$\left(\text{Une introduction} \left| \begin{array}{c}\text{à la}\\ \text{à des}\end{array} \right.\right)$ logique(s)
à destination d'étudiants de licence informatique en troisième année

Jean Lieber, Université de Lorraine, Jean.Lieber@loria.fr

Dernière version : 31 août 2020
Page du cours : http://homepages.loria.fr/JLieber/cours/logiqueL3/logique_L3.pdf

Remarques sur ce document

— Ce document rassemble des notes de cours utiles pour les intervenants et pour les étudiants mais ne constitue pas un polycopié complet. Il ne se substitue donc pas à des prises régulières de notes durant les cours. En particulier, le programme des examens ne se limite pas au contenu de ce polycopié.

— Ce document contient probablement des erreurs (d'orthographe, de grammaire ou d'autres natures). Merci de le signaler à son auteur (par exemple, à la fin d'un cours).

Remerciements

L'auteur de ces notes de cours tient à remercier Eugeniusz-Adam Cichon pour ses conseils et les notes de cours qu'il lui a données et Dmitry Sokolov pour ses retours sur ce document et également ses notes de cours.

Sommaire

*****

1 Introduction

La logique (ou les logiques) joue(nt) un rôle important en informatique, puisqu'on la (les) trouve dans différents domaines (représentation des connaissances, bases de données, spécifications formelles, preuves de programmes, etc.). L'étymologie du terme « logiciel » en témoigne. De plus, elle constitue une base essentielle pour les fondements de la science informatique, de la même façon que l'analyse et l'algèbre linéaire (entre autres) jouent un rôle important dans les sciences physiques.

Une façon d'aborder la logique en informatique passe par la manipulation d'ensembles. En algorithmique, il est assez aisé de représenter des ensembles finis définis en extension. Or, en mathématiques, on manipule beaucoup d'ensembles potentiellement infinis et définis en compréhension (on dit aussi « en intension », avec un « s »), c'est-à-dire par un ensemble de propriétés. La logique permet de faire cela (entre autres choses).

1.1 Plan du cours

La logique propositionnelle est présentée au chapitre 2 et permet d'illustrer différentes notions générales associées à une logique (syntaxe, sémantique, inférences, etc.). En terme ensemblistes, on peut la voir comme une représentation des opérations d'intersection, d'union, de complémentaire, d'inclusion et d'égalité ensembliste ($\cap$, $\cup$, $\setminus$, $\subseteq$, $=$). On peut aussi la considérer comme la manipulation d'expressions formelles sur l'algèbre de Boole à deux éléments, même si nous éviterons de faire la fréquente confusion entre la logique propositionnelle et cette algèbre.

La logique du premier ordre (aussi appelée « calcul des prédicats ») présentée au chapitre 3 permet de manipuler des notions supplémentaires sur les ensembles, à savoir les notions de relations et de fonctions.

La conclusion de ce cours (chapitre 4) sera essentiellement l'occasion de faire une ouverture vers d'autres logiques.

1.2 Quelques références

Ces notes de cours ont été rédigées à l'aide de différents documents. Parmi ceux-ci, on peut noter les notes de cours qu'Eugeniusz-Adam Cichon a généreusement donné à l'auteur de ce document ainsi que le livre suivant, déjà ancien, mais qui s'est avéré très utile (et est loin d'être obsolète) : « Outils logiques pour l'intelligence artificielle » de Jean-Paul Delahaye (éditions Eyrolles). À noter que le propos de ce livre est bien moins sur l'IA que sur la logique (l'IA a une histoire assez récente alors que l'histoire de la logique est bien plus longue). Le cours en ligne d'Andreas Herzig (`https://www.irit.fr/~Andreas.Herzig/C/`) a également été souvent utilisé et est conseillé à tout étudiant qui souhaiterait une lecture alternative et qui ne serait pas perturbé par les quelques différences de notations avec ce cours.

Les étudiants curieux (ne le sont-ils pas tous ?) peuvent également lire les deux ouvrages suivants :

— « Logicomix » de A. K. Doxiàdis *et alii* (édition Vuibert) : une introduction en bande dessinée à une partie de l'histoire de la logique de la fin du XIX^{e} siècle au début du XX^{e}.

— « Le théorème de Gödel » (édition du Seuil) qui réunit trois articles. Le premier, de A. Nagel, K. R. Newman constitue une histoire de la logique « avant Gödel » et est facilement compréhensible. Le deuxième est l'article de K. Gödel [1] de 1931 (traduit en français). Le troisième est un article de J.-Y. Girard.

1.3 Informations pratiques

L'UE est constituée de :

— 30 heures de cours ;

— 30 heures de travaux dirigés ;

— Le nombre d'heures suffisant en travail personnel pour assimiler avec précision les notions du cours et pour préparer les séances de TD.

Les modalités de contrôle de connaissances de cette UE sont, depuis la rentrée 2020-2021, sous la forme de contrôles continus. Plus précisément, la note sera constituée d'une moyenne pondérée de trois notes (*a priori* constituées à partir de deux examens sur table et d'une note issue de TD — passages au tableau et examens surprises en TD), avec un examen de « deuxième chance » (facultatif mais qui peut relever la moyenne).

*****

2 La logique propositionnelle

L'objectif de ce chapitre est double : s'il concerne principalement la logique propositionnelle, il introduit également des notions générales sur les logiques, applicables à d'autres formalismes.

De façon générale, une logique formelle est un couple $(\mathcal{L}, \models)$ où

— $\mathcal{L}$ est un langage (un ensemble potentiellement infini de mots) défini par une syntaxe. Une ***formule*** (de cette logique) est par définition un élément de $\mathcal{L}$. Une ***base de connaissances*** est par définition un ensemble fini de formules.

— $\models$ est une ***relation d'inférences*** liant bases de connaissances et formules. Si B est une base de connaissances et φ est une formule, l'expression $B \models \varphi$ se lit « B entraîne φ » et signifie intuitivement que « Si on suppose que toutes les formules de B sont vraies, alors on peut conclure que φ est vraie. »

Ce chapitre est organisé comme suit. D'abord, quelques rappels sur les ensembles utiles pour ce chapitre sont données. Puis, un parallèle et, surtout, une distinction entre logique propositionnelle et algèbre de Boole à deux éléments sont décrits. La syntaxe et la sémantique de la logique propositionnelle $(\mathcal{LP}, \models)$ sont décrits ensuite. Une

1. Kurt Gödel (1906-1978), mathématicien austro-américain.

autre façon d'aborder la relation d'inférences s'appuie sur la notion de système formel, comme la suite du chapitre l'explique. Enfin, des aspects algorithmiques de la logique propositionnelle sont détaillés.

2.1 Premier rappel sur les ensembles

La notion même d'ensemble ne sera pas définie. On supposera que le lecteur a une intuition suffisante de ce qu'est un ensemble et de ce que signifie qu'un objet mathématique x appartient à un ensemble E (noté $x \in E$). On notera $x \notin E$ si x n'appartient pas à E.

2.1.1 Opérations de base sur les ensembles

Soit E et F, deux ensembles. E est ***inclus*** dans F, noté $E \subseteq F$ ([2]) si quel que soit $x \in E$, on a aussi $x \in F$.

Dans la suite de ce chapitre, on considèrera un ensemble U (parfois appelé l'« univers », ou l'« univers du discours ») et les ensembles considérés sont tous des sous-ensembles de U (sauf mention explicite du contraire).

On peut alors reformuler l'inclusion $E \subseteq F$ ainsi : pour tout $x \in U$, **si** $x \in E$ **alors** $x \in F$.

E et F sont ***égaux***, noté $E = F$, si $E \subseteq F$ et $F \subseteq E$.

L'***intersection*** de E et de F (notée $E \cap F$) est l'ensemble des $x \in U$ qui sont dans E et dans F : $E \cap F = \{x \in U \mid x \in E \textbf{ et } x \in F\}$.

L'***union*** de E et de F (notée $E \cup F$) est l'ensemble des $x \in U$ qui sont dans E ou dans F : $E \cup F = \{x \in U \mid x \in E \textbf{ ou } x \in F\}$.

La ***différence ensembliste*** de E et F (notée $E \setminus F$) est l'ensemble des $x \in U$ qui sont dans E mais pas dans F : $E \setminus F = \{x \in U \mid x \in E \textbf{ et } x \notin F\} = \{x \in U \mid x \in E \textbf{ et non } x \in F\}$.

Le ***complémentaire*** de E (sous-entendu « dans l'univers U ») est l'ensemble $U \setminus E$, noté $\overline{E}$: $\overline{E} = \{x \in U \mid \textbf{ non } x \in E\}$.

On notera que ces définitions font appels à des opérations relatives aux booléens (éléments de l'algèbre de Boole [3] à deux éléments) : **si** . . . **alors** . . ., **et**, **ou** et **non**.

2.1.2 Ensembles finis et infinis

On dit que deux ensembles E et F sont ***équipotents*** si on peut les mettre en bijection : il existe une fonction bijective f de E dans F. Autrement écrit, pour tout $y \in F$ il existe $x \in E$ tel que $y = f(x)$ (surjection) et s'il existe $x_1 \in E$ et $x_2 \in E$ tels que $f(x_1) = f(x_2)$ alors, on a nécessairement $x_1 = x_2$ (injection).

Un ***ensemble infini*** est un ensemble qui est équipotent à un de ses sous-ensemble strict : E est infini s'il existe F, avec $F \subseteq E$ et $F \neq E$ qui peut être en bijonction avec F.

L'ensemble des entiers naturels, $\mathbb{N}$, est infini, par exemple, il peut être mis en bijonction avec $2\mathbb{N}$, l'ensemble des entiers pairs ($f : x \in \mathbb{N} \mapsto 2x \in 2\mathbb{N}$ est bijective).

Un ensemble équipotent avec $\mathbb{N}$ est dit ***dénombrable***. Un ensemble dénombrable est donc infini. Les ensembles suivants sont dénombrables : $p\mathbb{N}$ (les multiples de p, où p est un réel non nul), $\mathbb{Z}$ (les entiers relatifs), $\mathbb{Q}$ (les rationnels), $\mathbb{A}$ (les nombres algébriques). Un ensemble ***au plus dénombrable*** est un ensemble fini ou dénombrable.

L'ensemble $\mathbb{R}$ (des nombres réels) est infini indénombrable. On peut prouver cela par la méthode de la ***diagonale de Cantor*** (du nom du mathématicien Georg Cantor), qui est une méthode ayant inspiré beaucoup de preuves de théorèmes en logique. L'idée est de prouver par l'absurde que l'intervalle $[0; 1[$ de $\mathbb{R}$ n'est pas dénombrable. On suppose donc qu'il l'est et on considère une bijection f de $\mathbb{N}$ dans $[0; 1[$. Un élément de $[0; 1[$ peut être représenté par une suite infinie de chiffre après la virgule, par exemple : $1/7 = 0{,}14285714285714285\ldots$ et donc, c'est le cas pour tout $f(n)$ où $n \in \mathbb{N}$. On note a_i^n le i^{e} chiffre après la virgule de $f(n)$. Soit à présent le nombre x dont le i^{e} chiffre après la virgule est $(a_i^i + 1) \mod 10$. On a donc : $x \neq f(n)$ pour tout $n \in \mathbb{N}$. Or, $x \in [0; 1[$ ([4]). Donc, si l'hypothèse était correcte, il y aurait $n_0 \in \mathbb{N}$ tel que $x = f(n_0)$, d'où la contradiction.

2. On choisit ici la convention anglaise pour noter l'inclusion.
3. George Boole (1815-1864), mathématicien britannique.
4. En toute rigueur, il faut aussi éliminer les séquences infinies du chiffre 9.

2.2 Algèbre de Boole à deux éléments et logique propositionnelle

L'algèbre de Boole à deux éléments évoquée plus haut (et supposée connue) permet de manipuler des éléments d'un ensemble $\mathbb{B} = \{\texttt{V}, \texttt{F}\}$ à deux éléments. `V` se lit « vrai » et `F` se lit « faux », par convention. Parmi les opérations sur cette algèbre, on note les opérations **non**, **et** et **ou** (qui se lisent comme ils s'écrivent). Pour $x, y \in \mathbb{B}$, on a l'*égalité* :

$$x \textbf{ et non } y \;=\; \textbf{non } y \textbf{ et } x$$

x **et non** y et **non** y **et** x *sont* des éléments de $\mathbb{B}$ et ils coïncident (soit tous les deux valent `F` soit tous les deux valent `V`, cela dépend des valeurs de x et de y).

En logique propositionnelle, à la place des opérations, on a des *connecteurs*, notamment $\neg$, $\wedge$ et $\vee$ qui se lisent « non », « et » et « ou » : à l'oral, la confusion est possible avec les opérations. Considérons les deux formules de la logique propositionnelle : $a \wedge \neg b$ et $\neg b \wedge a$. Une formule est une simple expression, une chaîne de caractères. Ces deux formules sont donc différentes :

$$a \wedge \neg b \;\neq\; \neg b \wedge a$$

On peut faire le parallèle avec les deux chaînes de caractères : `"a ou non b"` et `"non b ou a"`, qui sont bien deux chaînes différentes.

En revanche, on peut interpréter ces deux formules. Par exemple, interpréter a par « Marcel est un dauphin. » et b par « Marcel chante juste. » Dans le cadre de cette interprétation, les deux formules ont la même signification : Marcel est un dauphin qui ne chante pas juste. En fait, on peut montrer que quelle que soit l'interprétation, la signification des deux formules est la même, ce qu'on écrit :

$$a \wedge \neg b \;\equiv\; \neg b \wedge a$$

En logique propositionnelle, on séparera donc la syntaxe (qui décrit la forme que prend une formule) de la sémantique (liée aux interprétations d'une formule).

2.3 Syntaxe de $(\mathcal{LP}, \models)$

L'expression suivante est une formule de $\mathcal{LP}$:

$$((\texttt{mammifère} \wedge \texttt{ovipare}) \Rightarrow (\texttt{ornithorynque} \vee \texttt{échidné}))$$

Cette expression se construit avec des ***variables propositionnelles*** (`mammifère`, `ovipare`, etc.), des ***connecteurs logiques*** ($\wedge$, $\vee$, $\Rightarrow$ dans cette expression, mais il y en a d'autres) et des parenthèses.

