q,... (un prédicat d'arité nulle est équivalent à un atome du calcul propositionnel) ;
- **les connecteurs :** les connecteurs sont ceux du calcul propositionnel ;
- **les quantificateurs :** les quantificateurs (ou quanters) sont $\forall$ et $\exists$.

Définition 2.2 -Terme- *Un terme est défini récursivement par :*

- *une variable est un terme ;*
- *un symbole de constante est un terme ;*
- *si f est un symbole de fonction d'arité n et si t_1,t_2,...,t_n sont des termes, alors $f(t_1,t_2,\ldots,t_n)$ est un terme ;*

Tout terme est obtenu par l'application des règles précédentes un nombre fini de fois.

Exemple

Par exemple, $Socrate$, x, y, $Carré(x)$, $Plus(x,y)$, $Plus(x,Carré(y))$, peuvent être des termes. Ici $Carré$ est une fonction d'arité 1 qui renvoie le carré de sont argument et $Plus$ est une fonction d'arité 2 qui renvoie la somme de ses deux arguments.

La définition des formules [1] en calcul des prédicats est très proche de la définition des formules en calcul propositionnel. Nous commençons par définir les atomes :

Définition 2.3 -Atome- *Si p est un symbole de prédicat d'arité n et que t_1,t_2,...,t_n sont des termes, alors $p(t_1,t_2,\ldots,t_n)$ est un atome, ou formule atomique.*

1. nous parlons ici de formules bien formées, bien entendu.

Définition 2.4 -Formule- *Une formule est définie récursivement par :*

- *toute formule atomique est une formule.*
- *si A et B sont des formules, alors $(A \rightarrow B)$, $(A \leftrightarrow B)$, $(A \wedge B)$ et $(A \vee B)$ sont des formules ;*
- *si A est une formule, alors $(\neg A)$ est une formule.*
- *si A est une formule et si x est une variable quelconque alors $(\forall x\, A)$ est une formule, on dit que A est la porté du quantificateur $\forall x$;*
- *si A est une formule et si x est une variable quelconque alors $(\exists x\, A)$ est une formule, on dit que A est la porté du quantificateur $\exists x$;*

Toute formule est obtenue par l'application des règles précédentes un nombre fini de fois.

Exemple

Par exemple, $\forall x\ Inf(x,Plus(x,x))$ ou encore $\forall x\ (H(x) \rightarrow M(x))$ sont des formules.

Parenthèsage

De la même manière qu'en logique des propositions, nous accepterons les formules dont le parenthèsage est partiellement ou complètement implicite. Pour cela, **les connecteurs et les quantificateurs sont traditionnellement classés de la façon suivante (par priorité décroissante des connecteurs et quantificateurs) :** $\forall$, $\exists$, $\neg$, $\wedge$, $\vee$, $\rightarrow$, $\leftrightarrow$. **Dans le cas où deux connecteurs ont même priorité, et en l'absence de parenthèse, l'associativité se fait de gauche à droite**. Ainsi, $\neg\forall x\ A \vee B$ doit se lire $((\neg(\forall x\ A)) \vee B)$.

2.1.3 Travaux dirigés (formules bien formées)

Dans exercices qui suivent, on posera que x, y, z sont des variables, que a, b, c sont des constantes, que f, g, h sont des fonctions et que p, q, r sont des prédicats.

Termes

Dites si les « écritures » suivantes correspondent à des termes :

1. y
2. c
3. $x \vee a$
4. $f(x)$
5. $f(x,a)$
6. $f(a \wedge b)$
7. $f(y(a),b)$
8. $g(h(x),f(x,y),c)$
9. $f(f(f(f(x))))$

Formules atomiques

Dites si les « écritures » suivantes correspondent à des formules atomiques :

1. p
2. $p(a)$
3. $p(x)$
4. $p(a \vee b)$
5. $q(x,y,b)$
6. $r(f(x))$
7. $q(g(f(a)),b)$
8. $p(b) \vee q(a)$

Formules

Dites si les « formules » suivantes sont des formules bien formées :

1. $a \vee b$
2. p
3. $p \vee r$
4. $\forall x\ p(x)$
5. $\forall x \vee q(y)$
6. $\exists x\ p(x) \vee q(y)$
7. $q(x,y,b) \rightarrow p(x) \vee p(a) \wedge r$
8. $\forall b\ q(x,y,b) \rightarrow p(x) \vee p(a) \wedge r$
9. $\exists y\ q(x,y,b) \rightarrow p(x) \vee p(a) \wedge r$
10. $\forall z \forall x\ (\exists y\ q(x,y,b) \rightarrow p(z) \vee p(a) \wedge r)$

2.1.4 Variables liées, variables libres

Considérons la formule élémentaire $\forall x p(x,y)$ où p est un symbole de prédicat à deux variables. L'occurrence de la variable x dans $p(x,y)$ est placée dans la portée du quantificateur $\forall$. Une telle occurrence de variable est dite *liée*.

Définition 2.5 -Variable liées- *L'ensemble des variables liées d'une formule A, noté $\mathcal{B}(A)$, se définit de façon inductive par :*

- *si A est un atome, $\mathcal{B}(A) = \phi$;*
- *si A est de la forme $(B \rightarrow C)$, $(B \leftrightarrow C)$, $(B \wedge C)$, $(B \vee C)$, alors $\mathcal{B}(A) = \mathcal{B}(B) \cup \mathcal{B}(C)$;*
- *si A est de la forme $(\neg B)$ alors $\mathcal{B}(A) = \mathcal{B}(B)$;*
- *si A est de la forme $(\forall x\ B)$ ou $(\exists x\ B)$, alors $\mathcal{B}(A) = \{x\} \cup \mathcal{B}(B)$.*

Définition 2.6 -Variable libre- *L'ensemble des variables libres d'une formule A, noté $\mathcal{F}(A)$, se définit de façon inductive par :*

- *si A est un atome, $\mathcal{F}(A)$ est égal à l'ensemble des variables apparaissant dans A;*

- *si A est de la forme $(B \to C)$, $(B \leftrightarrow C)$, $(B \wedge C)$, $(B \vee C)$, alors $\mathcal{F}(A) = \mathcal{F}(B) \cup \mathcal{F}(C)$;*
- *si A est de la forme $(\neg B)$ alors $\mathcal{F}(A) = \mathcal{F}(B)$;*
- *si A est de la forme $(\forall x\ B)$ ou $(\exists x\ B)$, alors $\mathcal{F}(A) = \mathcal{F}(B) - \{x\}$.*

Définition 2.7 -Formule close- *Une formule A telle que $\mathcal{F}(A) = \phi$ est dite close, fermée ou est encore appelée énoncé.*

2.1.5 Travaux dirigés (variables liées ou libres)

Dans les formules suivantes, dites si les variables utilisées sont des variables libres ou des variables liées.

- $p(x) \vee q(y)$
- $\forall x p(x) \vee q(y)$
- $\forall x p(x) \vee q(y,z)$
- $\exists z(\forall x p(x,z) \vee q(y,z))$
- $\exists z(p(x,z) \vee \exists y q(y,z))$
- $\forall x p(x) \vee q(x)$
- $p(x,y) \vee p(y,x) \wedge p(x,x) \to p(x,y)$
- $\forall x \forall y(p(x,y) \vee p(y,x)) \wedge p(x,x) \to p(x,y)$
- $\forall x \forall y(p(x,y) \vee p(y,x)) \wedge p(x,x) \to \exists x \forall y p(x,y)$
- $\forall x \forall y(p(x,y) \vee p(y,x)) \wedge \forall x p(x,x) \to \exists x \forall y p(x,y)$

2.1.6 Clôture d'une formule

Nous allons rencontrer rapidement une difficulté concernant les variables libres. Il est en effet possible d'interpréter une variable libre de deux façons dans une formule.

Si un théorème est démontré à partir d'une hypothèse $A(x)$ où x est pris dans le sens conditionnel, c'est à dire où x est contraint à prendre ses valeurs dans un sous-ensemble des valeurs possibles, alors le théorème ne s'applique que pour le même x. En revanche, un théorème démontré à partir de $A(x)$ où x est pris au sens de la généralité est valable pour tout x. Il faut bien prendre garde à ce type de phénomènes, source de sophisme logiques. Il est clair que nombre de problèmes proviennent de l'emploi de variables libres. Si nous nous contraignions à n'utiliser que des variables liées, l'ambiguïté disparaîtrait sur le champ.

On peut toujours ramener une formule non close à une formule close via les deux notions qui suivent.

Définition 2.8 -Clôture universelle- *On appelle clôture universelle de la formule A la formule liée $\forall x_1 \forall x_2 \ldots \forall x_q$ où x_1,x_2,...,x_q sont les variables libres apparaisant dans A.*

Définition 2.9 -Clôture existentielle- *On appelle clôture existentielle de la formule A la formule liée $\exists x_1 \exists x_2 \ldots \exists x_q$ où x_1,x_2,...,x_q sont les variables libres apparaisant dans A.*

2.2 Théorie des modèles

2.2.1 Introduction

Comme en calcul propositionnel, nous allons nous efforcer de construire un modèle permettant de dégager une interprétation sémantique de nos formules.