De façon générale, la syntaxe d'une formule propositionnelle est décrite par la grammaire suivante :

<formule> ::= <variable propositionnelle> | ¬(<formule>)
| (<formule><connecteur binaire><formule>)
<connecteur binaire> ::= ∧ | ∨ | ⇒ | ⇔

De façon générale, on peut voir une expression propositionnelle comme un arbre dont les feuilles contiennent des variables propositionnelles et les nœuds non feuilles contiennent les connecteurs. Ainsi, une ***sous-formule*** d'une formule φ est une formule correspondant à un sous-arbre de l'arbre de φ.

On omettra les parenthèses quand elles ne sont pas utiles. On peut enlever les parenthèses extérieures : $(a \wedge b)$ peut s'écrire $a \wedge b$. Par ailleurs, on considère l'ordre de priorité suivant : $\neg$ est le plus prioritaire, puis viennent (à égalité) $\wedge$ et $\vee$ et enfin (à égalité) $\Rightarrow$ et $\Leftrightarrow$. Par exemple,

$$\neg a \wedge b \Rightarrow \neg(c \Rightarrow d \vee e) \qquad \text{se lit} \qquad ((\neg a) \wedge b) \Rightarrow (\neg(c \Rightarrow (d \vee e)))$$

La formule $a \wedge b \vee c$ est syntaxiquement incorrecte : on ne sait pas s'il faut la lire $(a \wedge b) \vee c$ ou $a \wedge (b \vee c)$. De même pour la formule $a \wedge b \wedge c$, mais nous verrons plus loin que cette dernière écriture est acceptée car son ambiguïté est uniquement syntaxique, pas sémantique. En effet, on verra que $a \wedge (b \wedge c) \equiv (a \wedge b) \wedge c$.

La formule $\varphi_1 \wedge \varphi_2 \wedge \ldots \wedge \varphi_n$ est la ***conjonction*** de φ_1, φ_2, ...et φ_n. La formule $\varphi_1 \vee \varphi_2 \vee \ldots \vee \varphi_n$ est la ***disjonction*** de φ_1, φ_2, ...et φ_n.

De plus, l'ensemble des connecteurs binaires peut être enrichi ou appauvri sans que cela ne change l'expressivité du langage (cela sera envisagé plus loin aussi). Le seul connecteur logique unaire est $\neg$. Pour lire une formule, on s'appuie sur ceci :

le connecteur	$\neg$	$\wedge$	$\vee$	$\Rightarrow$	$\Leftrightarrow$
se lit	« non »	« et »	« ou »	« implique »	« équivaut à »

Chez certains auteurs, $\Rightarrow$ s'écrit $\rightarrow$ et $\Leftrightarrow$ s'écrit $\leftrightarrow$.

L'ensemble des variables propositionnelles se note $\mathcal{V}$. C'est un ensemble fini ou dénombrable. Dans le cadre de ce cours, on supposera généralement qu'il est fini (on parle alors de logique propositionnelle finie). En pratique, on définira rarement $\mathcal{V}$ et on considérera que c'est l'ensemble des variables propositionnelles qui apparaissent dans l'ensemble des formules utilisées dans un énoncé.

Sous-formules d'une formule. On peut assimiler une formule propositionnelle à un arbre dont la racine est le connecteur le plus « extérieur » et ses fils sont le(s) membre(s) du connecteur. Fort de cette représentation, on peut définir une sous-formule d'une formule comme étant un de ses sous-arbres.

Plus formellement, pour $\varphi \in \mathcal{LP}$:

— φ est une sous-formule de φ ;

— Si $\varphi = \neg\varphi_1$ alors toutes les sous-formules de φ_1 (y compris φ_1, évidemment) sont des sous-formules de φ ;

— Si $\varphi = \varphi_1 * \varphi_2$ où $*$ est n'importe quel connecteur binaire, alors les sous-formules de φ_1 et celles de φ_2 sont les sous-formules de φ.

Par exemple, les sous-formules de $\neg(a \Rightarrow (b \vee c))$ sont

$$\neg(a \Rightarrow (b \vee c)) \qquad a \Rightarrow (b \vee c) \qquad a \qquad b \vee c \qquad b \qquad \text{et} \qquad c$$

2.4 Sémantique de $(\mathcal{LP}, \models)$

2.4.1 La sémantique en théorie des modèles

Une approche pour définir $\models$ dans de nombreux formalismes s'appuie sur la ***sémantique en théorie des modèles*** (ou de Tarski). Étant donné un langage $\mathcal{L}$, on peut définir une relation d'inférence $\models$ en suivant la démarche suivante (ci-dessous, $\mathcal{I}$ est une interprétation, φ, φ_1 et φ_2 sont des formules, B est une base de connaissances) :

1. On définit une notion d'***interprétation*** (dont la nature dépend de la logique en cours de définition).
2. On définit la relation « ***satisfait*** » entre $\mathcal{I}$ et φ (cette relation dépend également de la logique en cours de définition). On note cette relation par le symbole $\models$: $\mathcal{I} \models \varphi$ se lit « $\mathcal{I}$ satisfait φ ». Notons que le symbole $\models$ est utilisé pour plusieurs relations (le contexte permet de lever l'ambiguïté).
3. On étend cela aux bases de connaissances : $\mathcal{I}$ satisfait B (noté $\mathcal{I} \models B$) si $\mathcal{I} \models \varphi$ pour toute $\varphi \in B$.
4. Un ***modèle*** de φ (resp., de B) est une interprétation $\mathcal{I}$ qui satisfait φ (resp., B).
5. B entraîne φ (noté $B \models \varphi$ ou $\models_B \varphi$) si tout modèle de B est un modèle de φ.
6. φ (resp., B) est satisfiable s'il existe un modèle de φ (resp., de B). Dans le cas contraire on dira que φ (ou B) est insatisfiable.
7. Dans le cadre des logiques de ce cours, les termes « insatisfiable », « incohérent », « contradictoire » et « inconsistent » sont équivalents. De même que les termes « satisfiable », « cohérent », « non contradictoire » et « consistent ».
8. $\varphi_1 \models \varphi_2$ si $\{\varphi_1\} \models \varphi_2$.
9. $\varphi_1 \equiv \varphi_2$ si $\varphi_1 \models \varphi_2$ et $\varphi_2 \models \varphi_1$.
   $\varphi_1 \equiv \varphi_2$ signifie donc que $\mathcal{I}$ est un modèle de φ_1 ssi $\mathcal{I}$ est un modèle de φ_2.

10. φ est une ***tautologie*** (noté $\models \varphi$) si toute interprétation est un modèle de φ.
11. $\varphi \models_B \psi$ si $B \cup \{\varphi\} \models \psi$.

Remarque 1 Les points 1 et 2 ci-dessus sont dépendants de la logique dont on est en train de définir la sémantique. Les autres points sont indépendants. Autrement écrit, si on a définit ce qu'est une interprétation et la relation « satisfait » entre une interprétation et une formule, le reste en découle.

Remarque 2 Les symboles $\models$ et $\equiv$ ne font *pas* partie du langage $\mathcal{L}$. Ce sont des symboles du « méta-langage ».

2.4.2 Sémantique en théorie des modèles pour $(\mathcal{LP}, \models)$

On se donne un ensemble à deux éléments, $\mathbb{B} = \{\mathtt{V}, \mathtt{F}\}$. $\mathtt{V}$ se lit « vrai » et $\mathtt{F}$ se lit « faux », par convention.

Une ***interprétation*** $\mathcal{I}$ est une application de $\mathcal{V}$ dans $\mathbb{B}$. Si $x \in \mathcal{V}$, $\mathcal{I}(x)$ sera noté $x^{\mathcal{I}}$ dans ce cours. Par exemple, si $\mathcal{V} = \{a, b, c\}$, $\mathcal{I}$ définie par $a^{\mathcal{I}} = \mathtt{F}$, $b^{\mathcal{I}} = \mathtt{V}$ et $c^{\mathcal{I}} = \mathtt{F}$ est une interprétation. On notera cette interprétation par $\bar{a}b\bar{c}$ et, plus généralement, on note une interprétation par une séquence de symboles x ou $\bar{x}$ pour $x \in \mathcal{V}$, sachant que chaque variable doit apparaître une fois et une seule dans cette séquence et que si x apparaît sans barre, alors $x^{\mathcal{I}} = \mathtt{V}$ et sinon, $x^{\mathcal{I}} = \mathtt{F}$.

Question 1 *Si $\mathcal{V}$ est fini et contient n variables, combien y a-t-il d'interprétations ?*

Soit $\mathcal{I}$ une interprétation et φ une formule. Le fait que $\mathcal{I}$ ***satisfasse*** φ peut être défini intuitivement par le fait de remplacer chaque variable x par la valeur booléenne $x^{\mathcal{I}}$, chaque connecteur par un opérateur sur les booléens ($\neg$ par **non**, $\wedge$ par **et**, $\vee$ par **ou**, etc.) et à évaluer l'expression booléenne, d'où le résultat $\varphi^{\mathcal{I}} \in \mathbb{B}$. Par exemple,

$$\text{Si } \mathcal{I} = \bar{a}b\bar{c}\bar{d} \text{ et } \varphi = (a \wedge \neg b) \vee \neg(c \vee \neg\neg d) \text{ alors } \varphi^{\mathcal{I}} = (\mathtt{V} \textbf{ et non } \mathtt{F}) \textbf{ ou non}(\mathtt{V} \textbf{ ou non non } \mathtt{F}) = \mathtt{V}$$

Plus précisément, la fonction $\mathcal{I}$ est étendue de $\mathcal{V}$ vers $\mathcal{LP}$ de la façon suivante ($\varphi, \varphi_1, \varphi_2 \in \mathcal{LP}$) :

$$(\neg\varphi)^{\mathcal{I}} = \begin{cases} \mathtt{F} & \text{si } \varphi^{\mathcal{I}} = \mathtt{V} \\ \mathtt{V} & \text{sinon} \end{cases} \qquad (\varphi_1 \wedge \varphi_2)^{\mathcal{I}} = \begin{cases} \mathtt{V} & \text{si } \varphi_1^{\mathcal{I}} = \varphi_2^{\mathcal{I}} = \mathtt{V} \\ \mathtt{F} & \text{sinon} \end{cases} \qquad (\varphi_1 \vee \varphi_2)^{\mathcal{I}} = \begin{cases} \mathtt{F} & \text{si } \varphi_1^{\mathcal{I}} = \varphi_2^{\mathcal{I}} = \mathtt{F} \\ \mathtt{V} & \text{sinon} \end{cases}$$

$$(\varphi_1 \Rightarrow \varphi_2)^{\mathcal{I}} = \begin{cases} \mathtt{V} & \text{si } \varphi_1^{\mathcal{I}} = \mathtt{F} \text{ ou } \varphi_2^{\mathcal{I}} = \mathtt{V} \\ \mathtt{F} & \text{sinon} \end{cases} \qquad (\varphi_1 \Leftrightarrow \varphi_2)^{\mathcal{I}} = \begin{cases} \mathtt{V} & \text{si } \varphi_1^{\mathcal{I}} = \varphi_2^{\mathcal{I}} \\ \mathtt{F} & \text{sinon} \end{cases}$$

De cela découle la relation $\models$ liant une base de connaissances et une formule, les notions de satisfiabilité, de tautologie, etc.

Pour tester si une formule est satisfiable, qu'elle est une tautologie, qu'une base de connaissances entraîne une formule, une façon de faire (peu efficace mais simple à comprendre) consiste à parcourir l'ensemble des interprétations. C'est ce qu'on appelle la méthode des tables de vérité.

Soit Ω l'ensemble des interprétations associées aux variables de $\mathcal{V}$. Étant donné $\varphi \in \mathcal{LP}$, on note $\mathcal{M}(\varphi)$ l'ensemble des modèles de φ. De la même façon, $\mathcal{M}(B)$ est l'ensemble des modèles de la base de connaissances B. On a donc, en particulier, $B \models \varphi$ ssi $\mathcal{M}(B) \subseteq \mathcal{M}(\varphi)$.

Par ailleurs, on note $\mathtt{Cn}(\varphi)$ l'ensemble des conséquences logiques de φ : $\mathtt{Cn}(\varphi) = \{\psi \in \mathcal{LP} \mid \varphi \models \psi\}$. De la même façon, $\mathtt{Cn}(B)$ est l'ensemble des conséquences de B. On a donc, en particulier, $B \models \varphi$ ssi $\mathtt{Cn}(B) \supseteq \mathtt{Cn}(\varphi)$.

Étant donné une formule φ, $\mathcal{M}(\varphi)$ peut se comprendre comme l'ensemble des instanciations de φ et $\mathtt{Cn}(\varphi)$ comme l'ensemble des propriétés de φ. La figure 1 montre un parallèle avec la programmation orientée objets. Notons que ce parallèle ne tient pas compte des mécanismes d'exception à l'héritage. Pour tenir compte de ceux-ci, d'autres logiques doivent être définies (cf. en conclusion les logiques non monotones).

Notions relatives à la logique propositionnelle	Notions relatives à la programmation par objets
formules propositionnelles φ et ψ telles que $\varphi \models \psi$	classes C et D telles que C est une sous-classe de D
modèle $\mathcal{I}$ d'une formule	instance (directe ou indirecte) d'une classe
Si $\mathcal{I}$ est un modèle de φ alors $\mathcal{I}$ est un modèle de ψ.	Si a est une instance de C alors a est une instance de D.
propriété d'une formule $\chi : \pi \in \texttt{Cn}(\chi)$	attribut ou méthode d'une classe
Si π est une propriété de ψ alors π est une propriété de φ.	Si p est un attribut (resp., une méthode) de D alors p est un attribut (resp., une méthode) de C (mécanisme d'héritage)

FIGURE 1 – Un parallèle entre la logique propositionnelle et la programmation orientée objets.

Exercice 1 *Soit $\varphi = a \wedge \neg(b \Rightarrow (c \vee \neg a))$. Est-ce une tautologie ? Est-elle satisfiable ? Si oui, donnez une interprétation de φ. Si non, montrer qu'elle n'est pas satisfiable en utilisant une table de vérité.*

Exercice 2 *Mêmes questions pour $\varphi = (a \vee \neg(b \vee a)) \vee b$.*

Exercice 3 *Montrez que $a \Rightarrow b \models \neg b \Rightarrow \neg a$.*

Exercice 4 *Montrez que $(a \wedge b) \vee c \equiv (a \vee c) \wedge (b \vee c)$.*

2.4.3 Quelques résultats utiles

Résultats généraux.

Théorème de la déduction. Le résultat suivant est appelé théorème de la déduction ($\varphi_1, \varphi_2 \in \mathcal{LP}$) :

$$\varphi_1 \models \varphi_2 \qquad \text{ssi} \qquad \models \varphi_1 \Rightarrow \varphi_2$$

Question 2 *Quelles sont les formules de $\mathcal{LP}$ qui interviennent ? Quels sont les symboles du langage et quels sont les symboles du méta-langage ? Comment lire ce résultat en utilisant les termes introduits plus haut (tautologie, etc.) ?*

Un corollaire du théorème de la déduction est ($\varphi_1, \varphi_2 \in \mathcal{LP}$) :

$$\varphi_1 \equiv \varphi_2 \qquad \text{ssi} \qquad \models \varphi_1 \Leftrightarrow \varphi_2$$

Lien entre tautologies et formules satisfiables. Le résultat suivant lie les notions de tautologie et de satisfiabilité (pour $\varphi \in \mathcal{LP}$) :

$$\varphi \text{ est une tautologie} \qquad \text{ssi} \qquad \neg\varphi \text{ est insatisfiable}$$

Un corollaire de ce résultat est le suivant (pour $\varphi \in \mathcal{LP}$) :

$$\varphi \text{ est une formule satisfiable} \qquad \text{ssi} \qquad \neg\varphi \text{ n'est pas une tautologie}$$

Assimiler une base de connaissances à la conjonction de ses formules. Soit $B = \{\varphi_1, \varphi_2, \ldots, \varphi_n\}$ une base de connaissances de $(\mathcal{LP}, \models)$. Soit $\varphi = \varphi_1 \wedge (\varphi_2 \wedge (\ldots \varphi_n))$. B et φ sont équivalents au sens où ils ont les mêmes modèles. Par conséquent, $B \models \psi$ ssi $\varphi \models \psi$. En logique propositionnelle, on peut donc ***assimiler une base de connaissances à une formule*** obtenue en faisant la conjonction des formules de cette base.

La substitution dans une tautologie est une tautologie. La ***substitution*** d'une variable propositionnelle x par une formule propositionnelle ψ dans une formule propositionnelle φ est le remplacement de toutes les occurrences de x par ψ dans φ. Le résultat est noté $\varphi[x\backslash\psi]$. Par exemple $(\neg a \Rightarrow b)[a\backslash(b \vee c)]$ est la formule $\neg(b \vee c) \Rightarrow b$. Si φ est une tautologie, alors toute substitution de variable par une formule dans φ est une tautologie.

Remplacer une sous-formule par une formule équivalente conserve l'équivalence logique. Si φ, α et β sont des formules telles que α est une sous-formule de φ et $\alpha \equiv \beta$, alors si on remplace la sous-formule α par β, la formule obtenue est équivalente à φ.

Par exemple, avec $\varphi = a \wedge (b \Rightarrow c)$, $\alpha = b \Rightarrow c$ et $\beta = \neg b \vee c$, les hypothèses sont vérifiées et donc, avec $\psi = a \wedge (\neg b \vee c)$, on a $\varphi \equiv \psi$.

Exercice 5 *Soit* `sat` *la fonction qui à $\varphi \in \mathcal{LP}$ associe* `V` *si φ est satisfiable et* `F` *sinon. En s'appuyant sur les résultats ci-dessus, expliquez comment utiliser cette fonction pour :*

— *Tester qu'une formule φ est une tautologie ;*

— *Tester que deux formules φ_1 et φ_2 sont équivalentes ;*

— *Faire le test $\varphi_1 \models \varphi_2$.*

Exercice 6 *Montrez qu'on a $a \models a$.*

Comment peut-on déduire de cela que $(b \wedge c) \models (b \wedge c)$?

Exercice 7 *Montrez qu'on a $a \wedge \neg a \models b \Rightarrow c$.*

De façon plus générale, donnez une condition nécessaire et suffisante portant sur une formule φ pour que, quelle que soit ψ, on ait $\varphi \models \psi$.

Exercice 8 *On a vu que toute substitution d'une variable par une formule dans une tautologie donne une tautologie. En revanche, quand on substitue une variable par une formule dans une formule satisfiable, le résultat n'est pas nécessairement satisfiable.*

Donnez un exemple montrant cela.

Que se passe-t-il quand on remplace une variable par une formule dans une formule insatisfiable ? Justifiez votre réponse.

Quelques équivalences. Soit φ, ψ et χ trois formules propositionnelles. On a les équivalences suivantes :

$$\varphi \wedge \psi \equiv \psi \wedge \varphi \quad (1)$$

$$\varphi \vee \psi \equiv \psi \vee \varphi \quad (2)$$

$$\varphi \wedge (\psi \wedge \chi) \equiv (\varphi \wedge \psi) \wedge \chi \quad (3)$$

$$\varphi \vee (\psi \vee \chi) \equiv (\varphi \vee \psi) \vee \chi \quad (4)$$

Les équivalences (1) et (2) peuvent être qualifiées de « commutativité modulo l'équivalence ». En particulier, les formules $a \wedge b$ et $b \wedge a$ sont *différentes* mais représentent la même chose, plus précisément : le même ensemble de modèles, puisque $\mathcal{M}(a \wedge b) = \mathcal{M}(b \wedge a)$.

De la même façon, les équivalences (3) et (4) indiquent une associativité modulo l'équivalence. Ces équivalences permettent d'oublier certaines parenthèses. Par exemple, $(a \vee \neg b) \vee \neg c$ peut s'écrire $a \vee \neg b \vee \not{c}$: même si on parenthèse autrement — i.e., $a \vee (\neg b \vee \neg c)$ — le résultat quoique syntaxiquement différent sera sémantiquement équivalent.

L'associativité et la commutativité permettent aussi d'écrire des expressions de la forme $\bigwedge_{i=1}^{n} \varphi_i$ ou de la forme $\bigvee_{i=1}^{n} \varphi_i$: quel que soit l'ordre et le parenthésage des φ_i le résultat sera équivalent.

Les deux équivalences suivantes indiquent que $\wedge$ est distributif sur $\vee$ et inversement, modulo l'équivalence logique :

$$\varphi \wedge (\psi \vee \chi) \quad \equiv \quad (\varphi \wedge \psi) \vee (\varphi \wedge \chi) \tag{5}$$
$$\varphi \vee (\psi \wedge \chi) \quad \equiv \quad (\varphi \vee \psi) \wedge (\varphi \vee \chi) \tag{6}$$

Les équivalences qui suivent concernent la négation logique ($\neg$) :

$$\neg\neg\varphi \quad \equiv \quad \varphi \tag{7}$$
$$\neg(\varphi \wedge \psi) \quad \equiv \quad \neg\varphi \vee \neg\psi \tag{8}$$
$$\neg(\varphi \vee \psi) \quad \equiv \quad \neg\varphi \wedge \neg\psi \tag{9}$$

(7) indique que la négation logique est involutive modulo l'équivalence. Les équivalences (8) et (9) sont appelées lois de De Morgan [5], lois qu'on retrouve dans d'autres domaines, en particulier dans les algèbres de Boole.

L'équivalence suivante est liée à la contraposition :

$$\varphi \Rightarrow \psi \quad \equiv \quad \neg\psi \Rightarrow \neg\varphi \tag{10}$$

Elle est utile pour faire un raisonnement par l'absurde.

Changer d'ensemble de connecteurs sans changer l'expressivité. Les équivalences qui suivent montrent qu'on peut, théoriquement, se passer des connecteurs $\Rightarrow$ et $\Leftrightarrow$:

$$\varphi \Rightarrow \psi \quad \equiv \quad \neg\varphi \vee \psi \tag{11}$$
$$\varphi \Leftrightarrow \psi \quad \equiv \quad (\varphi \Rightarrow \psi) \wedge (\psi \Rightarrow \varphi) \tag{12}$$

En effet, si on prend une formule propositionnelle quelconque et qu'on applique les remplacements suggérés par ces équivalences de gauche à droite tant que c'est possible, on arrivera systématiquement à une formule sans ces deux connecteurs. Par exemple :

$$a \Rightarrow (b \Leftrightarrow c) \xrightarrow{(11)} \neg a \vee (b \Leftrightarrow c) \xrightarrow{(12)} \neg a \vee ((b \Rightarrow c) \wedge (c \Rightarrow b))$$
$$\xrightarrow{(11)} \neg a \vee ((b \Rightarrow c) \wedge (\neg c \vee b)) \xrightarrow{(11)} \neg a \vee ((\neg b \vee c) \wedge (\neg c \vee b))$$

On notera qu'il y a plusieurs façons de faire (selon l'ordre d'application de ces transformations) mais qui mèneront toutes à une formule équivalente à la formule initiale.

On peut aussi se passer de $\vee$ en appliquant l'équivalence suivante :

$$\varphi \vee \psi \quad \equiv \quad \neg(\neg\varphi \wedge \neg\psi)$$

Par conséquent, on peut se contenter de l'ensemble de connecteurs $\{\neg, \wedge\}$ sans que cela nuise à l'expressivité du langage. Dans ce cas, on peut considérer les autres connecteurs comme des raccourcis (ou des « macros »), p. ex., $\varphi \vee \psi$ est un raccourci pour $\neg(\neg\varphi \wedge \neg\psi)$.

On peut aussi considérer d'autres connecteurs, dont voici un échantillon :

— Le connecteur d'arité 0 noté $\bot$ interprété comme suit : $\bot^{\mathcal{I}} = \mathtt{F}$ quelle que soit l'interprétation $\mathcal{I}$;

— Le connecteur d'arité 0 noté $\top$ interprété comme suit : $\top^{\mathcal{I}} = \mathtt{V}$ quelle que soit l'interprétation $\mathcal{I}$;

— Le connecteur binaire $\uparrow$ (« non et » ou « nand ») tel que $(\varphi \uparrow \psi)^{\mathcal{I}} = \mathtt{F}$ ssi $\varphi^{\mathcal{I}} = \psi^{\mathcal{I}} = \mathtt{V}$;

5. Augustus de Morgan (1806-1871), mathématicien britannique.

— Le connecteur binaire $\downarrow$ (« non ou » ou « nor ») tel que $(\varphi \downarrow \psi)^{\mathcal{I}} = \mathtt{V}$ ssi $\varphi^{\mathcal{I}} = \psi^{\mathcal{I}} = \mathtt{F}$;
— Le connecteur binaire $\oplus$ (« ou exclusif » ou « xor ») tel que $(\varphi \oplus \psi)^{\mathcal{I}} = \mathtt{V}$ ssi $\varphi^{\mathcal{I}} \neq \psi^{\mathcal{I}}$.

Exercice 9 *Montrez comment chacun de ces nouveaux connecteurs peut être remplacé par les connecteurs de l'ensemble donné initialement ($\{\neg, \wedge, \vee, \Rightarrow, \Leftrightarrow\}$).*

À titre d'exemple, $\bot$ peut être remplacé par $(a \wedge \neg a)$ (où a est une variable propositionnelle quelconque).

Exercice 10 *On considère les ensembles de connecteurs suivants : $\{\neg, \vee\}$, $\{\neg, \Rightarrow\}$, $\{\uparrow\}$.*

Pour chacun de ces ensembles de connecteurs, montrer qu'on peut se contenter de formules n'utilisant que ces connecteurs sans perdre l'expressivité de la logique propositionnelle.

Quelques conséquences logiques. Étant donné $\varphi, \psi, \chi \in \mathcal{LP}$, on a les conséquences logiques suivantes :

$$\varphi \wedge \psi \quad \models \quad \varphi \tag{13}$$
$$\varphi \quad \models \quad \varphi \vee \psi \tag{14}$$
$$\varphi \wedge (\varphi \Rightarrow \psi) \quad \models \quad \psi \tag{15}$$

Par ailleurs, on rappelle que $\alpha \equiv \beta$ ssi $\alpha \models \beta$ et $\beta \models \alpha$.

Comment montrer ces équivalences et ces conséquences logiques ? À titre d'exemple, considérons l'équivalence (9) : $\neg(\varphi \vee \psi) \equiv \neg\varphi \wedge \neg\psi$. Il faut la prouver pour toute formule φ et toute formule ψ. En utilisant le corollaire du théorème de la déduction, cela revient à prouver que la formule $\neg(\varphi \vee \psi) \Leftrightarrow \neg\varphi \wedge \neg\psi$ est une tautologie.

Dans un premier temps, on va considérer le cas particulier où φ et ψ sont des variables propositionnelles (puisque $\mathcal{V} \subseteq \mathcal{LP}$). On notera $a = \varphi$ et $b = \psi$. Pour prouver que $\Phi = \neg(a \vee b) \Leftrightarrow \neg a \wedge \neg b$ est une tautologie, on peut remplir une table de vérité (ou la prouver d'une autre manière) :

	a	b	$a \vee b$	$\neg(a \vee b)$	$\neg a \wedge \neg b$	$\Phi = \neg(a \vee b) \Leftrightarrow \neg a \wedge \neg b$
$\mathcal{I}_1$	F	F	F	V	V	V
$\mathcal{I}_2$	F	V	V	F	F	V
$\mathcal{I}_3$	V	F	V	F	F	V
$\mathcal{I}_4$	V	V	V	F	F	V

Toutes les interprétations ($\mathcal{I}_1$ à $\mathcal{I}_4$) satisfont Φ, c'est donc bien une tautologie.

La formule $\neg(\varphi \vee \psi) \Leftrightarrow \neg\varphi \wedge \neg\psi$ est le résultat de la substitution dans Φ de a par φ et de b par ψ : $\Phi[a \backslash \varphi][b \backslash \psi]$. Or, on a vu que quand on substituait une variable d'une tautologie par une formule, le résultat était toujours une tautologie. Par conséquent, $\neg(\varphi \vee \psi) \Leftrightarrow \neg\varphi \wedge \neg\psi$ est bien une tautologie et on a montré l'équivalence (9).

Exercice 11 *Choisissez au hasard au moins 4 équivalences ou implications à montrer.*

En appliquant le même principe que dans l'exemple ci-dessus, montrez ces équivalences.

Le principe de l'indépendance à la syntaxe. De façon générale, un mécanisme inférentiel, que ce soit une déduction (comme $\models$) ou une inférence non déductive (ou hypothétique) associe à des formules données comme prémisses des formules données comme conclusions. Par exemple, de la prémisse $a \wedge b$ on peut conclure b selon la relation d'inférence $\models$.