Cependant, en calcul des prédicats, il n'est pas possible d'appliquer une méthode des tables de vérité directement dérivée du calcul propositionnel, en raison des domaines de valeur des variables de chacun des prédicats. En fait, pour une formule atomique $p(x_1, \ldots, x_n)$, nous allons avoir besoin d'une fonction (appelée fonction d'interprétation) chargée de donner un sens au symbole p, et donc de calculer sa valeur de vérité selon la valeur des $x_1, \ldots, x_n$.

2.2.2 Interprétation

Définition 2.10 -Interprétation- *Une interprétation d'une formule F est caractérisée par la donnée d'un ensemble de définition $\mathcal{D}$ non vide, appelé le domaine de l'interprétation et d'une fonction d'interprétation I qui associe :*

- *à chaque symbole de constante de F, un élément de $\mathcal{D}$;*
- *à chaque symbole de fonction f d'arité n de F, le graphe d'une fonction de $\mathcal{D}^n \rightarrow \mathcal{D}$ définissant f ;*
- *à chaque symbole de prédicat p d'arité n de F, le graphe d'une fonction de $\mathcal{D}^n \rightarrow \{\mathbb{T}, \mathbb{F}\}$ définissant p.*

Définition 2.11 -Interprétation des termes- *Soit une formule F et I une interprétation de cette formule. On peut étendre l'interprétation I aux termes de F :*

- *à chaque symbole de constante, on associe sa valeur selon I ;*
- *à chaque variable, on associe la variable elle-même ;*
- *à chaque terme $f(t_1,\ldots,t_n)$, on associe le terme $f'(t'_1,\ldots,t'_n)$ où $t'_1,\ldots,t'_n$ sont les interprétations des $t_1,\ldots,t_n$ et f' est l'interprétation de f.*

Définition 2.12 -Interprétation des formules- *Elle est définie par :*

- *si F est un atome $p(t_1,\ldots,t_n)$, $I(F)$ est la fonction $p'(t'_1,\ldots,t'_n)$ où p' est l'interprétation de p et où chaque t'_i est l'interprétation de t_i ;*
- *si F est de la forme $(\neg G)$, $(G \to H)$, $(G \leftrightarrow H)$, $(G \wedge H)$, $(G \vee H)$, $I(F)$ est définie par les mêmes lois fonctionnelles que celles définies pour le calcul propositionnel ;*
- *si F est de la forme $\forall x G(x,y_1,\ldots,y_n)$, pour tout n- uplet $(a_1,\ldots,a_n) \in \mathcal{D}^i$, $I(F)(a_1,\ldots,a_n) = \mathbb{T}$ si pour toute valeur $a \in \mathcal{D}$, $I(G)(a,a_1,\ldots,a_n) = \mathbb{T}$,*
  $I(F)(a_1,\ldots,a_n) = \mathbb{F}$ sinon ;
- *si F est de la forme $\exists x G(x,y_1,\ldots,y_n)$, pour tout n- uplet $(a_1,\ldots,a_n) \in \mathcal{D}^i$, $I(F)(a_1,\ldots,a_n) = \mathbb{T}$ s'il existe une valeur $a \in \mathcal{D}$, $I(G)(a,a_1,\ldots,a_n) = \mathbb{T}$,*
  $I(F)(a_1,\ldots,a_n) = \mathbb{F}$ sinon ;

2.2.3 Validité et consistance

Définition 2.13 -Formule valide- *Une formule F est dite valide ssi, pour toute interprétation I, on a $I(F) = \mathbb{T}$.*

Définition 2.14 -Formule satisfiable- *Une formule A sera dite satisfiable, ou **sémantiquement consistante**, s'il existe une interprétation I telle que $I(A) = \mathbb{T}$. L'interprétation est alors un modèle de A.*

Définition 2.15 -Formules invalides- *Une formule invalide est fausse dans au moins une interprétation.*

Définition 2.16 -Formule insatisfiable- *Une formule insatisfiable, ou **sémantiquement inconsistante**, ou encore **antitautologie**, est une formule fausse dans toute interprétation.*

Définition 2.17 -Formules contingentes- *Une formule contingente est vraie dans certaines interprétations et fausse dans d'autres.*

Définition 2.18 -Conséquence logique- *Soit une formule B et une famille de n formules A_i ; On dit que B est conséquence des A_i si pour toute interprétation I telle que $\forall A_i, I(A_i) = \mathbb{T}$, on a aussi $I(B) = \mathbb{T}$. On note alors $A_1,A_2,\ldots,A_n \models B$.*

2.2.4 Table de vérité

Exemple

Soit la formule $A = p(y) \vee \forall x(p(x) \to q)$ où p est défini sur le domaine $\mathcal{D} = \{1,2\}$. Nous allons essayer de montrer que cette formule n'est pas valide. Pour cela nous allons construire la table de vérité de A. Tout d'abord nous devons considérer les quatre

x	$I_1(p)(x)$	$I_2(p)(x)$	$I_3(p)(x)$	$I_4(p)(x)$
1	$\mathbb{T}$	$\mathbb{T}$	$\mathbb{F}$	$\mathbb{F}$
2	$\mathbb{T}$	$\mathbb{F}$	$\mathbb{T}$	$\mathbb{F}$

TAB. 2.1 – *Interprétations possibles*

interprétations possibles pour p (table 2.1). Nous pouvons ensuite nous lancer dans la construction de la table de vérité de A (table 2.2) en prenant en compte les différentes interprétations possibles de p et de q et les différentes valeurs possibles de y.

Détaillons le calcul des deux premières lignes par exemple. L'interprétation de p utilisé est I_1 et on suppose que l'interprétation de q est $\mathbb{T}$. L'interprétation de $p(x) \rightarrow q$ est une fonction d'une variable x et est toujours égale à $\mathbb{T}$ d'après l'interprétation I_1. On en déduit que la formule close $\forall x(p(x) \rightarrow q)$ s'interprète par la fonction constante $\mathbb{T}$. L'interprétation de $p(y)$ est une fonction d'une variable y constante égale à $\mathbb{T}$ (d'après I_1). On en déduit que sous I_1, l'interprétation de la formule A est la fonction d'une variable constante et égale à $\mathbb{T}$. D'où les deux premières lignes.

On déduit du contenu de la table que la formule n'est pas valide puisqu'il existe un domaine et une interprétation I (telle que $I(p) = I_2(p)$ et $I_q = \mathbb{F}$) pour lesquels l'interprétation de A n'est pas la fonction constante $\mathbb{T}$.

$I(p)$	$I(q)$	y	$I(\forall x(p(x) \rightarrow q))$	$I(p(y) \vee \forall x(p(x) \rightarrow q))(y)$
$I_1(p)$	$\mathbb{T}$	1	$\mathbb{T}$	$\mathbb{T}$
$I_1(p)$	$\mathbb{T}$	2	$\mathbb{T}$	$\mathbb{T}$
$I_1(p)$	$\mathbb{F}$	1	$\mathbb{F}$	$\mathbb{T}$
$I_1(p)$	$\mathbb{F}$	2	$\mathbb{F}$	$\mathbb{T}$
$I_2(p)$	$\mathbb{T}$	1	$\mathbb{T}$	$\mathbb{T}$
$I_2(p)$	$\mathbb{T}$	2	$\mathbb{T}$	$\mathbb{T}$
$I_2(p)$	$\mathbb{F}$	1	$\mathbb{F}$	$\mathbb{T}$
$I_2(p)$	$\mathbb{F}$	2	$\mathbb{F}$	$\mathbb{F}$
$I_3(p)$	$\mathbb{T}$	1	$\mathbb{T}$	$\mathbb{T}$
$I_3(p)$	$\mathbb{T}$	2	$\mathbb{T}$	$\mathbb{T}$
$I_3(p)$	$\mathbb{F}$	1	$\mathbb{F}$	$\mathbb{F}$
$I_3(p)$	$\mathbb{F}$	2	$\mathbb{F}$	$\mathbb{T}$
$I_4(p)$	$\mathbb{T}$	1	$\mathbb{T}$	$\mathbb{T}$
$I_4(p)$	$\mathbb{T}$	2	$\mathbb{T}$	$\mathbb{T}$
$I_4(p)$	$\mathbb{F}$	1	$\mathbb{T}$	$\mathbb{T}$
$I_4(p)$	$\mathbb{F}$	2	$\mathbb{T}$	$\mathbb{T}$

TAB. 2.2 – *Table de vérité de* $p(y) \vee \forall x(p(x) \rightarrow q)$

Problème des domaines infinis

Vérifier la validité d'une formule dans le cas où les domaines de ses variables sont finis ne pose aucun problème, du moins d'ordre théorique. Il suffit de construire, comme en calcul propositionnel, une table de vérité.

En revanche si un ou plusieurs des domaines sont infinis, le problème est tout autre. On ne peut plus construire de table de vérité et l'on est forcé de tenir des raisonnements d'ordre général pour démontrer la validité d'une formule (pour démontrer qu'elle n'est pas valide, il suffit bien sûr d'exhiber un cas où elle est fausse).