Un tel mécanisme inférentiel se fait sur des formules, éléments de syntaxe, mais selon une interprétation sémantique. Par exemple, supposer que $a \wedge b$ est vrai ou supposer que $b \wedge a \wedge a$ est vrai c'est la même chose et donc, partant de l'une ou l'autre prémisse, on doit arriver aux mêmes conclusions.

Le principe de l'indépendance à la syntaxe d'une inférence indique que si on a une inférence partant de prémisses φ_1, ..., φ_n et arrivant à une conclusion χ alors, si on remplace les φ_i par des formules équivalentes φ'_i (i.e., $\varphi_i \equiv \varphi'_i$), on doit pouvoir inférer χ' équivalent à χ. Pour la relation inférentielle $\models$ considérée entre deux formules, cela s'écrit comme suit :

$$\text{Si } \varphi \models \chi \text{ et } \varphi \equiv \varphi' \text{ alors } \varphi' \models \chi$$

(on a sauté une étape ici : $\varphi' \models \chi'$ avec $\chi' \equiv \chi$, d'où la conclusion).

En revanche si on a une inférence qui compte le nombre d'occurrences d'une variable dans une formule, cette inférence ne respecte pas l'indépendance à la syntaxe et ne devrait pas être considérée comme une inférence au sein de la logique (c'est bien une inférence, mais elle se situe en dehors du cadre formel de cette logique).

2.5 Un système formel correct et complet pour $(\mathcal{LP}, \models)$

La sémantique en théorie des modèles fait appel à des valeurs de vérité (`V` et `F`). Une autre façon de définir une relation d'inférence passe par la notion de système formel et de preuve : l'idée générale est la *construction* de formules à partir de formules données au départ (axiomes et hypothèses) en s'appuyant sur des règles d'inférence.

2.5.1 Notion de système formel

Pour définir un système formel S, il faut définir un langage $\mathcal{L}$, un ensemble d'axiomes *Ax* et un ensemble fini *RI* de règles d'inférence : $S = (\mathcal{L}, Ax, RI)$.

On considérera l'exemple du système formel $S^+ = (\mathcal{L}^+, Ax^+, RI^+)$ qui définit la notion d'addition.

Un élément de $\mathcal{L}$ est appelé une « formule bien formée » ou ***fbf*** (à ne pas confondre *a priori* avec la notion de formule, même si les notions vont coïncider dans la suite du cours). Par exemple, on se donne la grammaire suivante pour $\mathcal{L}^+$:

```
<fbf> ::= <entier> + <entier> = <entier>
<entier> ::= 0 | S<entier>
```

Ainsi, $\mathcal{L}^+$ représente les équations x + y = z où x, y et z sont des entiers naturels représentés sous la forme des entiers bâtons (`0`, `S0`, `SS0`, `SSS0`, etc.).

L'ensemble des ***axiomes***, *Ax*, est un sous-ensemble de $\mathcal{L}$: ce sont des fbf qu'on se donne comme points de départ, les éléments de base de la construction. Souvent, l'ensemble des axiomes est infini et est défini par un ensemble de ***schémas d'axiomes***. Par exemple, Ax^+ est défini par un seul schéma d'axiomes :

$$\texttt{0} + x = x \qquad \text{où } x \text{ est de type } \texttt{<entier>}$$

Les axiomes sont donc les fbf `0 + 0 = 0`, `0 + S0 = S0`, `0 + SS0 = SS0`, `0 + SSS0 = SSS0`, etc.

Les ***règles d'inférence*** décrivent comment des fbf (appelées ***conclusions***) peuvent être produites en partant d'autres fbf (appelées ***prémisses***). Une règle d'inférence s'écrit en général sous la forme

$$\frac{\text{schémas de fbf décrivant les prémisses}}{\text{schémas de fbf décrivant les conclusions}}\ \text{nom de la règle}$$

Par exemple, $RI^+ = \{r\}$ avec

$$\frac{x + y = z}{\texttt{S}x + y = \texttt{S}z}\, r \qquad \text{où } x, y \text{ et } z \text{ sont de type } \texttt{<entier>}$$

Cette règle a une seule prémisse et une seule conclusion.

Étant donné un système formel S, on peut définir la notion de ***preuve*** dans ce système. On considère un ensemble fini *Hyp* de fbf, l'ensemble des *hypothèses*. Soit f, une fbf. Une preuve de f dans le système formel S est une séquence finie de fbf $f_1, f_2, \ldots, f_n$ telle que :

— $f_n = f$ (la dernière fbf de la séquence est la fbf à prouver) ;
— Pour tout f_i $(1 \leqslant i \leqslant n)$ f_i est
  — Soit un axiome ($f_i \in Ax$),
  — Soit une hypothèse ($f_i \in Hyp$),
  — Soit résulte de l'application d'une règle d'inférence $r \in RI$ sur une ou des formule(s) f_k avec $k < i$.

Par exemple, dans S^+, considérons $Hyp = \{\texttt{SS0 + S0 = SSS0}\}$, alors on a la preuve f_1, f_2, f_3 de `SSSS0 + S0 = SSSSS0` suivante :

Nom de la fbf	fbf	explications
f_1	`SS0 + S0 = SSS0`	hypothèse ($f_1 \in Hyp$)
f_2	`SSS0 + S0 = SSSS0`	application de r sur f_1 ($x =$ `SS0`, $y =$ `S0`)
f_3	`SSSS0 + S0 = SSSSS0`	application de r sur f_2 ($x =$ `SSS0`, $y =$ `S0`)

S'il existe une preuve de la fbf f à partir d'un ensemble d'hypothèses *Hyp* dans le système formel S, alors on notera $Hyp \vdash^S f$. Par exemple, $\{\texttt{SS0 + S0 = SSS0}\} \vdash^{S^+} \texttt{SSSS0 + S0 = SSSSS0}$.

Un ***théorème*** de S est une fbf qui peut être prouvée à partir d'un ensemble d'hypothèses vide. Autrement écrit, f est un théorème de S si $\emptyset \vdash^S f$. On écrira simplement cela : $\vdash^S f$, voire $\vdash f$ quand il n'y a pas d'ambiguïté. Par exemple, $\vdash^{S^+}$ `SS0 + SS0 = SSSS0` :

Nom de la fbf	fbf	explications
f_1	`0 + SS0 = SS0`	axiome ($f_1 \in Ax$)
f_2	`S0 + SS0 = SSS0`	application de r sur f_1 ($x =$ `0`, $y =$ `SS0`)
f_3	`SS0 + SS0 = SSSS0`	application de r sur f_2 ($x =$ `S0`, $y =$ `SS0`)

Question 3 *Si on note Th l'ensemble des théorèmes de S. Que peut-on dire de ses liens d'inclusion avec Ax et avec $\mathcal{L}$?*

Remarque 3 Un autre système système formel que celui donné en exemple consiste à prendre comme seul axiome la fbf `0 + 0 = 0` et à considérer deux règles d'inférence : r et $\dfrac{x + y = z}{x + \texttt{S}y = \texttt{S}z}\, r'$. On peut montrer que tout théorème de S^+ est un théorème de ce système formel et inversement.

Étant donné une logique $(\mathcal{L}, \models)$, les logiciens essaient de construire un système formel $S = (\mathcal{L}, Ax, RI)$ de façon à avoir des liens forts entre les relations $\models$ et $\vdash^S$. On notera qu'une fbf de S est un élément de $\mathcal{L}$ et est donc une formule de cette logique. Idéalement, on aimerait avoir un système formel correct et complet, ce qui signifie que $\models$ et $\vdash^S$ coïncident et donc que tout théorème est une tautologie et réciproquement.

2.5.2 Un système formel pour $(\mathcal{LP}, \models)$

Le système formel S décrit ci-dessous est dû à D. Hilbert [6].

Dans ce système formel, les fbf ne sont pas toutes les formules propositionnelles : les seuls connecteurs utilisés sont $\neg$ et $\Rightarrow$. Cependant, on a vu qu'on pouvait se contenter d'eux sans que cela nuise à l'expressivité. En effet, les autres connecteurs peuvent être vus comme des abréviations s'appuyant sur ces deux connecteurs. En particulier, $\varphi \vee \psi$ peut être vu comme une abréviation de $\neg\varphi \Rightarrow \psi$ et $\varphi \wedge \psi$ peut être vu comme une abréviation de $\neg(\varphi \Rightarrow \neg\psi)$.

Les axiomes sont définis par les trois schémas d'axiomes suivants (où α, β et γ sont des fbf) :

$$\alpha \Rightarrow (\beta \Rightarrow \alpha) \tag{SA1}$$

$$(\alpha \Rightarrow (\beta \Rightarrow \gamma)) \Rightarrow ((\alpha \Rightarrow \beta) \Rightarrow (\alpha \Rightarrow \gamma)) \tag{SA2}$$

$$(\neg\alpha \Rightarrow \neg\beta) \Rightarrow (\beta \Rightarrow \alpha) \tag{SA3}$$

6. David Hilbert, mathématicien allemand (1862-1943).

On rappelle que le fait que ce soit des schémas d'axiomes entraîne le fait que le remplacement de α, β et γ par n'importe quelle fbf donne un axiome. Par exemple, $\neg a \Rightarrow ((b \Rightarrow c) \Rightarrow \neg a)$ est un axiome qui instancie (SA1).

La seule règle d'inférence de S est la suivante :

$$\frac{\alpha \qquad \alpha \Rightarrow \beta}{\beta}\, mp \qquad \text{pour } \alpha, \beta, \text{ deux fbf}$$

Cette règle s'appelle le *modus ponens*.

Par exemple, on peut montrer $\vdash^S a \Rightarrow a$ ($a \Rightarrow a$ est un thèorème de S) de la façon suivante :

Nom de la fbf	fbf	explications
f_1	$(a \Rightarrow ((a \Rightarrow a) \Rightarrow a)) \Rightarrow ((a \Rightarrow (a \Rightarrow a)) \Rightarrow (a \Rightarrow a))$	axiome (cf. (SA2))
f_2	$a \Rightarrow ((a \Rightarrow a) \Rightarrow a)$	axiome (cf. (SA1))
f_3	$(a \Rightarrow (a \Rightarrow a)) \Rightarrow (a \Rightarrow a)$	mp
f_4	$a \Rightarrow (a \Rightarrow a)$	axiome (cf. (SA1))
f_5	$a \Rightarrow a$	mp

Cet exemple illustre le fait que construire « à la main » une preuve dans ce système formel n'est généralement pas très facile (l'exercice ci-dessous montre néanmoins des exemples simples). Il existe des systèmes formels permettant de prouver plus facilement une formule (par exemple, la « déduction naturelle » : voir section 2.6) et des formalismes dans lesquels la définition de l'inférence se fait plus facilement par un système formel que par la sémantique.

Exercice 12 *Soit a et b, deux variables propositionnelles.*

Montrez les résultats suivants : $a \vdash^S b \Rightarrow a$, $\quad a \vdash^S a$, $\quad \neg b \Rightarrow \neg a \vdash^S a \Rightarrow b$.

Soit à présent, φ et ψ, deux formules propositionnelles.

Sans écrire de preuve du système formel S, répondez aux questions suivantes : $\varphi \vdash^S \psi \Rightarrow \varphi$? $\quad \varphi \vdash^S \varphi$? $\neg\psi \Rightarrow \neg\varphi \vdash^S \varphi \Rightarrow \psi$? Justifiez votre réponse.

2.5.3 Lien entre $\models$ et $\vdash^S$

Soit une logique $(\mathcal{L}, \models)$ et un système formel $S = (\mathcal{L}, Ax, RI)$ dont les fbf sont les formules de la logique.

On dit que S est ***correct*** par rapport à $(\mathcal{L}, \models)$ si, pour toute base de connaissances B et toute formule φ on a

$$B \vdash^S \varphi \quad \text{entraîne} \quad B \models \varphi \qquad \text{(correction de } \vdash^S\text{)}$$

On dit que S est ***complet*** par rapport à $(\mathcal{L}, \models)$ si, pour toute base de connaissances B et toute formule φ on a

$$B \models \varphi \quad \text{entraîne} \quad B \vdash^S \varphi \qquad \text{(complétude de } \vdash^S\text{)}$$

Par conséquent, si S est correct et complet, alors les relations $\models$ et $\vdash^S$ coïncident :

$$B \vdash^S \varphi \quad \text{si et seulement si} \quad B \models \varphi \quad \text{(correction et complétude de } \vdash^S\text{)}$$

Dans ce cas, l'ensemble des théorèmes de $\vdash^S$ est égal à l'ensemble des tautologies de $(\mathcal{L}, \models)$.

On peut montrer que le système formel S de Hilbert (section 2.5.2) est correct et complet par rapport à $(\mathcal{L}, \models)$.

Prouver qu'un système formel est correct vis-à-vis d'une logique est d'habitude assez simple. Pour cela, on prouve d'abord que tous les axiomes sont des tautologies. Ensuite, on montre que si les prémisses d'une règles d'inférences satisfont une interprétation $\mathcal{I}$, alors ses conclusions satisfont également $\mathcal{I}$. Le reste de la preuve se fait par récurrence sur la longueur de la preuve.

En revanche, la preuve de complétude est généralement beaucoup plus ardue (et ne sera pas envisagée dans ce cours).

Exercice 13 *On considère le système formel de Hilbert auquel on enlève un des trois schémas d'axiomes.*
Ce système formel est-il toujours correct par rapport à $(\mathcal{L}, \models)$? Pourquoi ?
À votre avis, est-il toujours complet ?
Pour répondre à la première question, il faut des arguments logiques. Pour répondre à la seconde question, cela n'est pas demandé.

Exercice 14 *Montrez que le système formel de Hilbert est correct pour la logique propositionnelle.*

Exercice 15 ⚠ *Cet exercice est un peu moins facile (on pourra ne le considérer que dans les grandes lignes).*
On considère à nouveau le système formel S^+. Définissez une sémantique en théorie des modèles pour $\mathcal{L}^+$ tel que S^+ soit correct et complet pour la logique $(\mathcal{L}^+, \models)$.

2.6 La déduction naturelle : un autre système formel correct et complet pour $(\mathcal{LP}, \models)$

La déduction naturelle est un système formel dû à G. Gentzen[7]. Un de ses avantages par rapport au système formel de Hilbert présenté ci-dessus est qu'elle est plus facile à manipuler, même si elle s'appuie sur plusieurs règles d'inférences (mais aucun axiome).

Remarque 4 La présentation de la déduction naturelle faite ci-dessous ne correspond pas exactement au système formel original de Gentzen, mais à une présentation introduisant un autre ensemble de règles.

2.6.1 Formules bien formées de la déduction naturelle

Les formules bien formées de la déduction naturelle sont les formules de la logique propositionnelle dans lesquelles les seuls connecteurs sont $\neg$, $\wedge$, $\vee$, $\Rightarrow$ et $\bot$.

Les autres connecteurs sont considérés comme des abréviations (p. ex., $\top$ sera une abréviation pour $\neg\bot$ et $\alpha \Leftrightarrow \beta$, une abréviation pour $(\alpha \Rightarrow \beta) \wedge (\beta \Rightarrow \alpha)$).

2.6.2 Règles d'inférence de la déduction naturelle

Règles liées à l'implication. L'unique règle d'inférence du système formel de Hilbert — le *modus ponens* — est aussi une règle d'inférence de la déduction naturelle. Rappelons-le :

$$\frac{\alpha \qquad \alpha \Rightarrow \beta}{\beta}\, mp \qquad \text{pour } \alpha, \beta, \text{ deux fbf}$$

On appelle aussi cette règle la règle d'*élimination de l'implication* : le connecteur $\Rightarrow$ apparaît en prémisse et pas en conclusion. Pour cette raison, on peut noter mp par $é_{\Rightarrow}$.

La règle d'inférence suivante est l'*introduction de l'implication*. Intuitivement, elle peut se comprendre ainsi :
— On veut prouver $\alpha \Rightarrow \beta$.
— On fait (temporairement) l'hypothèse α.
— Sur cette base, on prouve β.
— Donc, sous l'hypothèse α, on a β, ce qui se traduit par $\alpha \Rightarrow \beta$.

7. Gerhard Gentzen, mathématicien et logicien allemand (1909-1945).

Formellement, elle est définie comme suit :

$$\dfrac{\begin{array}{c}[\alpha]\\ \vdots\\ \beta\end{array}}{\alpha \Rightarrow \beta}\, i_{\Rightarrow} \qquad \text{pour } \alpha, \beta, \text{ deux fbf}$$

La notation $[\alpha]$ signifie qu'on fait temporairement l'hypothèse α. Si, après des étapes de raisonnements (symbolisés par « $\vdots$ ») on arrive à inférer β (en s'appuyant éventuellement sur α), alors on aura prouvé $\alpha \Rightarrow \beta$, sans nécessairement que α soit prouvé. L'hypothèse α est dite *déchargée* par la règle d'inférence (elle cesse d'être faite). Par exemple, on peut montrer $\vdash a \Rightarrow a$ de la façon suivante :

Nom de la fbf	fbf	explications
f_1	$[a]$	On suppose temporairement a.
f_2	a	On déduit a de f_1 (dans ce contexte).
f_3	$a \Rightarrow a$	On applique $i_{\Rightarrow}$ (avec $\alpha = a$ et $\beta = a$).

Le terme « introduction de l'implication » se comprend par le fait que le connecteur $\Rightarrow$ n'apparaît pas dans la prémisse de la règle mais apparaît dans sa conclusion. Notons dans cette preuve la présence, avant les formules f_1 et f_2, d'une barre d'indentation ($|$). Elle correspond au contexte de l'hypothèse temporaire f_1. Ce n'est que dans ce contexte qu'on peut inférer quelque chose à partir de cette hypothèse temporaire. En fermant ce contexte, on *décharge* cette hypothèse.

Règles liées à la conjonction. Trois règles sont liées au connecteur $\wedge$.

Les règles d'*élimination de la conjonction* sont définies par :

$$\dfrac{\alpha \wedge \beta}{\alpha}\, é_{\wedge 1} \qquad \dfrac{\alpha \wedge \beta}{\beta}\, é_{\wedge 2} \qquad \text{pour } \alpha, \beta, \text{ deux fbf}$$

La règle d'*introduction de la conjonction* est définie par :

$$\dfrac{\alpha \qquad \beta}{\alpha \wedge \beta}\, i_{\wedge} \qquad \text{pour } \alpha, \beta, \text{ deux fbf}$$

Règles liées à la disjonction. Trois règles sont liées au connecteur $\vee$.

La règle d'*élimination de la disjonction* est définie par :

$$\dfrac{\alpha \vee \beta \qquad \begin{array}{c}[\alpha]\\ \vdots\\ \gamma\end{array} \qquad \begin{array}{c}[\beta]\\ \vdots\\ \gamma\end{array}}{\gamma}\, é_{\vee} \qquad \text{pour } \alpha, \beta \text{ et } \gamma, \text{ trois fbf}$$

Autrement écrit, si on peut prouver γ partant de α et partant de β, alors, de $\alpha \vee \beta$, on peut prouver γ en utilisant $é_{\vee}$.

On a deux règles d'*introduction de la disjonction* :

$$\dfrac{\alpha}{\alpha \vee \beta}\, i_{\vee 1} \qquad \dfrac{\beta}{\alpha \vee \beta}\, i_{\vee 2} \qquad \text{pour } \alpha, \beta, \text{ deux fbf}$$

Règles liées à la négation logique. Deux règles sont liées au connecteur $\neg$.

La règle d'*élimination de la négation* est :

$$\dfrac{\neg\neg\alpha}{\alpha}\, é_{\neg} \qquad \text{pour } \alpha, \text{ une fbf}$$

La règle d'*introduction de la négation* est :

$$\dfrac{\begin{array}{c}[\alpha]\\ \vdots \\ \beta \\ \vdots \\ \neg\beta\end{array}}{\neg\alpha} i_\neg \qquad \text{pour } \alpha, \beta, \text{ deux fbf}$$

Autrement écrit, si, partant de l'hypothèse temporaire α, on arrive à prouver une formule et sa négation, c'est que α est faux.

Règles liées au connecteur $\bot$. Deux règles sont liées au connecteur $\bot$.

La règle d'*élimination du* $\bot$ est :

$$\dfrac{\bot}{\alpha} é_\bot \qquad \text{pour } \alpha, \text{ une fbf}$$

La règle d'*introduction du* $\bot$ est :

$$\dfrac{\alpha \qquad \neg\alpha}{\bot} i_\bot \qquad \text{pour } \alpha, \text{ une fbf}$$

Règle de réduction à l'absurde. La régle de *réduction à l'absurde* est liée au raisonnement par l'absurde (intuitivement si, partant de l'hypothèse $\neg\alpha$ j'arrive à une contradiction, c'est que cette hypothèse est absurde et donc qu'on a α) :

$$\dfrac{\begin{array}{c}[\neg\alpha]\\ \vdots \\ \bot\end{array}}{\alpha} r\grave{a}a$$

2.6.3 Preuves en déduction naturelle : principe et exemples

Principe. La différence avec les systèmes formels présentés dans la section précédente (en particulier, le système formel de Hilbert) est la notion de contexte : en début de contexte on fait une hypothèse (marquée par des crochets et un début d'indentation) et cette hypothèse est déchargée en fin de contexte : cette hypothèse pourra être utilisée dans le contexte (et dans ses sous-contextes éventuels), mais pas à l'extérieur du contexte.

On notera $B \vdash^{DN} \varphi$ si la formule φ peut être déduite par déduction naturelle à partir de l'ensemble de formules B.

Les figures 2 et 3 sont des exemples de preuves en déduction naturelle.

Exercice 16 *Montrez les résultats suivants :*

- $\neg\neg a \vdash^{DN} a$ *de deux façons différentes (l'une utilisant* $é_\neg$ *et l'autre sans l'utiliser) ;*
- $a \vdash^{DN} \neg\neg a$ *;*
- $(a \wedge b) \vee (a \wedge \neg b) \vdash^{DN} a$
- $(a \vee b) \wedge (a \vee \neg b) \vdash^{DN} a$ *(indication : on fait un raisonnement par l'absurde avec* $\neg a$ *puis on va prouver d'abord* b *— en utilisant l'hypothèse temporaire* a *et* $é_\vee$ *sur* $a \vee b$ *— et ensuite* $\neg b$ *— de façon similaire) ;*
- $\vdash^{DN} a \vee \neg a$ *(indication : on fait un raisonnement par l'absurde à partir de* $\neg(a \vee \neg a)$ *puis, si on suppose* a*, on peut en déduire* $a \vee \neg a$ *d'où contradiction, d'où* $\neg a$ *; de façon symétrique, on suppose* $\neg a$ *pour en déduire* $\neg\neg a$ *et donc on a une contradiction entre* $\neg a$ *et* $\neg\neg a$ *qui permet de conclure).*

Exercice 17 *Montrez que la déduction naturelle est un système formel correct pour la logique propositionnelle.*

Nom de la fbf	fbf	explications
f_1	$[a \Rightarrow b \wedge c]$	hypothèse temporaire
f_2	$\quad [a]$	hypothèse temporaire
f_3	$\quad b \wedge c$	mp sur f_2 et f_1
f_4	$\quad b$	$é_{\wedge 1}$ sur f_3
f_5	$\quad c$	$é_{\wedge 2}$ sur f_3
f_6	$a \Rightarrow b$	$i_{\Rightarrow}$ sur f_2 et f_4
f_7	$a \Rightarrow c$	$i_{\Rightarrow}$ sur f_2 et f_5
f_8	$(a \Rightarrow b) \wedge (a \Rightarrow c)$	$i_{\wedge}$ sur f_6 et f_7
f_9	$(a \Rightarrow b \wedge c) \Rightarrow ((a \Rightarrow b) \wedge (a \Rightarrow c))$	$i_{\Rightarrow}$ sur f_1, f_8

FIGURE 2 – Preuve de $\vdash^{DN} (a \Rightarrow b \wedge c) \Rightarrow ((a \Rightarrow b) \wedge (a \Rightarrow c))$.

Nom de la fbf	fbf	explications
f_1	$\neg a \vee b$	hypothèse
f_2	a	hypothèse
f_3	$\quad [\neg a]$	hypothèse temporaire
f_4	$\quad \neg a$	déduit de f_3 (dans ce contexte)
f_5	$\quad \bot$	$i_{\bot}$ sur f_2 et f_4
f_6	$\quad b$	$é_{\bot}$ sur f_5
f_7	$\quad [b]$	hypothèse temporaire
f_8	$\quad b$	déduit de f_7 (dans ce contexte)
f_9	b	$é_{\vee}$ sur f_1, (f_3 à f_6) et (f_7 à f_8)

FIGURE 3 – Preuve de $\{\neg a \vee b, a\} \vdash^{DN} b$.

2.7 Aspects algorithmiques

2.7.1 Complexité

Comme cela a été évoqué dans l'exercice 5, les inférences évoquées dans ce chapitre dans la logique propositionnelle se ramènent au problème de la satisfiabilité, problème algorithmique appelé SAT. Réciproquement, SAT peut se ramener à ces autres inférences. En l'occurrence, « se ramener à » se fait de façon rapide : on parle, en complexité, de « pont polynomial ». Par conséquent, étudier la complexité des inférences revient à étudier la complexité de SAT.

Le théorème de Cook (ou théorème de Cook-Levin) indique que ***SAT est un problème NP-complet*** [8]. Les notions de complexité de problèmes algorithmiques ne sont pas étudiées dans ce cours, mais cela signifie que personne à cette heure ne connaît d'algorithme de complexité non exponentielle dans le pire cas pour résoudre SAT.

2.7.2 Un algorithme pour SAT

L'algorithme décrit dans cette section peut être vu comme une amélioration du parcours d'une table de vérité.

Redescription de l'algorithme SAT utilisant des tables de vérité. Une manière pour implanter SAT consiste à parcourir une table de vérité, laquelle contient 2^n lignes (ou interprétations), n étant le nombre de variables propositionnelles. Ce parcours peut être vu comme un parcours récursif :

8. Stephen A. Cook, informaticien et mathématicien américano-canadien (1939-) et Leonid A. Levin, informaticien russo-ukraino-américain (1948-).

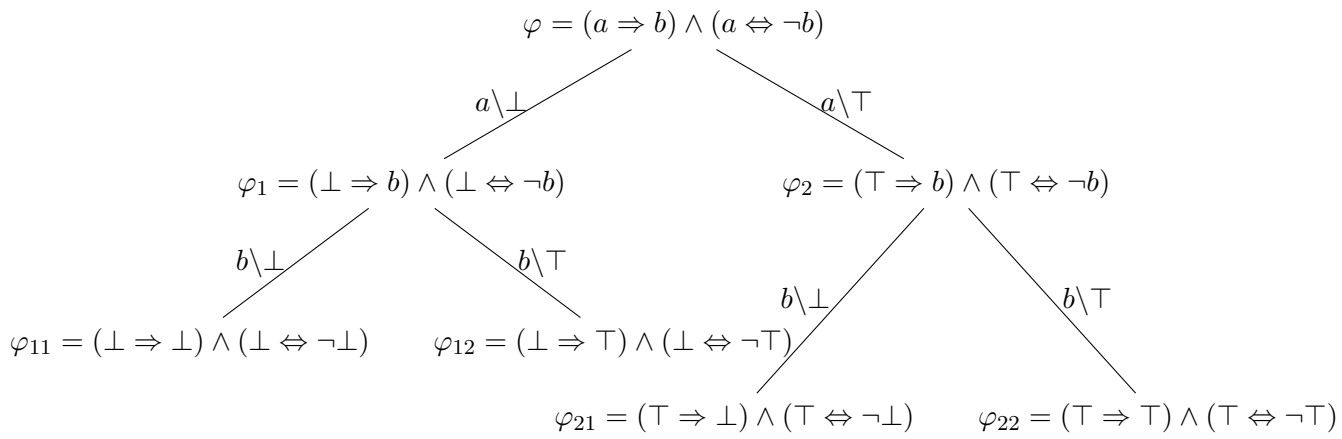


FIGURE 4 – Satisfiabilité d'une formule par table de vérité vue comme un parcours d'arbre binaire.