Cette particularité rend la théorie des modèles du calcul des prédicats beaucoup plus délicate à manipuler que la théorie des modèles du calcul propositionnel.

2.2.5 Travaux dirigés (théorie des modèles)

2.2.6 Traduction

Traduire les énoncés suivants dans le langage de la logique des prédicats :

1. Tout est relatif.
2. Il n'est point de petites affaires.
3. Si on aime la vie, on craint la mort.
4. Un sot trouve toujours un plus sot qui l'admire.
5. Bien mal acquis ne profite jamais.
6. Chaque âge a ses plaisirs, son esprit et ses mœurs.
7. Il y a un remède à tout, hors la mort.
8. Ce qui vient du diable retourne au diable.

2.2.7 Interprétation

Soit la formule suivante :

$$F = \forall x \exists y p(x,y) \rightarrow \exists y p(yy)$$

– Dans les interprétations suivantes, dites si la formule F est vraie ou fausse :
  1. $\mathcal{D} = \{Hommes\}$ et $p(x,y)$ signifie x est le père de y.
  2. $\mathcal{D} = \{Hommes\}$ et $p(x,y)$ signifie y est le père de x.
  3. $\mathcal{D} = \mathbb{R}$ et $p : \mathbb{R} \times \mathbb{R} \rightarrow \{\mathbb{T}, \mathbb{F}\}$ tel que $p(x,y)$:"$x \neq y$"
  4. $\mathcal{D} = \{\alpha\}$ et $p : \mathcal{D} \times \mathcal{D} \rightarrow \{\mathbb{T}, \mathbb{F}\}$ tel que $p(\alpha,\alpha) = \mathbb{T}$
  5. $\mathcal{D} = \{\alpha\}$ et $p : \mathcal{D} \times \mathcal{D} \rightarrow \{\mathbb{T}, \mathbb{F}\}$ tel que $p(\alpha,\alpha) = \mathbb{F}$

– Que peut on dire, en terme de validité et de consistance, au sujet de la formule F ?

– Que pensez vous de la formule suivante :

$$\forall x A(x) \rightarrow \exists y A(y)$$

2.3 Équivalence des formules bien formées

2.3.1 Formules équivalentes

Définition 2.19 -Formules équivalentes- *Deux formules sont équivalentes quand elles ont même valeur dans toutes interprétation (notation : $A \equiv B$).*

Propriété 2.1 -Formules équivalentes- *Soient $A(x)$ et $B(x)$ deux formules atomiques bien formées.*

- *Les formules équivalentes de la logique des propositions demeurent équivalentes en logique des prédicats.*
- $\forall x A(x) \wedge \forall x B(x) \equiv \forall x (A(x) \wedge B(x))$
- $\exists x A(x) \vee \exists x B(x) \equiv \exists x (A(x) \vee B(x))$
- $\neg(\forall x A(x)) \equiv \exists x \neg A(x)$
- $\neg(\exists x A(x)) \equiv \forall x \neg A(x)$

Remarque :

- ★ $\forall x A(x) \vee \forall x B(x) \not\equiv \forall x (A(x) \vee B(x))$ ★
- ★ $\exists x A(x) \wedge \exists x B(x) \not\equiv \exists x (A(x) \wedge B(x))$ ★

2.3.2 Formes prénexes

La principale différence syntaxique entre calcul propositionnel et calcul des prédicats est la présence de quantificateurs à l'intérieur des formules. L'idée à la base de cette section est de construire une formule équivalente dans laquelle les quantificateurs seront rejetés en tête de la formule.

Définition 2.20 -Matrice- *Une matrice est une formule du calcul des prédicats ne contenant aucun quantificateur.*

Définition 2.21 -Forme prénexe- *Une forme prénexe est une formule du calcul des prédicats de la forme $\mathcal{Q}_1 x_1 \ldots \mathcal{Q}_n x_n M$, où $\mathcal{Q}$ désigne $\exists$ ou $\forall$ et M est une matrice.*

Théorème 2.1 -Forme prénexe équivalente- *Toute formule admet une forme prénexe équivalente.*

Algorithme de construction - A partir d'une formule quelconque, voici comment construire la forme prénexe associée.

1. Supprimer les connecteurs d'équivalence et d'implication.
2. Renommer certaines variables liées de manière à n'avoir plus de variable quantifiée deux fois.
3. Supprimer les quantificateurs inutiles (dont la variable quantifiée n'apparaît pas dans leur portée)s'il y en a.
4. Transférer toutes les occurrences de la négation devant les atomes en utilisant les règles de réécriture vues pour la logique des propositions et les règles supplémentaires suivantes :
   - $\neg \forall x A \rightsquigarrow \exists x \neg A$;

– $\neg\exists x A \rightsquigarrow \forall x \neg A$.

5. faire passer les quantificateurs en tête en utilisant les règles de réécriture suivantes (et en utilisant l'associativité, la commutativité ou le renommage de variable si nécessaire) :
   – $(\forall x A \wedge B) \rightsquigarrow \forall x (A \wedge B)$ si B ne contient pas x ;
   – $(\exists x A \wedge B) \rightsquigarrow \exists x (A \wedge B)$ si B ne contient pas x ;
   – $(\forall x A \vee B) \rightsquigarrow \forall x (A \vee B)$ si B ne contient pas x ;
   – $(\exists x A \vee B) \rightsquigarrow \exists x (A \vee B)$ si B ne contient pas x.

Exemple

Nous allons construire la forme prénexe équivalente à la formule :

$$\forall x p(x) \wedge \exists y q(y) \rightarrow \exists y (p(y) \wedge q(y))$$

1. $\neg(\forall x p(x) \wedge \exists y q(y)) \vee \exists y (p(y) \wedge q(y))$ par suppression de $\rightarrow$;
2. $\neg(\forall x p(x) \wedge \exists y q(y)) \vee \exists z (p(z) \wedge q(z))$ par renommage des variables ;
3. $(\exists x \neg p(x) \vee \forall y \neg q(y)) \vee \exists z (p(z) \wedge q(z))$ par transfert de la négation ;
4. $\exists x \forall y \exists z (\neg p(x) \vee \neg q(y) \vee (p(z) \wedge q(z)))$ par déplacement des quantificateurs.

2.4 Travaux dirigés (Logique des prédicats)

2.4.1 Compréhension

Expliquez les deux inégalités de la remarque de la section 2.3.1 : trouvez des exemples qui l'illustre.

2.4.2 Traduction I

En utilisant le prédicat $mieux\_vaut(x,y)$ (lire x vaut mieux que y) traduire les énoncés suivants dans le langage de la logique des prédicats :

1. Mieux vaut une médaille de bronze qu'aucune médaille.
2. Rien est mieux qu'une médaille d'or.
3. Une médaille d'argent est préférable à une médaille de bronze.

2.4.3 Traduction II

Traduire les raisonnements suivants dans le langage de la logique des prédicats :

1. Tous les politiciens parlent à la radio.
   Parmi les gens qui parlent à la radio, certains sont des menteurs.
   Donc, quelques politiciens sont menteurs.
2. Seuls les oiseaux ont des plumes.
   Aucun mammifère n'est un oiseau.
   Donc, tout mammifère est dépourvu de plumes.

3. Toutes les médailles d'or sont remportées par quelqu'un.
   Tous ceux qui ont gagné des médailles d'or sont des champions.
   Les gens fatigués en gagnent pas de médailles d'or.
   Il existe des médailles d'or.
   Il y a donc des champions qui ne sont pas fatigués.

2.4.4 Forme prénexe

Mettre sous forme prénexe les formules suivantes :

1. $\forall x A(x) \rightarrow \exists y A(y)$
2. $\neg\forall x(A(x) \vee B(x)) \rightarrow \exists x A(x)$
3. $\forall x \exists y P(x,y) \rightarrow \exists y P(yy)$
4. $\forall x(P(x) \wedge Q(x) \rightarrow \exists y(S(xy) \wedge \forall z R(zyx)))$

Chapitre 3

Programmation logique : PROLOG

3.1 Présentation

PROLOG est né d'un pari : créer un langage de très haut niveau, même inefficace au sens des informaticiens de l'époque. L'efficacité consistait alors à faire exécuter très rapidement par une machine des programmes écrits laborieusement. Le pari était donc de pouvoir écrire rapidement des programmes, quitte à ce que la machine les exécute laborieusement.

PROLOG signifie *PRO*grammation en *LOG*ique. Le nom du langage est significatif. Programmer en PROLOG, c'est exprimer sous forme de relations logiques les différentes propriétés d'un système. Une fois le problème formalisé, le langage sera, le plus souvent, capable de fournir les réponses aux questions posées par l'utilisateur en utilisant une méthode de résolution automatique.

La programmation en PROLOG est assez différente de la programmation dans un language classique. PROLOG est un language dit *déclaratif*, en ce sens qu'on ne décrit pas comme dans les langages traditionnels (dits *procéduraux)* une méthode (ou algorithme) de résolution, mais que l'on déclare les propriétés du problème, le système se chargeant lui-même d'appliquer une méthode de résolution.