— Si φ ne contient pas de variable, alors, on peut l'évaluer directement, par exemple : pour toute interprétation $\mathcal{I}$, $(\top \wedge (\neg\top \vee \bot))^{\mathcal{I}} = (\texttt{V}$ **et** $(\textbf{non}\ \texttt{V}\ \textbf{ou}\ \texttt{F})) = \texttt{F}$ (rappelons que $\bot^{\mathcal{I}} = \texttt{F}$ et $\top^{\mathcal{I}} = \texttt{V}$). Dans ce cas, l'algorithme termine sur la valeur de cette évaluation.
— Si au contraire φ contient au moins une variable, on *choisit* une de ces variables. Notons-là x.
— On crée les deux formules $\varphi[x\backslash\bot]$ et $\varphi[x\backslash\top]$.
— Si l'appel récursif de l'algorithme sur $\varphi[x\backslash\bot]$ donne `V`, on retourne `V`.
— Sinon, on retourne le résultat de l'appel récursif sur $\varphi[x\backslash\top]$.

Considérons l'exemple suivant. Soit $\varphi = (a \Rightarrow b) \wedge (a \Leftrightarrow \neg b)$. Si on choisit a avant b (noté $a \prec b$), le déroulement de l'algorithme peut se voir comme un parcours en profondeur de l'arbre binaire de la figure 4.

Cet arbre est parcouru en profondeur et chaque fois qu'une feuille est générée, elle est évaluée. φ_{11} est évaluée en premier et donne $\varphi_{11}^{\mathcal{I}} = \texttt{F}$. φ_{12} est évaluée en deuxième et donne $\varphi_{12}^{\mathcal{I}} = \texttt{V}$. Par conséquent, l'exploration de l'arbre s'arrête et l'algorithme indique que φ est satisfiable. De plus, le chemin de φ à φ_{12} donne une interprétation $\mathcal{I} = \overline{a}b$. Ainsi, seules deux des quatre interprétations sont parcourues dans cet exemple : $\overline{ab}$ et $\overline{a}b$. Si on veut toutes les interprétations, il faut parcourir tout l'arbre.

Utilisation de simplifications. Dans bien des cas, il n'est pas nécessaire de parcourir une branche d'un arbre binaire représentant l'algorithme par table de vérité jusqu'au bout. En effet, on peut, en utilisant des règles de simplification, savoir qu'une formule sera équivalente à $\bot$ ou à $\top$, même si elle contient des variables. Ainsi $(a \wedge \neg b) \vee \top$ est équivalent à $\top$ et donc, après une telle simplification, il n'est pas nécessaire de poursuivre l'algorithme (alors qu'il restait deux variables).

Ci-dessous sont présentées des règles de simplification sous la forme d'équivalences à lire de gauche à droite (φ est une formule propositionnelle, x est une variable propositionnelle) :

$$\neg\bot \equiv \top \qquad \neg\top \equiv \bot \qquad \bot \wedge \varphi \equiv \bot \qquad \top \wedge \varphi \equiv \varphi \qquad \bot \vee \varphi \equiv \varphi \qquad \top \vee \varphi \equiv \top$$
$$\bot \Rightarrow \varphi \equiv \top \qquad \varphi \Rightarrow \bot \equiv \neg\varphi \qquad \top \Rightarrow \varphi \equiv \varphi \qquad \varphi \Rightarrow \top \equiv \top \qquad \bot \Leftrightarrow \varphi \equiv \neg\varphi \qquad \top \Leftrightarrow \varphi \equiv \varphi$$
$$\neg\neg\varphi \equiv \varphi \qquad x \wedge \neg x \equiv \bot \qquad x \vee \neg x \equiv \top$$

En plus de ces règles, il y a les règles induites par commutativité des opérations $\wedge$, $\vee$ et $\Leftrightarrow$ et associativité des opérations $\wedge$ et $\vee$ (commutativité et associativité modulo l'équivalence, évidemment). Par exemple, l'équivalence $\bot \wedge \varphi \equiv \bot$ entraîne l'équivalence $\varphi \wedge \bot \equiv \bot$. Ainsi, $\bot \vee \neg(\top \Rightarrow (a \vee \neg\top)) \equiv \neg a$.

Si on décrit le déroulement de l'algorithme par un arbre binaire parcouru en profondeur, les feuilles simplifiées en $\bot$ ne sont pas poursuivies mais le parcours se poursuit par *backtracking* et les feuilles simplifiées en $\top$ conduisent à la fin de l'algorithme (avec résultat `V`).

La figure 5 décrit un exemple. L'arbre de recherche sans les simplifications contient $2^0 + 2^1 + \ldots + 2^n = 2^{n+1} - 1$ nœuds pour une formule insatisfiable. Sans les simplifications, l'arbre de cette figure contiendrait donc

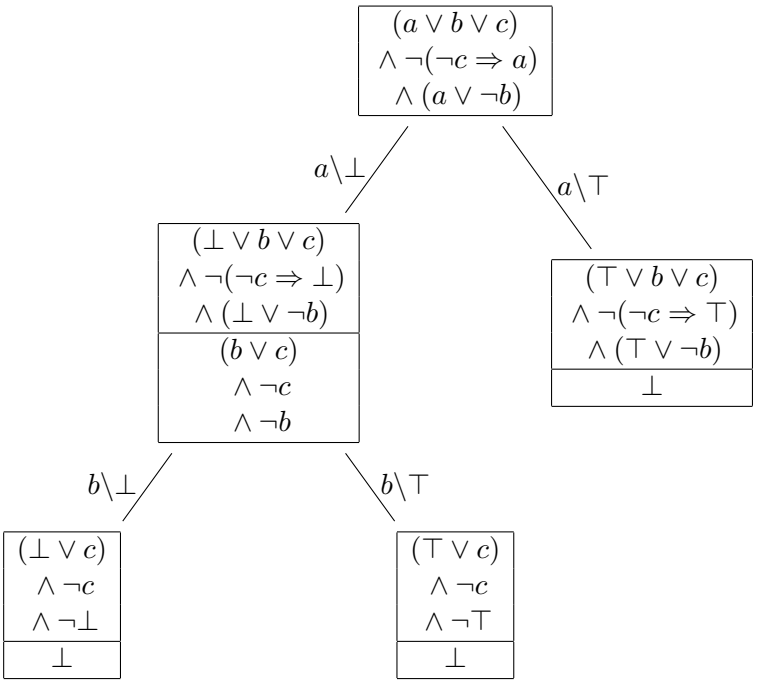


FIGURE 5 – Application de l'algorithme de satisfiabilité utilisant des simplifications sur $\varphi = (a \vee b \vee c) \wedge \neg(\neg c \Rightarrow a) \wedge (a \vee \neg b)$ avec l'ordre $a \prec b \prec c$ (la partie supérieure des encadrés en deux parties est la formule après substitution, la partie inférieure est obtenue par simplification de la partie supérieure). La conclusion est que φ est insatisfiable.

15 nœuds au lieu de 5. On notera que l'ordre sur les variables peut influer le nombre de nœuds parcourus et donc le temps de calcul.

Exercice 18 *Utilisez la méthode précédente pour étudier la satisfiabilité de* $\varphi = \neg(a \Rightarrow (b \Leftrightarrow (c \vee \neg a))) \wedge \neg(b \Rightarrow a)$*, avec l'ordre* $a \prec b \prec c$ *et l'ordre* $c \prec b \prec a$*.*

Exercice 19 *Utilisez la méthode précédente pour prouver que la formule suivante est une tautologie :* $\varphi = (a \Rightarrow (b \Rightarrow c)) \Leftrightarrow ((a \Rightarrow b) \Rightarrow (a \Rightarrow c))$*.*

Exercice 20 *Soit* $\mathcal{V} = \{é, m, or, ov, t\}$ *un ensemble de variables qui sont choisies pour signifier ceci :*

$é$:	*être un échidné*	ov :	*être un ovipare*
m :	*être un mammifère*	t :	*être l'individu Toto*
or :	*être un ornithorynque*		

On considère ensuite les formules suivantes :

$\varphi_1 = (m \wedge ov) \Leftrightarrow (é \vee or)$	(Les seuls mammifères ovipares sont les échidnés et les ornithorynques.)
$\varphi_2 = \neg(é \wedge or)$	(Un individu ne peut pas être à la fois un échidné et un ornithorynque.)
$\varphi_3 = t \Rightarrow m \wedge ov \wedge \neg é$	(Toto est un mammifère ovipare qui n'est pas un échidné.)
$\varphi_4 = t \Rightarrow or$	(Toto est un ornithorynque.)

Utilisez la méthode précédente pour montrer que $\{\varphi_1, \varphi_2, \varphi_3\} \models \varphi_4$*.*

Vous essayerez de choisir l'ordre des variables pour rendre le calcul moins long.

Exercice 21 *Les équivalences donnant les règles de simplification ne concernent que les connecteurs $\neg$, $\wedge$, $\vee$, $\Rightarrow$ et $\Leftrightarrow$. Proposez des règles pour les connecteurs $\uparrow$, $\downarrow$ et $\oplus$.*

2.7.3 Transformations de formules

D'autres algorithmes bien plus efficaces existent. Par exemple, l'algorithme de Davis et Putnam est très utilisé. Beaucoup de ces algorithmes supposent que les formules considérées sont sous des ***formes normales*** particulières. Cette section a pour objectif de décrire trois formes normales souvent utilisées. Elle est aussi l'occasion d'introduire différentes notions.

Un ***littéral*** est une formule de la forme a (littéral dit positif) ou $\neg a$ (littéral dit négatif), pour $a \in \mathcal{V}$.

Une formule φ est sous ***forme normale négative*** (FNN ou, en anglais, NNF) si le connecteur $\neg$ n'apparaît que devant les variables propositionnelles dans φ. Autrement écrit, une formule sous FNN est une formule dans laquelle $\neg$ n'apparaît pas en dehors d'un littéral. Mettre une formule sous FNN c'est trouver une formule logiquement équivalente qui est sous cette forme. Pour cela, on utilise l'involutivité de la négation et les lois de de Morgan. Par exemple :

$$\neg(\neg(a \vee \neg b) \wedge (c \vee d)) \xrightarrow{(8)} \neg\neg(a \vee \neg b) \vee \neg(c \vee d) \xrightarrow{(7)} (a \vee \neg b) \vee \neg(c \vee d) \xrightarrow{(9)} (a \vee \neg b) \vee (\neg c \wedge \neg d)$$

Une ***clause*** est une disjonction de littéraux. Par exemple, $a \vee \neg b \vee c \vee \neg d$ est une clause contenant deux littéraux positifs (a et c) et deux littéraux négatifs ($\neg b$ et $\neg d$). Une clause unaire est une clause avec un seul littéral : a et $\neg b$ sont des clauses unaires.

Une formule sous ***forme normale conjonctive*** (FNC ou, en anglais, CNF) est une conjonction de clauses. On peut mettre toute formule sous FNC en utilisant l'involutivité de la négation, les lois de de Morgan, les lois de distributivité et les équivalences relatives aux connecteurs autres que $\neg$, $\wedge$ et $\vee$.

Une formule sous ***forme normale disjonctive*** (FND ou, en anglais, DNF) est une disjonction de conjonctions de littéraux. On peut mettre toute formule sous FND en utilisant les mêmes équivalences que pour la mise sous FNC.

On notera qu'une formule sous FNC ou sous FND est nécessairement sous FNN.

Exercice 22 *Mettez les formules suivantes sous FNN, FNC et FND :*

$$\varphi_1 = \neg(a \vee \neg b) \qquad \varphi_2 = \neg(a \Leftrightarrow b) \wedge c \qquad \varphi_3 = \neg(a \Rightarrow b) \wedge a \qquad \varphi_4 = \neg(a \wedge \neg(b \vee \neg(c \wedge \neg d)))$$

2.7.4 La méthode des tableaux sémantiques

La méthode des tableaux sémantiques est une approche générale pour effectuer des déductions dans de nombreuses logiques. Nous allons en présenter une version simple pour la logique propositionnelle et l'étude du problème SAT.

Soit $\varphi \in \mathcal{LP}$. Dans l'exemple suivi de cette section, on essaiera de prouver $\models \psi$ avec

$$\psi = ((s \Rightarrow h) \wedge (h \Rightarrow m)) \Rightarrow (s \Rightarrow m)$$

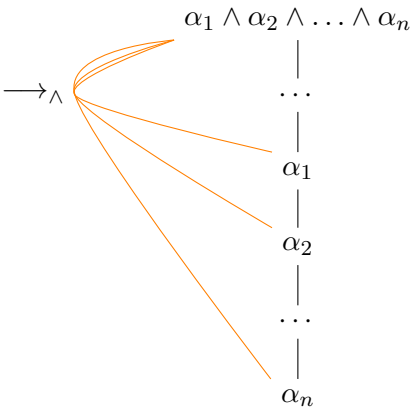
qui une formalisation du célèbre syllogisme : Socrate est un homme, tout homme est mortel donc Socrate est mortel. Dans cette formalisation, s signifie « être Socrate », h, « être un homme » et m, « être mortel ». Il suffit donc de montrer que $\varphi = \neg\psi$ est insatisfiable.

La première étape consiste à mettre la formule sous FNN. Soit φ' sous FNN telle que $\varphi \equiv \varphi'$. Dans l'exemple, on a la suite d'équivalences suivante :

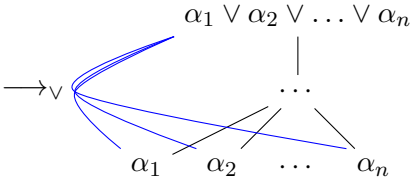
$$\begin{aligned}\varphi &\equiv \neg(\neg((\neg s \vee h) \wedge (\neg h \vee m)) \vee (\neg s \vee m)) \\ &\equiv (\neg s \vee h) \wedge (\neg h \vee m) \wedge s \wedge \neg m = \varphi'\end{aligned}$$

Dans un deuxième temps, on construit un arbre de formules dont la racine est φ' et les nœuds suivants sont obtenus par les deux transformations suivantes :

$\longrightarrow_\wedge$ Si dans une branche de l'arbre on a une formule de la forme $\alpha = \alpha_1 \wedge \alpha_2 \wedge \ldots \wedge \alpha_n$ alors on prolonge « verticalement » la branche par les nœuds contenant les formules α_i, sauf les α_i qui sont déjà dans la branche :



$\longrightarrow_\vee$ Si dans une branche de l'arbre on a une formule de la forme $\alpha = \alpha_1 \vee \alpha_2 \vee \ldots \vee \alpha_n$ alors on prolonge « horizontalement » la branche par les nœuds contenant les formules α_i, sauf les α_i qui sont déjà dans la branche :



Si, lors du développement de cet arbre, on a dans une même branche deux nœuds avec les formules x et $\neg x$ où $x \in \mathcal{V}$ alors cette branche contient un conflit, noté $\square$, car la base de connaissances $\{x, \neg x\}$ est insatisfiable. On ne prolonge pas une branche au-delà d'un tel conflit.