Les formules logiques que PROLOG peut utiliser sont les clauses de Horn de la logique des prédicats telles que nous les avons décrites au chapitre 1. PROLOG met en application les techniques de résolution et d'unification de la logique des prédicats (nous n'avons pas abordé ces technique pour la logique des prédicats, mais nous avons vu la résolution en logique des propositions).

PROLOG a vu le jour en 1971 sur une idée d'Alain Colmerauer qui travaillait alors aux problèmes de traduction automatique au Groupe Intelligence Artificielle de Marseille-Luminy. Le premier langage PROLOG a été créé par Alain Colmerauer avec l'aide de Philippe Roussel dans le cadre d'un contrat de communication homme-machine.

Nous nous basons pour présenter PROLOG sur le système SWI-PROLOG (`http://www.swi.psy.uva.nl/projects/SWI-Prolog/`).

3.2 Terminologie

L'objet de cette section est de décrire rapidement l'emploi de certains termes dans le contexte PROLOG.

Base de faits - On appelle base de fait l'ensemble des faits relatifs au problème que l'utilisateur formalise en PROLOG. « Médor est un chien » peut se représenter par un fait. Formellement, les faits sont des clauses de Horn positives.

Base de règles - On appelle base de règles l'ensemble des règles relatives au problème que l'utilisateur formalise en PROLOG. « Un chien est un mammifère » peut se représenter par une règle. Les règles sont des clauses de Horn strictes.

Question - La question est ce que l'utilisateur demande au système. C'est une clause de Horn négative.

Moteur d'inférence de PROLOG - Le moteur d'inférence de PROLOG est la partie du système qui réalise les inférences logiques sur les faits et la question posée par l'utilisateur, de façon à donner une (ou plusieurs) réponses. Il opère sur un ensemble de faits et de règles, donc un ensemble de clauses de Horn négatives ou strictes. Sur un plan théorique, le moteur PROLOG fait une preuve d'inconsistance sur la base de clauses de Horn.

3.3 Syntaxe

Nous allons ici présenter la syntaxe du SWI-PROLOG de manière informelle et simplifiée.

Les variables - Une variable commence par une majuscule ou le signe « _ ». Par exemple, `X`, `XA`, `A123`, `_a` sont des symboles de variables autorisés en SWI-PROLOG tandis que `ab`, `a1b`, `r2d2` sont invalides.

Les symboles de prédicats - un symbole de prédicat est un mot commencent par une minuscule. Il peut avoir des arguments, auquel cas il est suivi d'une parenthèse ouvrante, de la liste des arguments séparés par des virgules et d'une parenthèse fermante. Les arguments peuvent être des variables comme X ou des symboles de constantes comme `toto` par exemple. Les symboles de fonctions et de constantes ont la même structure syntaxique que les prédicats, ambiguïté qui peut parfois porter à confusion. Par exemple, `chien(X)`, `toto(A,B,C)`, `truc(fop(X), Y)` sont des utilisations de symboles de prédicats correctes.

Les listes - une liste est une suite d'objets entre « [] » séparés par des « , ». Les parenthèses sont utilisées pour délimiter les sous-listes. Par exemple, les listes `[a,b,toto,2,]` et `[a,(b,c),d,]` sont des listes syntaxiquement correctes.

Les faits - Un fait s'écrit `predicat_tete.`. Un exemple de fait correctement exprimé est `chien(fido).` pour signifier que fido est un chien.

Les règles - Une règle s'écrit `predicat_tete :- predicat_1, predicat_2 ...` Un exemple de prédicat correctement exprimé est `mammifere(X) :- chien(X) .`. Ce prédicat peut signifier « pour tout X, si X est un chien alors X est un mammifère ».

Attention, le signe le signe `:-`, qui s'écrit souvent `->` sur d'autres interpréteurs PROLOG, n'est pas un signe d'implication ! Il peut s'interpréter comme un signe de réécriture. En

logique nous écririons la dernière clause $chien(X) \rightarrow mammifere(X)$. Là encore, la syntaxe du langage peut induire en erreur.

3.4 Premier programme (SWI-PROLOG)

Voici un premier exemple de programme PROLOG :

```
chien(fido).
mammifere(X) :- chien(X).
```

La première ligne est un fait qui déclare que fido est un chien. La seconde ligne est une règle qui déclare que tout chien est un mammifère.
Une question que l'on pourrait poser serait :

```
?- mammifere(fido).
```

La réponse serait :

```
Yes
```

Il faut interpréter cette réponse comme *vraie*. On pourrait poser la question :

```
> mammifere(X).
```

La réponse serait :

```
X = fido
```

3.5 Travaux pratiques (Arbre généalogique)

Le but de ce TP est de représenter en PROLOG un arbre généalogique, puis d'écrire des relations qui permettront d'interroger cette base de donnée.

L'arbre généalogique à représenter est celui de la figure 3.1.

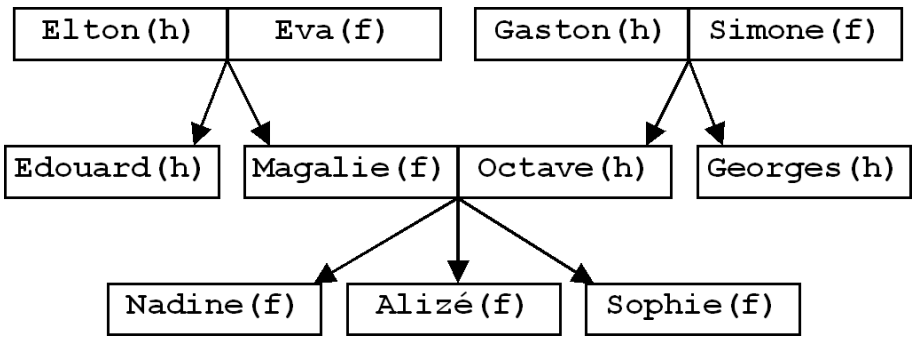

FIG. 3.1 – *Arbre généalogique*

Complétez la base de faits suivante pour que l'intégralité de l'arbre généalogique soit représenté :

```
homme(elton).
homme(gaston).
...

femme(eva).
```

```
femme(simone).
femme(magali).
...

parent(elton,edouard).
parent(elton,magali).
parent(eva,edouard).
parent(eva,magali).
...
```

Dans ce program `homme(X)` signifie que `X` est un homme, `femme(X)` signifie que `X` est une femme et `parent(X,Y)` signifie que `X` est un `parent` de `Y`.

En utilisant la convention d'écriture `relation(X,Y)` signifiant « `X` est `relation` de `Y`», établir les règles suivantes :

1. `enfant(X,Y)` ;
2. `fils(X,Y)` ;
3. `fille(X,Y)` ;
4. `pere(X,Y)` ;
5. `mere(X,Y)` ;
6. `grand_parent(X,Y)` ;
7. `grand_pere(X,Y)` ;
8. `grand_mere(X,Y)` ;
9. `petit_enfant(X,Y)` ;
10. `petit_fils(X,Y)` ;
11. `petite_fille(X,Y)` ;
12. `frere(X,Y)` ;
13. `soeur(X,Y)` ;
14. `oncle(X,Y)` ;
15. `tante(X,Y)`.

Pensez à tester vos règles au fur et à mesure !

3.6 Travaux pratiques (Règles syntaxiques en PROLOG)

3.6.1 Micro grammaire en PROLOG

Micro grammaire

Voici un exemple de micro-grammaire pour décrire quelques phrases simples en français :

```
phrase --> syntagme_nominal, syntagme_verbal.

syntagme_nominal --> article, nom_commun.

syntagme_verbal --> verbe, syntagme_nominal.
syntagme_verbal --> verbe.

article --> [le].
article --> [la].

nom_commun --> [souris].
nom_commun --> [chat].

verbe --> [mange].
verbe --> [guette].
```

Avec cette grammaire on peut générer des phrases comme :

```
le chat guette la souris.
le chat mange.
```

En PROLOG

Dans un premier temps, on aimerait avoir un programme en PROLOG qui accepte des phrases et nous dit si elles sont correctes d'après la grammaire spécifiée.

Si on convient que `phrase_correcte(X)` signifie `X` est une liste de mots formant une phrase syntaxiquement correcte d'après la grammaire spécifiée, alors nous aurions :

```
?- phrase_correcte([le, chat, mange, la, souris]).

Yes
?- phrase_correcte([le, chat, mange, souris]).

No
```

Cela pourrait être fait de la manière suivante :

- soit `append(X, Y, Z)` un prédicat prédéfinie qui est vrai quand la liste `Z` peut s'unifier avec la concaténation des listes `X` et `Y` ;
- soit `syntagme_nominal(X)` un prédicat vrai si `X` est un syntagme nominal ;
- soit `syntagme_verbal(X)` un prédicat vrai si `X` est un syntagme verbal.

La règle `phrase_correcte(X)` qui ferait exactement ce que l'on aimerait pourrait s'écrire en PROLOG :

```
phrase_correcte(P) :-
  append(GN, GV, P),
  syntagme_nominal(GN),
  syntagme_verbal(GV).
```

Ecrivez la suite du programme en PROLOG.

Testez votre programme avec les requête suivantes :

```
?- phrase_correcte([le, chat, mange, la, souris]).

?- phrase_correcte([le, chat, mange, souris]).

?- phrase_correcte([la, chat, mange, le, souris]).

?- phrase_correcte([la, souris | R]).

?- phrase_correcte(P).
```

Quelles sont vos remarque ?

Problèmes

Comme vous l'avez peut-être remarqué, notre programme est fortement perfectible. Il pose, à l'état actuel, trois problèmes que nous allons essayer de corriger dans les sections qui suivent.

Problème de performance : notre programme n'est absolument pas performant en analyse et totalement inutilisable en génération.

Décomposition syntaxique : le programme se contente de dire si une phrase est syntaxiquement correcte ou pas, mais dans le cas où une phrase est correcte, il ne donne pas sa décomposition.

Accords en genre : les accords en genre ne sont pas gérés.

3.6.2 Problème de performance

Examinons de plus près comment fonctionne notre prédicat `phrase_correcte(X)`.

En analyse (*i.e.* lorsque l'on soumet une phrase à analyser au prédicat), le prédicat `append(GN, GV, P)` va générer toutes les décomposition possible de la liste `P` en deux sous-listes `GN` et `GV`. On fait ici un grand nombres de décompositions inutiles pour n'en utiliser que très peu par la suite.

En génération (*i.e.* lorsque le prédicat doit lui même générer des phrases, ou des portions de phrases), le prédicat `append(GN, GV, P)` peut être amené à générer des listes `GN` et `GV` au hasard ce qui conduit ici à une inefficacité catastrophique.

La bonne méthode consiste à laisser le prédicat `syntagme_nominal` décider lui-même de la sous-liste dont-il a besoin. Ainsi nous définirons une règle

`syntagme_nominal(Phrase, RestePhrase)`

signifiant :

– il y a un syntagme nominal en tête de la phrase `Phrase`, et

– ce qui reste de la phrase après se syntagme nominal est `RestePhrase`.

Nous devrions alors avoir :

```
?- syntagme_nominal([le, chat, mange, la, souris], RestePhrase).

RestePhrase = [mange, la, souris]

Yes
```

Par soucis d'homogénéité, toutes nos règles, hormis `phrase_correcte(X)`, serons de la forme :

`règle(Portion, RestePortion)`

La règle `phrase_correcte(X)` qui ferait exactement ce que l'on aimerait pourrait s'écrire en PROLOG :

```
phrase_correcte(X) :-
  syntagme_nominal(X, R1),
  syntagme_verbal(R1, []).
```

La liste vide `[]` dans `syntagme_verbal(R1, [])` signifie que, pour que `X` soit une phrase correcte, il faut qu'il ne reste rien une fois enlevé le syntagme nominal et le syntagme verbal.

Ecrivez la suite du programme en PROLOG.

Testez votre programme avec les requête suivantes :

```
?- phrase_correcte([le, chat, mange, la, souris]).

?- phrase_correcte(P).
```

3.6.3 Décomposition syntaxique

Quand une phrase est correcte, il serait intéressant d'avoir sa décomposition syntaxique. Par exemple :

```
?- phrase_correcte([le, chat, mange, la, souris], DecompositionSyntaxique).

DecompositionSyntaxique = ph(sn(art([le]), nc([chat])),
                             sv(manger([mange]),sn(art([la]),
                             nc([souris]))))

Yes
```

La règle `phrase_correcte(X)` qui ferait exactement ce que l'on aimerait pourrait s'écrire en PROLOG :

```
phrase_correcte(X, ph(TRGN,TRGV)) :-
  syntagme_nominal(X, R1, TRGN),
  syntagme_verbal(R1, [], TRGV).
```

Ici `ph(TRGN,TRGV)` est un terme où `ph(x,y)` est une fonction à laquelle on passe deux arguments : la décomposition syntaxique du syntagme nominal (`TRGN`) et la décomposition syntaxique du syntagme verbal (`TRGV`).

Ecrivez la suite du programme en PROLOG puis testez le.

3.6.4 Accords en genre

Modifiez le dernier programme écrit pour prendre en compte l'accord en genre. A vous de décider, voici le résultat :

```
?- phrase_correcte(Ph, Dec).

Ph = [le, chat, mange, le, chat]
Dec = ph(sn(art([le]), nc([chat])), sv(manger([mange]),sn(art([le]), nc([chat])))) ;

Ph = [le, chat, mange, la, souris]
Dec = ph(sn(art([le]), nc([chat])), sv(manger([mange]), sn(art([la]), nc([souris])))) ;

Ph = [le, chat, guette, le, chat]
Dec = ph(sn(art([le]), nc([chat])), sv(guetter([guette]), sn(art([le]), nc([chat])))) ;

Ph = [le, chat, guette, la, souris]
Dec = ph(sn(art([le]), nc([chat])), sv(guetter([guette]), sn(art([la]), nc([souris])))) ;

Ph = [le, chat, mange]
Dec = ph(sn(art([le]), nc([chat])), sv(manger([mange]))) ;

Ph = [le, chat, guette]
Dec = ph(sn(art([le]), nc([chat])), sv(guetter([guette]))) ;

Ph = [la, souris, mange, le, chat]
Dec = ph(sn(art([la]), nc([souris])), sv(manger([mange]), sn(art([le]), nc([chat])))) ;

Ph = [la, souris, mange, la, souris]
Dec = ph(sn(art([la]), nc([souris])), sv(manger([mange]), sn(art([la]), nc([souris])))) ;

Ph = [la, souris, guette, le, chat]
Dec = ph(sn(art([la]), nc([souris])), sv(guetter([guette]), sn(art([le]), nc([chat])))) ;

Ph = [la, souris, guette, la, souris]
Dec = ph(sn(art([la]), nc([souris])), sv(guetter([guette]), sn(art([la]), nc([souris])))) ;

Ph = [la, souris, mange]
Dec = ph(sn(art([la]), nc([souris])), sv(manger([mange]))) ;

Ph = [la, souris, guette]
Dec = ph(sn(art([la]), nc([souris])), sv(guetter([guette]))) ;
```

Annexe A

Correction des travaux dirigés : logique des propositions

A.1 Travaux dirigés 1.2.2

Formules bien formées

Les « formules » suivantes sont-elle des formules bien formées ?

1. $a \vee b \wedge c$ ..... : oui.
2. $\neg a \vee b \wedge c$ ..... : oui.
3. $a \vee \neg b \wedge c$ ..... : oui.
4. $a\neg \vee b \wedge c$ ..... : non.
5. (a) ..... : oui.
6. $(a)b$ ..... : non.
7. $\neg b(a)$ ..... : non.
8. $a \vee \neg(b \wedge c)$ ..... : oui.
9. $a\neg(\vee b \wedge c)$ ..... : non.
10. $a \rightarrow b$ ..... : oui.
11. $a \leftarrow a$ ..... : non.
12. $a \rightarrow b \leftrightarrow c$ ..... : oui.
13. $a \rightarrow \neg(b \leftrightarrow c)$ ..... : oui.
14. $a\neg \rightarrow b$ ..... : non.
15. $a \vee (b \leftrightarrow c)\neg c \rightarrow d$ ..... : non.
16. $a \vee (b \leftrightarrow c) \rightarrow \neg c \rightarrow d$ .. : oui.

Parenthèsage

Explicitez le parenthèsage implicite des formules suivantes :

1. $a \rightarrow b \rightarrow c \equiv ((a \rightarrow b) \rightarrow c)$
2. $a \vee b \wedge c \equiv (a \vee (b \wedge c))$
3. $a \vee b \wedge c \leftrightarrow d \rightarrow \neg\ e \vee f \wedge g \equiv ((a \vee (b \wedge c)) \leftrightarrow (d \rightarrow ((\neg e) \vee (f \wedge g))))$

Simplifiez au maximum le parenthèsage des formules suivantes :

1. $(a) \equiv a$
2. $((a \vee b)) \equiv a \vee b$
3. $((a) \wedge (b)) \equiv a \wedge b$
4. $(\neg(a \vee b)) \equiv \neg(a \vee b)$
5. $\neg(((a) \wedge b)) \equiv \neg(a \wedge b)$
6. $a \vee (b \wedge c) \equiv a \vee b \wedge c$
7. $(a \vee b) \wedge c \equiv (a \vee b) \wedge c$
8. $((a \wedge b) \rightarrow c) \equiv a \wedge b \rightarrow c$
9. $(a \wedge (b \rightarrow c)) \equiv a \wedge (b \rightarrow c)$
10. $((a \vee b) \wedge c) \leftrightarrow e \equiv (a \vee b) \wedge c \leftrightarrow e$
11. $(((a \vee b) \wedge c) \leftrightarrow e) \rightarrow f \equiv ((a \vee b) \wedge c \leftrightarrow e) \rightarrow f$
12. $((a \wedge b) \vee c) \leftrightarrow (e \rightarrow f) \equiv a \wedge b \vee c \leftrightarrow e \rightarrow f$
13. $(((a \rightarrow b) \rightarrow c) \rightarrow d) \equiv a \rightarrow b \rightarrow c \rightarrow d$
14. $(a \wedge (b \wedge c)) \equiv a \wedge b \wedge c$
15. $(a \rightarrow (b \rightarrow c)) \equiv a \rightarrow (b \rightarrow c)$

Formalisation d'un énoncé

Traduisez les énoncés suivants en formule logique :

1. Quand il fait beau, Jean est heureux ;
   il fait soleil ;
   donc, Jean est heureux.
2. Quand il fait beau, Jean est heureux ;
   or, Jean est malheureux ;
   donc il fait mauvais.