Si toutes les branches aboutissent à un conflit alors la formule φ' est insatisfiable (et donc, φ l'est également).

En revanche, si on ne peut pas prolonger davantage l'arbre et qu'il existe au moins une branche sans conflit, alors on peut montrer que φ' (et donc φ) est satisfiable. Par ailleurs, l'analyse d'une branche satisfiable permet de mettre en évidence au moins une interprétation.

La figure 6 décrit le déroulement de la méthode et montre que φ est insatisfiable.

Exercice 23 *Reprenez les exercices 18, 19 et 20 en utilisant la méthode des tableaux sémantiques.*

La méthode des tableaux sémantiques, comme indiquée plus haut, s'applique à de nombreux formalismes. Pour cela, sont définis :

— Des règles de transformation (telles que $\longrightarrow_\wedge$ et $\longrightarrow_\vee$) qui à une ou des formule(s) associent une ou des formule(s) ;

— Des moyens d'identifier des conflits, i.e., des incohérences faciles à détecter de façon syntaxique (p. ex., $\{x, \neg x\}$) ;

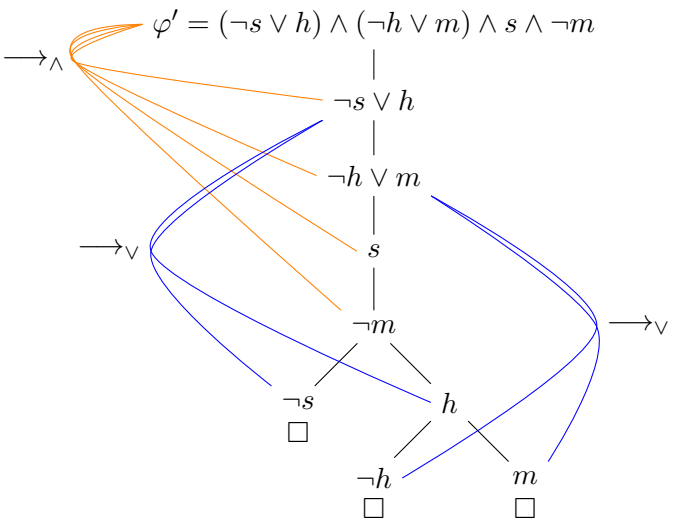


FIGURE 6 – Méthode des tableaux sémantiques en logique propositionnelle sur un exemple.

— D'éventuelles méta-règles pour contrôler le développement de l'arbre, assurer la terminaison et améliorer le temps de calcul.

Les *logiques de descriptions* sont des formalismes très développés qui sont équivalents à des fragments de la logique du premier ordre (à quelques détails près). Dans ces logiques, étudiées en représentation des connaissances et dans les bases de données, de nombreuses variantes très optimisées de la méthode des tableaux sémantiques ont été (et sont encore) étudiées.

2.7.5 Inférences avec des clauses de Horn propositionnelles

Une façon d'avoir des inférences plus rapides consiste à réduire l'expressivité de la logique : on perd en expressivité mais on gagne en temps de calcul. On considère la logique $(\mathcal{LHP}, \models_{\mathcal{LHP}})$ (logique de Horn propositionnelle) qui est un fragment de $(\mathcal{LP}, \models)$: si $\varphi \in \mathcal{LHP}$ alors $\varphi \in \mathcal{L}$ et si $B \models_{\mathcal{LHP}} \varphi$ alors $B \models \varphi$ (pour alléger l'écriture, on écrira $\models$ à la place de $\models_{\mathcal{LHP}}$).

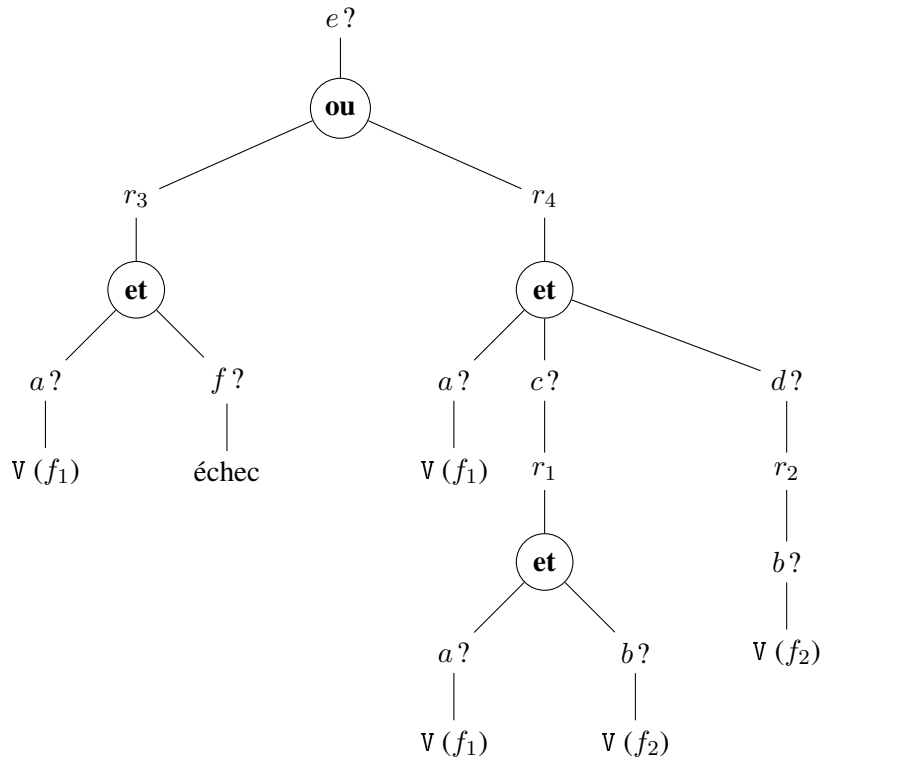
Une formule de $\mathcal{LHP}$ est une ***clause de Horn*** [9], c'est-à-dire une clause contenant exactement un littéral positif :

$\neg a \vee \neg b \vee c$,	$\neg a \vee b$	et	c	sont des clauses de Horn
$\neg a \vee \neg b \vee \neg c$,	$a \vee \neg b \vee c$	et	$\neg b$	ne sont pas des clauses de Horn

Une clause de Horn contenant au moins deux littéraux est équivalente à une implication dont la partie gauche est une conjonction de variables et la partie droite est une seule variable. Par exemple, $\neg a \vee \neg b \vee \neg c \vee d \equiv a \wedge b \wedge c \Rightarrow d$. Une clause de Horn a un seul littéral est réduite à une variable. Par extension, on appellera également clause de Horn ce genre d'implication. Dans ce contexte, on appelle les clauses de Horn à plusieurs littéraux des ***règles*** et les clauses de Horn à un seul littéral, des ***faits***. Ainsi, on peut définir la base de connaissances suivante de $(\mathcal{LHP}, \models)$:

$$
\begin{aligned}
B = \{ & a \wedge b \Rightarrow c, && \text{(règle } r_1\text{)} \\
& b \Rightarrow d, && \text{(règle } r_2\text{)} \\
& a \wedge f \Rightarrow e, && \text{(règle } r_3\text{)} \\
& a \wedge c \wedge d \Rightarrow e, && \text{(règle } r_4\text{)} \\
& a, && \text{(fait } f_1\text{)} \\
& b\} && \text{(fait } f_2\text{)}
\end{aligned}
$$

9. Alfred Horn, mathématicien américain (1918-2001).



Légende :

x ?	On cherche à prouver x.
V (f_k)	La variable au-dessus est validée car c'est un fait f_k de la base.

FIGURE 7 – Déroulement d'un chaînage arrière (le parcours de l'arbre « et-ou » se fait en profondeur).

Si on cherche à montrer que $B \models e$, on peut procéder de la même manière que dans la section précédente (toute clause de Horn est une formule propositionnelle), mais on peut également procéder par un mécanisme de ***chaînage arrière***, consistant à partir du fait à démontrer, pour remonter vers les faits connus.

La figure 7 décrit, sur l'exemple, le déroulement de ce chaînage arrière, par un arbre et-ou : il faut que chaque réponse sous un nœud « et » donne V et qu'au moins une réponse sous un nœud « ou » donne V pour que le résultat soit V.

Ce problème d'inférence avec les clauses de Horn est polynomial (et, en pratique, rapide).

Notons qu'un chaînage avant est également possible : dans ce formalisme, on peut facilement générer tous les faits qui sont vérifiés. En revanche, ce chaînage avant ne se généralise pas au cadre des clauses de Horn en logique du premier ordre, alors que le chaînage arrière, si.

Exercice 24 *Soit $r'_4 = a \wedge d \Rightarrow f$ et $B' = (B \setminus \{r_4\}) \cup \{r'_4\}$ où B est la base de connaissances décrite ci-dessus.*
Utilisez le chaînage arrière pour montrer $B \models e$.

Exercice 25 *On considère la base de $(\mathcal{LHP}, \models)$ suivante :*

$$B = \{a \wedge b \Rightarrow c, \quad \text{(règle } r_1\text{)}$$
$$c \Rightarrow b, \quad \text{(règle } r_2\text{)}$$
$$a\} \quad \text{(fait } f_1\text{)}$$

Si on cherche à prouver que $B \models c$ en utilisant le chaînage arrière, que constate-t-on ?
Comment remédier à cette situation ?

2.8 Représenter des connaissances en logique propositionnelle

La représentation des connaissances a pour objet la transformation de connaissances informelles (souvent exprimées en « langage naturel ») vers une logique.

Par exemple, considérons l'énoncé suivant :

> « La logique est une matière passionnante. Quand une matière est trop facile, elle n'est pas passionnante. »

Le formaliser consiste à construire une base de connaissances de la logique choisie et pour cela, on construit d'abord le vocabulaire qui, en logique propositionnelle correspond aux variables propositionnelles.

Dans ce formalisme, on ne considère qu'un individu à la fois, donc les objets considérés doivent être « de même type » (les guillemets sont là pour dire que cette notion de type est intuitive et ne correspond pas à une notion formelle de type). Ici, on a des notions à considérer que sont « logique », « matière passionnante » et « matière trop facile », qu'on va traduire par les variables propositionnelles `logique`, `matière_passionnante` et `matière_trop_facile` (autant donner des noms explicites).

Ensuite, on peut souvent procéder phrase par phrase. « La logique est une matière passionnante. » peut être lue sous la forme « Si x est la logique alors x est une matière passionnante. » On voit qu'il y a une implication et cette phrase est formalisée en φ_1 avec

$$\varphi_1 \quad = \quad \texttt{logique} \Rightarrow \texttt{matière\_passionnante}$$

Pour formaliser la deuxième phrase, « Quand une matière est trop facile, elle n'est pas passionnante. », on peut procéder comme précédemment et parvenir à la phrase « Si x est une matière trop facile alors x n'est pas passionnante. » qu'on formalisera en

$$\varphi_2 \quad = \quad \texttt{matière\_trop\_facile} \Rightarrow \neg\texttt{matière\_passionnante}$$

D'où la base de connaissances $\{\varphi_1, \varphi_2\}$.

On peut montrer qu'on a :

$$\{\varphi_1, \varphi_2\} \models \varphi_3 \qquad \text{avec } \varphi_3 \quad = \quad \texttt{logique} \Rightarrow \neg\texttt{matière\_trop\_facile}$$

φ_3 peut s'interpréter en français par « La logique n'est pas une matière trop facile. » Par conséquent, si on admet l'énoncé original, il faut (pour peu qu'on soit rationnel) admettre cette conclusion. De façon duale, si on conteste φ_3, cela signifie qu'on conteste φ_1 ou φ_2 (ou les deux).

Remarque 5 Le choix des noms des variables propositionnelles est important du point de vue des humains qui les manipulent. En terme de logique, ils sont indifférents, de la même façon que les identifieurs d'un programme pourraient s'appeler `i1`, `i2`, `i3`, ... Cela ne « dérangerait » pas le compilateur ou l'interpréteur, mais ce ne serait pas très pratique pour les programmeurs.

Par exemple, on pourrait interpréter `logique` par « être un paon », `matière_passionnante` par « être un bel oiseau » et `matière_trop_facile` par « être un oiseau chanteur ». L'inférence $\{\varphi_1, \varphi_2\} \models \varphi_3$ se lira :

> Si on suppose que le paon est un bel oiseau et qu'un oiseau chanteur n'est pas un bel oiseau, alors on peut en déduire que le paon n'est pas un oiseau chanteur.

Exercice 26 *« Traduire » en logique propositionnelle les phrases suivantes :*

- *Quand il pleut, je prends mon parapluie ou mon chapeau.*
- *Selon qu'il fasse nuit ou jour, j'allume la lumière ou pas.*
- *Tout ce qui est rare est cher.*
- *Un diamant qui n'est pas cher, c'est rare.*

Exercice 27 *On considère les affirmations suivantes (d'après Lewis Carroll*[10]*) :*

(1) Aucun canard ne danse la valse.

(2) Aucun officier ne refuse jamais de danser la valse.

(3) Toutes mes volailles sont des canards.

Transcrire chaque affirmation en formule propositionnelle.

Sachant que Maurice est une de mes volailles, on se pose la question suivante : « Maurice est-il officier ? » On demande :

— *De formaliser cette question sous la forme $\varphi \models \psi$.*

— *D'y répondre, par exemple en appliquant un algorithme implantant SAT.*

2.9 Quand la logique propositionnelle est-elle insuffisante ?

Comme l'illustrent les exemples et exercices de représentation de connaissances ci-dessus, la logique propositionnelle est adaptée pour représenter des connaissances sur « un individu à la fois ». Si on veut représenter par exemple, le fait que les amis de mes amis sont mes amis, cela devient plus difficile d'utiliser la logique propositionnelle.

En revanche, la logique du premier ordre permet de faire cela.

*****

3 La logique du premier ordre

Ce chapitre est organisé comme le précédent. Dans un premier temps, des rappels (supplémentaires) sur la théorie des ensembles sont donnés avant de décrire la logique du premier ordre $(\mathcal{L}1\mathcal{O}, \models)$: syntaxe, sémantique et système formel. Puis, des aspects algorithmiques et des problématiques de représentation des connaissances seront étudiés.