1. Propositions élémentaires :
   $b \equiv$ *il fait beau* ;
   $h \equiv$ *Jean est heureux.*
   Hypothèse 1 : $H_1 \equiv b \rightarrow h$ (*Quand il fait beau, Jean est heureux*)
   Hypothèse 2 : $H_2 \equiv b$ (*il fait soleil*)
   Conclusion : $C \equiv h$ (*Jean est heureux*)
   Formulation : $H_1 \wedge H_2 \rightarrow C \equiv (b \rightarrow h) \wedge b \rightarrow h$
2. Propositions élémentaires :
   $b \equiv$ *il fait beau* ;
   $h \equiv$ *Jean est heureux.*
   Hypothèse 1 : $H_1 \equiv b \rightarrow h$ (*Quand il fait beau, Jean est heureux*)
   Hypothèse 2 : $H_2 \equiv \neg b$ (*il fait mauvais*)
   Conclusion : $C \equiv \neg h$ (*Jean est malheureux*)
   Formulation : $H_1 \wedge H_2 \rightarrow C \equiv (b \rightarrow h) \wedge \neg b \rightarrow \neg h$
3. Propositions élémentaires :
   $b \equiv$ *il fait beau* ;
   $h \equiv$ *Jean est heureux.*

Hypothèse 1 : $H_1 \equiv b \rightarrow h$ (*Quand il fait beau, Jean est heureux*)
Hypothèse 2 : $H_2 \equiv \neg h$ (*Jean est malheureux*)
Conclusion : $C \equiv \neg b$ (*il fait mauvais*)
Formulation : $H_1 \wedge H_2 \rightarrow C \equiv (b \rightarrow h) \wedge \neg h \rightarrow \neg b$

Remarque : ajouter la proposition élémentaire $s \equiv$ *il fait soleil* produirait une traduction correcte mais de piètre qualité car elle aboutirait à une formule dont on ne pourrait rien tirer.

A.2 Travaux dirigés 1.3.5

Équivalence de formules

A l'aide d'une table de vérité, vérifiez les deux équivalences de Morgan.

Première équivalence : $\neg(A \vee B) \equiv \neg A \wedge \neg B$

A	B	$\neg A$	$\neg B$	$A \vee B$	$\neg(A \vee B)$	$\neg A \wedge \neg B$
$\mathbb{F}$	$\mathbb{F}$	$\mathbb{T}$	$\mathbb{T}$	$\mathbb{F}$	$\mathbb{T}$	$\mathbb{T}$
$\mathbb{F}$	$\mathbb{T}$	$\mathbb{T}$	$\mathbb{F}$	$\mathbb{T}$	$\mathbb{F}$	$\mathbb{F}$
$\mathbb{T}$	$\mathbb{F}$	$\mathbb{F}$	$\mathbb{T}$	$\mathbb{T}$	$\mathbb{F}$	$\mathbb{F}$
$\mathbb{T}$	$\mathbb{T}$	$\mathbb{F}$	$\mathbb{F}$	$\mathbb{T}$	$\mathbb{F}$	$\mathbb{F}$

Les deux formules sont équivalentes (soit toutes les deux *fausses*, soit toutes les deux *vraies*) dans toutes les interprétations (quelque soient les valuations de A et B), il s'agit donc de formules équivalentes.

Deuxième équivalence : $\neg(A \wedge B) \equiv \neg A \vee \neg B$

A	B	$\neg A$	$\neg B$	$A \wedge B$	$\neg(A \wedge B)$	$\neg A \vee \neg B$
$\mathbb{F}$	$\mathbb{F}$	$\mathbb{T}$	$\mathbb{T}$	$\mathbb{F}$	$\mathbb{T}$	$\mathbb{T}$
$\mathbb{F}$	$\mathbb{T}$	$\mathbb{T}$	$\mathbb{F}$	$\mathbb{F}$	$\mathbb{T}$	$\mathbb{T}$
$\mathbb{T}$	$\mathbb{F}$	$\mathbb{F}$	$\mathbb{T}$	$\mathbb{F}$	$\mathbb{T}$	$\mathbb{T}$
$\mathbb{T}$	$\mathbb{T}$	$\mathbb{F}$	$\mathbb{F}$	$\mathbb{T}$	$\mathbb{F}$	$\mathbb{F}$

Les deux formules sont équivalentes dans toutes les interprétations, elles sont donc équivalentes.

Valeur de formules

1. A l'aide d'une table de vérité, étudiez la validité et la consistance des formules que vous avez trouvées pour l'exercice « Formalisation d'un énoncé » du TD 1.2.2.

   Premier énoncé : $(b \rightarrow h) \wedge b \rightarrow h$

b	h	$b \rightarrow h$	$(b \rightarrow h) \wedge b$	$(b \rightarrow h) \wedge b \rightarrow h$
$\mathbb{F}$	$\mathbb{F}$	$\mathbb{T}$	$\mathbb{F}$	$\mathbb{T}$
$\mathbb{F}$	$\mathbb{T}$	$\mathbb{T}$	$\mathbb{F}$	$\mathbb{T}$
$\mathbb{T}$	$\mathbb{F}$	$\mathbb{F}$	$\mathbb{F}$	$\mathbb{T}$
$\mathbb{T}$	$\mathbb{T}$	$\mathbb{T}$	$\mathbb{T}$	$\mathbb{T}$

L'énoncé est vrai dans toutes les interprétations (quelques soient les valuations de b et h), il est donc valide et donc, à fortiori, consistant.

Deuxième énoncé : $(b \to h) \wedge \neg h \to \neg b$

b	h	$b \to h$	$\neg h$	$(b \to h) \wedge \neg h$	$\neg b$	$(b \to h) \wedge \neg h \to \neg b$
$\mathbb{F}$	$\mathbb{F}$	$\mathbb{T}$	$\mathbb{T}$	$\mathbb{T}$	$\mathbb{T}$	$\mathbb{T}$
$\mathbb{F}$	$\mathbb{T}$	$\mathbb{T}$	$\mathbb{F}$	$\mathbb{F}$	$\mathbb{T}$	$\mathbb{T}$
$\mathbb{T}$	$\mathbb{F}$	$\mathbb{F}$	$\mathbb{T}$	$\mathbb{F}$	$\mathbb{F}$	$\mathbb{T}$
$\mathbb{T}$	$\mathbb{T}$	$\mathbb{T}$	$\mathbb{F}$	$\mathbb{F}$	$\mathbb{F}$	$\mathbb{T}$

L'énoncé est vrai dans toutes les interprétations, il est donc valide et donc, à fortiori, consistant.

2. A l'aide d'une table de vérité, étudiez la validité et la consistance de la formule suivante :

$$F = \underbrace{(P \wedge Q \to \neg R)}_{F_1} \wedge \underbrace{\neg P \wedge R}_{F_2} \to \neg Q$$

Pour simplifier les écritures, nous désigneront par F_1 la formule $P \wedge Q \to \neg R$ et par F_2 la formule $\neg P \wedge R$.
L'écriture de table de vérité complète devient vite fastidieuse. Aussi est-il souvent intéressant de ne construire qu'une table de vérité partielle. Par exemple, pour la formule précédente, on peut construire une table de vérité partielle comme décrit ci-dessous.

(a) Si on regarde la table de vérité de l'implication $A \to B$ (tableau 1.3) on se rend compte que quand B est *vrai*, $A \to B$ est *vrai* quelque soit la valeur de A.
Ainsi, quand $\neg Q$ est *vrai* $\underbrace{(P \wedge Q \to \neg R) \wedge \neg P \wedge R}_{A} \to \underbrace{\neg Q}_{B}$ est également *vrai*, quelque soit la valeur de $(P \wedge Q \to \neg R) \wedge \neg P \wedge R$.
Or $\neg Q$ *vrai* est équivalent à Q *faux*. Donc, quand Q est *faux* la formule F est *vrai*.
On en déduit immédiatement les lignes *(1)*, *(2)*, *(5)* et *(6)*, soit la moitié de la table de vérité !

(b) De même, si on regarde à nouveau la table de vérité de l'implication $A \to B$ (tableau 1.3) on se rend compte que quand A est *faux*, $A \to B$ est *vrai* quelque soit la valeur de B.
Ainsi, quand $(P \wedge Q \to \neg R) \wedge \neg P \wedge R$ est *faux*, la formule $\underbrace{(P \wedge Q \to \neg R) \wedge \neg P \wedge R}_{A} \to \underbrace{\neg Q}_{B}$ est *vrai*, quelque soit la valeur de $\neg Q$.
Or $(P \wedge Q \to \neg R) \wedge \neg P \wedge R$ est la conjonction des trois formules : $P \wedge Q \to \neg R$, $\neg P$ et R. Donc $(P \wedge Q \to \neg R) \wedge \neg P \wedge R$ est *faux* dès que l'une des trois formules est *fausse*.
Ainsi, quand R est *faux*, la conjonction $(P \wedge Q \to \neg R) \wedge \neg P \wedge R$ est également *fausse* et la formule F est alors *vraie*.
On en déduit immédiatement les lignes *(3)* et *(7)*.

(c) De même, quand P est *vrai*, $\neg P$ est *faux*, ce qui entraîne que la conjonction $(P \wedge Q \to \neg R) \wedge \neg P \wedge R$ est également *fausse* et donc que la formule F est

vraie.
On en déduit immédiatement la ligne *(8)*.

(d) Il ne reste plus qu'à évaluer de manière classique la ligne *(4)* de la table de vérité.

P	Q	R	$P \wedge Q$	$\neg R$	F_1	$\neg P$	F_2	$F_1 \wedge F_2$	$\neg Q$	F	
$\mathbb{F}$	$\mathbb{F}$	$\mathbb{F}$	.....	...	..	...	..	.....	...	$\mathbb{T}$	*(1)*
$\mathbb{F}$	$\mathbb{F}$	$\mathbb{T}$	.....	...	..	...	..	.....	...	$\mathbb{T}$	*(2)*
$\mathbb{F}$	$\mathbb{T}$	$\mathbb{F}$	.....	...	..	...	..	.....	...	$\mathbb{T}$	*(3)*
$\mathbb{F}$	$\mathbb{T}$	$\mathbb{T}$	$\mathbb{F}$	$\mathbb{F}$	$\mathbb{T}$	$\mathbb{T}$	$\mathbb{T}$	$\mathbb{T}$	$\mathbb{F}$	$\mathbb{F}$	*(4)*
$\mathbb{T}$	$\mathbb{F}$	$\mathbb{F}$	.....	...	..	...	..	.....	...	$\mathbb{T}$	*(5)*
$\mathbb{T}$	$\mathbb{F}$	$\mathbb{T}$	.....	...	..	...	..	.....	...	$\mathbb{T}$	*(6)*
$\mathbb{T}$	$\mathbb{T}$	$\mathbb{F}$	.....	...	..	...	..	.....	...	$\mathbb{T}$	*(7)*
$\mathbb{T}$	$\mathbb{T}$	$\mathbb{T}$	.....	...	..	...	..	.....	...	$\mathbb{T}$	*(8)*

Traduction

Traduire en formule logique les énoncés suivants, puis étudier leur validité et leur consistance :

1. Quand on fait de l'alpinisme, on est montagnard ou sportif ;
   un montagnard est un sportif ;
   donc, quand on est pas sportif, on ne fait pas d'alpinisme.
2. Il ne dit rien s'il travaille ou s'il est seul ;
   il se repose mais il se tait ;
   donc il est seul.
3. Jean ne sort que s'il fait beau ;
   or, il pleut ;
   donc, Jean reste chez lui.

1. Propositions élémentaires :
   $A \equiv$ *Faire de l'alpinisme* ;
   $M \equiv$ *Être montagnard* ;
   $S \equiv$ *Être sportif.*
   Hypothèse 1 : $H_1 \equiv A \rightarrow M \vee S$
   Hypothèse 2 : $H_2 \equiv M \rightarrow S$
   Conclusion : $C \equiv \neg S \rightarrow \neg A$
   Formulation : $H_1 \wedge H_2 \rightarrow C \equiv (A \rightarrow M \vee S) \wedge (M \rightarrow S) \rightarrow (\neg S \rightarrow \neg A)$
   Une table de vérité permettra de montrer que cette formule est valide (et donc, à fortiori, consistante).
2. Propositions élémentaires :
   $P \equiv$ *Ne pas parler* ;
   $T \equiv$ *Travailler* ;
   $S \equiv$ *Être seul.*
   Hypothèse 1 : $H_1 \equiv T \vee S \rightarrow P$
   Hypothèse 2 : $H_2 \equiv \neg T \wedge P$
   Conclusion : $C \equiv S$
   Formulation : $H_1 \wedge H_2 \rightarrow C \equiv (T \vee S \rightarrow P) \wedge (\neg T \wedge P) \rightarrow S$

Une table de vérité permettra de montrer que cette formule est invalide mais consistante. En fait, elle n'est fausse que dans l'interprétation :

$$\{m(T) = \mathbb{F},\, m(S) = \mathbb{F},\, m(P) = \mathbb{T}\}$$

3. Propositions élémentaires :
   $S \equiv$ *Sortir* ;
   $B \equiv$ *Beau temps*.
   Hypothèse 1 : $H_1 \equiv S \rightarrow B$
   Hypothèse 2 : $H_2 \equiv \neg B$
   Conclusion : $C \equiv \neg S$
   Formulation : $H_1 \wedge H_2 \rightarrow C \equiv (S \rightarrow B) \wedge \neg B \rightarrow \neg S$
   Une table de vérité permettra de montrer que cette formule est valide (et donc, à fortiori, consistante).

Démonstration

Montrer que les deux premières équivalences de l'absorption sont justes en utilisant, entre autres, la distributivité.

- $A \vee (\neg A \wedge B) \equiv A \vee B$
  $\equiv (A \vee \neg A) \wedge (A \vee B)$
  $\equiv \mathbb{T} \wedge (A \vee B)$
  $\equiv A \vee B$
- $A \wedge (\neg A \vee B) \equiv A \wedge B$
  $\equiv (A \wedge \neg A) \vee (A \wedge B)$
  $\equiv \mathbb{F} \vee (A \wedge B)$
  $\equiv A \wedge B$

Simplification

En utilisant des équivalences de formules (propriété 1.1 page 11), tenter de trouver une forme plus simple pour les formules qui suivent. Entre parenthèses est précisée la règle d'équivalence utilisée pour la nouvelle formule. La règle de *commutativité*, qui intervient souvent, est généralement non spécifiée.

1. $(a \rightarrow a) \vee (a \rightarrow b)$
   $\equiv (\neg a \vee a) \vee (\neg a \vee b)$ *(1re de implication)*
   $\equiv \mathbb{T} \vee (\neg a \vee b)$ *(1re de complémentarité)*
   $\equiv \mathbb{T}$ *(3e de élément neutre)*
2. $a \wedge (\neg a \vee c)$
   $\equiv a \wedge c$ *(2e de absorption)*
3. $\neg a \wedge (a \rightarrow b)$
   $\equiv \neg a \wedge (\neg a \vee b)$ *(1re de implication)*
   $\equiv \neg a$ *(4e de absorption)*

4. $a \vee (\neg c \vee (b \rightarrow c))$
   $\equiv a \vee (\neg c \vee (\neg b \vee c))$ *(1re de implication)*
   $\equiv \neg c \vee c \vee a \vee \neg b$ *(1re de associativité)*
   $\equiv \mathbb{T} \vee a \vee \neg b$ *(1re de complémentarité)*
   $\equiv \mathbb{T}$ *(3e de élément neutre)*
5. $a \wedge \neg b \vee b \wedge \neg a \rightarrow a$
   $\equiv \neg((a \wedge \neg b) \vee (b \wedge \neg a)) \vee a$ *(1re de implication)*
   $\equiv (\neg(a \wedge \neg b) \wedge \neg(b \wedge \neg a)) \vee a$ *(1re de Morgan)*
   $\equiv ((\neg a \vee b) \wedge (\neg b \vee a)) \vee a$ *(2e de Morgan)*
   $\equiv (\neg a \vee b \vee a) \wedge (\neg b \vee a \vee a)$ *(1re de distributivité)*
   $\equiv (\mathbb{T} \vee b) \wedge (\neg b \vee a)$ *(1re de complémentarité et 2e de idempotence)*
   $\equiv \mathbb{T} \wedge (\neg b \vee a)$ *(3e de élément neutre)*
   $\equiv \neg b \vee a$ *(2e de élément neutre)*

A.3 Travaux dirigés 1.4.2

Algorithme de Quine

Étudiez la validité et la consistance de la formule suivante :

$$F = (P \wedge Q \rightarrow \neg R) \wedge \neg P \wedge R \rightarrow \neg Q$$

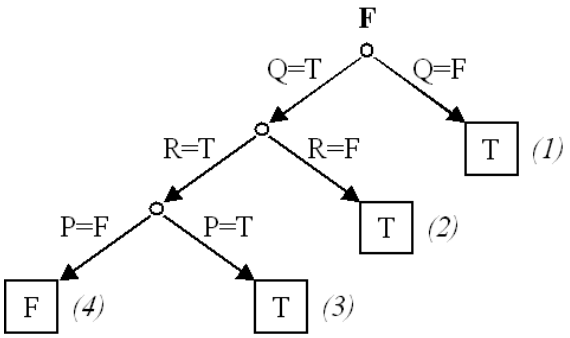

FIG. A.1 – *Algorithme de Quine*

Cet algorithme permet de construire l'arbre sémantique partiel représenté sur la figure A.1. L'explication de la construction est la même que celle de la table de vérité partielle vue dans le TD 1.3.5 page 48 pour cette même formule. La valeur *(1)* correspond à l'interprétation partielle $\{Q = \mathbb{F}\}$. La formule F est donc *vrai* pour toutes les interprétation où Q est *faux*...

Algorithme de réduction

En utilisant l'algorithme de réduction, étudiez la validité de la formule suivante :

$$F = (A \rightarrow B) \wedge (B \rightarrow C) \rightarrow (A \rightarrow C)$$

On va suppose qu'il existe une assignation de A, B et C qui rend cette formule fausse. On en déduit alors :

1. $(A \to B) \wedge (B \to C)$ est assigné à $\mathbb{T}$ et $A \to C$ est assignée à $\mathbb{F}$ (seule interprétation qui peut rendre la formule fausse) ;
2. $A \to C$ assignée à $\mathbb{F}$ implique $A = \mathbb{T}$ et $C = \mathbb{F}$;
3. $(A \to B) \wedge (B \to C)$ assigné à $\mathbb{T}$ implique $A \to B$ assigné à $\mathbb{T}$ et $B \to C$ assigné à $\mathbb{T}$;
4. $A \to B$ assigné à $\mathbb{T}$ implique $A = \mathbb{F}$ ou $B = \mathbb{T}$, or, d'après *(2)*, $A = \mathbb{T}$, donc on ne peut avoir que $B = \mathbb{T}$;
5. $B \to C$ assigné à $\mathbb{T}$ implique $B = \mathbb{F}$ ou $C = \mathbb{T}$, or $B = \mathbb{F}$ est en contradiction avec *(4)* et $C = \mathbb{T}$ est en contradiction avec *(2)*, donc la formule est valide.

A.4 Travaux dirigés 1.4.4

Formes conjonctives normales

Les formules suivantes sont-elles en forme conjonctive normale ? Combien contiennent-elles de clauses ?

1. ..... : oui, 0;
2. $\varnothing$ ..... : oui, 1 ;
3. $a \vee b$ ..... : oui, 1 ;
4. $\neg(a \vee b) \vee c$ ..... : non ;
5. $a \wedge b \wedge \neg c$ ..... : oui, 3 ;
6. $a \vee b \wedge c \vee d$ ..... : non ;
7. $(a \vee b) \wedge (\neg a \vee c) \vee d$ ..... : non ;
8. $(a \vee b) \wedge (\neg a \vee c \vee d)$ ..... : oui, 2 ;
9. $(a \vee b) \wedge (\neg a \vee c \wedge d)$ ..... : non ;
10. $(a \vee b) \wedge (\neg a \vee c) \wedge d$ ..... : oui, 3 ;
11. $(\neg a \vee b) \wedge \neg(\neg a \vee c) \wedge (d \vee e)$ ..: non ;
12. $(\neg a \vee b) \wedge a \wedge \neg c \wedge (d \vee e)$ ... : oui, 4.

Normalisation

Forme normale conjonctive des formules :

1. $p \wedge q \to p \vee q$,
   f.n.c. : $\neg p \vee \neg q \vee p \vee q$;
   f.n.c.p. sous d'ensemble de clauses : $\{\}$;
2. $p \vee q \to p \wedge q$,
   f.n.c. : $(\neg p \vee q) \wedge (\neg q \vee p)$;
   f.n.c.p. sous d'ensemble de clauses : $\{\neg p \vee q \ , \ \neg q \vee p\}$;
3. $p \leftrightarrow (p \to r)$,
   f.n.c. : $(\neg p \vee r) \wedge p$;
   f.n.c.p. sous d'ensemble de clauses : $\{\neg p \vee r\}$;

4. $p \leftrightarrow (q \rightarrow r)$,
   f.n.c. : $(\neg p \vee \neg q \vee r) \wedge (q \vee p) \wedge (\neg r \vee p)$;
   f.n.c.p. sous d'ensemble de clauses : $\{\neg p \vee \neg q \vee r\ ,\ q \vee p\ ,\ \neg r \vee p\}$;

A.5 Travaux dirigés 1.4.6

Principe de résolution

Voici le raisonnement que nous suivrons pour chacun des énoncés :

- F est valide ;
- ssi $\neg F$ est inconsistant ;
- ssi l'ensemble de clauses correspondant à la forme conjonctive normale de $\neg F$ est insatisfiable ;
- ssi on peut déduire $\varnothing$ de cet ensemble de clauses.

1. $F \equiv (A \rightarrow B) \wedge A \rightarrow B$
   $\neg F \equiv (A \rightarrow B) \wedge A \wedge \neg B$
   $S_{\neg F} = \{\underbrace{\neg A \vee B}_{(1)}, \underbrace{A}_{(2)}, \underbrace{\neg B}_{(3)}\}$
   (a) Résolvante entre (3) et (1) : $R_4 = \neg A$.
   (b) Résolvante entre R_4 et (2) : $R_5 = \varnothing$.
   F est valide.
2. $F \equiv (A \rightarrow B) \wedge \neg B \rightarrow \neg A$
   $\neg F \equiv (A \rightarrow B) \wedge \neg B \wedge A$
   $S_{\neg F} = \{\underbrace{\neg A \vee B}_{(1)}, \underbrace{A}_{(2)}, \underbrace{\neg B}_{(3)}\}$
   Même résolution que pour la formule 1. ci-dessus.
3. $F \equiv (A \rightarrow B) \wedge (B \rightarrow C) \rightarrow (A \rightarrow C)$
   Non corrigé.
4. $(A \rightarrow M \vee S) \wedge (M \rightarrow S) \rightarrow (\neg S \rightarrow \neg A)$
   Non corrigé.

Comparatif

Soit la formule suivante :

$$(p \rightarrow q) \wedge ((q \wedge \neg r) \rightarrow s) \wedge (\neg t \rightarrow (\neg r \wedge \neg s)) \rightarrow (p \rightarrow t)$$

Table de vérité : la formule contient 5 variables propositionnelles différentes (p, q, r, s, et t), la table de vérité contiendra donc $2^n = 32$ lignes. Sa réalisation est relativement fastidieuse et le risque d'erreur sur une telle table n'est pas négligeable.
Non corrigé.

Table de vérité partielle ***Non corrigé.***

Algorithme de Quine ***Non corrigé.***

Principe de résolution ***Non corrigé.***

Exercice complet

Traduire l'énoncé suivant dans le formalisme de la logique des propositions, puis étudier sa validité au moyen de l'algorithme de Quine puis du principe de résolution :

- quand un bateau fait naufrage, cela fait parler les journalistes ;
- les sponsors sont inquiets quand les journalistes parlent et qu'un bateau fait naufrage ;
- un bateau fait naufrage ;
- donc les sponsors sont inquiets.

Traduction de l'énoncé ***Non corrigé.***
Algorithme de Quine ***Non corrigé.***
Algorithme de réduction ***Non corrigé.***

A.6 Travaux dirigés 1.4.8

Principe de résolution sur des clauses de Horn

Utiliser le principe de résolution appliqué à des clauses de Horn pour étudier la validité des énoncés suivants :

1. $(A \to B) \wedge A \to B$;
2. $(A \to B) \wedge \neg B \to \neg A$
3. $(A \to B) \wedge (B \to C) \to (A \to C)$

Enquête logique

Trois personnes, *Adrien*, *Béatrice* et *Christophe*, accusées d'un meurtre, déclarent respectivement :

- Béatrice est coupable et Christophe est innocent (Adrien) ;
- si Adrien est coupable, alors Christophe l'est aussi (Béatrice) ;
- je suis innocent, mais au moins l'une des deux personnes est coupable (Christophe).

Utiliser le formalisme du calcul des propositions pour traduire les questions suivantes et donner la réponse :

1. Les trois déclarations sont-elles compatibles? ***(Non corrigé)***
2. L'un des témoignages peut-il se déduire des autres? Lequel? ***(Non corrigé)***
3. Si tous sont innocents, lequel a menti? ***(Non corrigé)***
4. Si tous disent la vérité, qui est coupable? ***(Non corrigé)***
5. Si seuls les innocents disent la vérité, qui est coupable? ***(Non corrigé)***

Bibliographie

Alliot, J.-M., & Schiex, T. (1994). *Intelligence artificielle & informatique théorique.* Cépaduès éditions.

Arcangeli. (1995-1996). *Logique formelle et calculabilité.* (Cours dispensé à l'Institut National des Sciences Appliquées de Toulouse, en seconde année de Génie Informatique et Industriel)

Hulaas, J. (2002). *Les règles grammaticales en prolog.* http://cui.unige.ch/~blond/-Langages/Prolog/Tutorial/6_grammaire.html. (Université de genève)

Nugues, P. (2000a). *Informatique linguistique et le langage prolog.* (Notes de cours, ISMRA, 2ème année de Génie Informatique)

Nugues, P. (2000b). *La programmation logique et le prolog.* (Notes de cours, ISMRA, 2ème année de Génie Informatique)

Siegel, P. (1997-1998). *Eléments de logique.* (Cours de Diplôme d'Etude Approfondie dispensé à la faculté des sciences de Luminy)