3.1 Deuxième rappel sur les ensembles

Soit E_1, E_2, …, E_n : n ensembles (avec $n \geqslant 1$). Le ***produit cartésien*** $E_1 \times E_2 \times \ldots \times E_n$ est l'ensemble des n-uplets $(u_1, u_2, \ldots, u_n)$ tels que $u_1 \in E_1$, $u_2 \in E_2$, …, $u_n \in E_n$. Si $E_1 = E_2 = \ldots = E_n = E$, alors $E_1 \times E_2 \times \ldots \times E_n$ est noté E^n.

Une ***relation*** R sur $E_1 \times E_2 \times \ldots \times E_n$ est un sous-ensemble de ce produit cartésien. $(u_1, u_2, \ldots, u_n) \in R$ se note souvent $R(u_1, u_2, \ldots, u_n)$ ou, dans le cas $n = 2$, $u_1\ R\ u_2$ (de façon infixée).

Dans le cas où $n = 1$, le produit cartésien E_1 est assimilé à l'ensemble E_1 : un ensemble peut être vu comme une relation unaire.

Une ***relation binaire*** est une relation sur un produit cartésien entre deux ensembles. Par exemple, la relation $<$ sur $\mathbb{N} \times \mathbb{N}$ est l'ensemble des couples (x, y) tels que x est strictement plus petit que y :

$$\begin{aligned} < = \{&(0,1),(0,2),(0,3),(0,4),\ldots,\\ &(1,2),(1,3),(1,4),(1,5),\ldots,\\ &(2,3),(2,4),(2,5),(2,6),\ldots,\\ &\ldots\} \end{aligned}$$

Soit E et F, deux ensembles. Une ***fonction partielle*** de E dans F est une relation f sur $E \times F$ telle que pour tout $x \in E$ il existe *au plus* un $y \in F$ tel que $x\ f\ y$. Autrement écrit : pour $x \in E$, $y_1, y_2 \in F$, si $x\ f\ y_1$ et $x\ f\ y_2$

10. Charles Dodgson dit Lewis Carroll, écrivain et logicien britannique (1832-1898).

alors $y_1 = y_2$. On dit que la fonction partielle f est définie en x s'il existe $y \in F$ tel que $x\ f\ y$, dans ce cas, ce y est uniquement défini et on le note souvent par $f(x)$. Une ***fonction*** de E dans F est une fonction partielle de E dans F qui est définie en tout $x \in E$.

On définit une fonction à n arguments comme une fonction sur un produit cartésien $E_1 \times E_2 \times \ldots \times E_n$ à valeur dans F. Dans ce cas, au lieu de noter $f((u_1, u_2, \ldots, u_n))$ on note d'habitude $f(u_1, u_2, \ldots, u_n)$. Pour déclarer une telle fonction, on écrit :

$$\begin{array}{rccc} f: & E_1 \times E_2 \times \ldots \times E_n & \longrightarrow & F \\ & (u_1, u_2, \ldots, u_n) & \longmapsto & f(u_1, u_2, \ldots, u_n) \end{array}$$

ou, plus simplement :

$$f : (u_1, u_2, \ldots, u_n) \in E_1 \times E_2 \times \ldots \times E_n \mapsto f(u_1, u_2, \ldots, u_n) \in F$$

3.2 Syntaxe de $(\mathcal{L}1\mathcal{O}, \models)$

Exemples de formules. Les expressions suivantes sont des formules de $\mathcal{L}1\mathcal{O}$:

$$\forall x\ \neg \texttt{solitaire}(x) \Rightarrow \exists y\ \texttt{connaît}(x, y) \qquad (\varphi_1)$$

$$\forall x \forall y \forall z\ \texttt{ami\_de}(x, y) \wedge \texttt{ami\_de}(y, z) \Rightarrow \texttt{ami\_de}(x, z) \qquad (\varphi_2)$$

$$\texttt{ami\_de}(\texttt{léonne}, \texttt{maurice}) \wedge \texttt{ami\_de}(\texttt{maurice}, \texttt{henriette}) \qquad (\varphi_3)$$

$$\neg(\exists x \exists y\ \texttt{ami\_de}(x, y) \wedge \neg \texttt{connaît}(x, y)) \qquad (\varphi_4)$$

$$\exists x\ \texttt{connaît}(x, \texttt{mère}(\texttt{mère}(\texttt{père}(x)))) \qquad (\varphi_5)$$

$$\exists x\ \texttt{connaît}(x, y) \Rightarrow \texttt{connaît}(y, x) \qquad (\varphi_6)$$

φ_1 est une formalisation de « Toute personne non solitaire connaît quelqu'un. » φ_2 est une formalisation de « Les amis des amis de toute personne sont aussi ses amis. » φ_3 est une formalisation de « Léonne est amie de Maurice et Maurice est ami de Henriette. » φ_4 est une formalisation de « Il n'existe pas de personnes qui sont amis sans se connaître. » φ_5 est une formalisation de « Il existe au moins une personne qui connaît la mère de la mère de son père. » La formule φ_6 est particulière : sa vérité est dépendante de y ; on appelle cela une formule ouverte (on reviendra sur cette notion).

Dans ces formules apparaissent :

— Des ***connecteurs*** : $\neg$, $\wedge$ et $\Rightarrow$;
— Des ***variables*** : x, y et z ;
— Des ***constantes*** : `léonne`, `maurice` et `henriette` ;
— Des ***prédicats*** : le prédicat unaire `solitaire`, les prédicats binaires `ami_de` et `connaît` ;
— Des ***symboles de fonctions*** unaires : `mère` et `père` ;
— Des ***quantificateurs*** : $\forall$ (« pour tout », le ***quantificateur universel***) et $\exists$ (« il existe », le ***quantificateur existentiel***).

Définition de la syntaxe. Pour définir la syntaxe de la logique du premier ordre, on suppose qu'on dispose d'un ensemble au plus dénombrable de symboles et que cet ensemble est partitionné comme suit :

— Les variables (notées usuellement x, y, z, etc., avec éventuellement des indices) ;
— Les constantes (notées usuellement a, b, c, etc., avec éventuellement des indices, ou avec plusieurs lettres, comme `léonne`) ;
— Les prédicats (notés usuellement p, q, r, etc., avec éventuellement des indices, ou avec plusieurs lettres, comme `ami_de`), eux-mêmes partitionnés en prédicats unaires (ou d'***arité*** 1), binaires (ou d'arité 2), ternaires (ou d'arité 3), etc.,
— Les symboles de fonctions (notés usuellement f, g, h, etc., avec éventuellement des indices, ou avec plusieurs lettres, comme `mère`), eux-mêmes partitionnés en symboles de fonctions unaires (ou d'***arité*** 1), binaires (ou d'arité 2), ternaires (ou d'arité 3), etc.,

On notera $\texttt{arité}(p)$ et $\texttt{arité}(f)$ l'arité du prédicat p et du symbole de fonction f.

Un symbole de fonction est parfois appelé simplement « fonction » par abus de langage.

On peut considérer que les constantes sont des symboles de fonction d'arité 0 (écrits d'habitude sans les parenthèses : a au lieu de $a()$).

Pour définir ce qu'est une formule de $(\mathcal{L}1\mathcal{O}, \models)$, il faut d'abord définir la notion d'atome (formule non décomposable en sous-formules) laquelle s'appuie sur la notion de terme.

Un ***terme*** est une expression décrite par la grammaire suivante :

$$\texttt{<terme>} ::= \texttt{<constante>} \mid \texttt{<variable>} \mid \texttt{<symbole de fonction>}(\underbrace{\texttt{<terme>}, \ldots \texttt{<terme>}}_{n \text{ termes}})$$

où le nombre n de termes est l'arité du symbole de fonction.

Attention : un terme *n'est pas* une formule (on ne lui associe pas de valeur de vérité, mais — comme on le verra plus loin — une valeur du domaine d'interprétation).

Soit a et b des constantes, x et y des variables, f un symbole de fonction unaire et g un symbole de fonction binaire. Les expressions suivantes sont des exemples de termes :

$$a \quad x \quad f(b) \quad f(y) \quad f(f(a)) \quad g(a,y) \quad g(a,g(b,g(a,g(b,g(a,f(f(x)))))))$$

Un ***atome*** est une expression décrite par la grammaire suivante :

$$\texttt{<atome>} ::= \texttt{<prédicat>}(\underbrace{\texttt{<terme>}, \ldots \texttt{<terme>}}_{n \text{ termes}})$$

où le nombre n de termes est l'arité du prédicat.

Soit p un prédicat unaire, q un prédicat binaire et a, b, x, y, f et g, comme introduits ci-dessus. Les expressions suivantes sont des exemples d'atomes :

$$p(a) \quad p(x) \quad q(a,x) \quad q(b,f(a)) \quad q(f(g(a,x)),g(f(a),f(x)))$$

Une ***formule*** de la logique du premier ordre est une expression de la forme :

$$\begin{aligned}\texttt{<formule>} ::= {} & \texttt{<connecteur d'arité 0>} \mid \texttt{<atome>} \mid \neg\texttt{<formule>} \\ & \mid \texttt{<formule><connecteur binaire><formule>} \mid \texttt{<quantificateur><variable><formule>}\end{aligned}$$

avec les parenthésages adéquats (qui n'ont pas été rajoutés ci-dessus). Les connecteurs sont les mêmes qu'en logique propositionnelle.

Remarque 6 Convention sur les parenthésages. Pour les connecteurs, on utilise les mêmes priorités qu'en logique propositionnelle. Ainsi,

$$\neg p(x,y) \vee q(f(a)) \Rightarrow p(a,y) \wedge q(a) \quad \text{se lit} \quad ((\neg p(x,y)) \vee q(f(a))) \Rightarrow (p(a,y) \wedge q(a))$$

Pour les quantifications (tels que $\forall x$ ou $\exists y$), on suppose qu'elles sont moins prioritaires que toutes les autres opérations. Ainsi,

$$\forall x\, p(x) \Rightarrow \exists y\, q(x,y) \wedge \forall x\, r(x,y,a) \quad \text{se lit} \quad \forall x\, (p(x) \Rightarrow \exists y\, (q(x,y) \wedge \forall x\, (r(x,y,a))))$$

Autre exemple : $\forall x\, \varphi \wedge \exists x\, \psi$ se lit $\forall x\, (\varphi \wedge \exists x\, \psi)$ et non pas $(\forall x\, \varphi) \wedge (\exists x\, \psi)$.

Formules ouvertes et formules closes. Considérons la formule suivante :

$$\forall x\ \neg q(x, y)$$

la formule $\neg q(x, y)$ est la *portée* du quantificateur. De façon générale, dans les formules $\forall y\ \varphi$ et $\exists y\ \varphi$, la portée des quantificateurs est φ. Une variable dans la portée d'un quantificateur Qx ($Q \in \{\forall, \exists\}$) est dite ***variable liée*** (par le quantificateur). Sinon, c'est une ***variable libre***. Donc, dans la formule précédente, x est liée et y est libre.

Question 4 *Dans l'exemple suivant*

$$(\forall x \exists y\ r(x, y, z) \Rightarrow q(z, u)) \wedge (\neg \exists u\ p(u) \wedge \neg q(u, v))$$

quelles sont les variables libres et les variables liées ?

Considérons à présent la formule suivante :

$$\underbrace{p(x)}_{(a)} \wedge \underbrace{(\forall x\ q(x, f(x)))}_{(b)} \wedge \underbrace{(\exists x\ q(f(x), x))}_{(c)}$$

Dans les sous-formules (a), (b) et (c), la variable x joue des rôles différents : c'est une variable libre dans (a) (et dans la formule entière), une variable liée dans (b) par le quantificateur universel et une variable liée dans (c) par le quantificateur existentiel. De façon générale, on essaiera d'éviter ce genre de formules où une variable joue des rôles différents, quand cela nuira à la lisibilité. Ainsi, on pourra réécrire cette formule par exemple en :

$$p(x) \wedge (\forall y\ q(y, f(y))) \wedge (\exists z\ q(f(z), z))$$

Les deux formules sont logiquement équivalentes (du point de vue de la sémantique présentée plus loin), mais la deuxième écriture est plus lisible.

Une ***formule ouverte*** est une formule dans laquelle il y a au moins une variable libre. Sinon, c'est une ***formule close***.

Substitution d'une variable par un terme. Soit φ une formule dont x est une variable libre et t un terme. La substitution de x par t dans φ est obtenue en remplaçant dans φ chaque occurrence libre de x par t. Le résultat est noté $\varphi[x\backslash t]$. Si x n'est pas une variable libre de φ, alors $\varphi[x\backslash t] = \varphi$. Exemples :

$$(p(x, a) \wedge q(x) \wedge \exists x\ p(a, x))[x\backslash f(b)] = p(f(b), a) \wedge q(f(b)) \wedge \exists x\ p(a, x)$$
$$(\forall y\ p(x, y))[y\backslash x] = \forall y\ p(x, y)$$

Exercice 28 *À quelles conditions (portant notamment sur les arités des prédicats et les arités des symboles de fonction) les expressions suivantes sont-elles des formules (syntaxiquement correctes) de la logique du premier ordre ?*

$$\forall x\ p(x, x) \Rightarrow p(f(x), f(x))$$
$$(\neg \exists x \forall x\ q(x, x)) \vee (\neg \forall y \exists y\ q(y, y))$$
$$(\exists x\ p(x)) \Rightarrow (\forall x\ p(p(x)))$$
$$p(x, y) \wedge \wedge \neg \exists z\ r(x, y, z)$$
$$(\forall x\ p(x, y)) \Rightarrow (\exists z\ r(x, y, z))$$

Quelles sont les formules ouvertes parmi ces formules ? Pour chaque formule ouverte, indiquez quelles sont les variables libres.

Pour cet exercice, il est bon de se rappeler la convention de parenthésage utilisée dans ce cours (voir remarque 6, page 29).

Remarque 7 On peut faire un parallèle entre variables libres/liées et variables *parlantes/muettes* en mathématiques. Ainsi, dans l'expression $\sum_{i=1}^{n}\prod_{j=1}^{p}(i+j)$, les variables muettes sont i et j et les variables parlantes sont n et p. Les variables muettes sont des variables qui peuvent être remplacées par d'autres variables sans que cela change la valeur de l'expression. En $\mathcal{L}1\mathcal{O}$, les variables liées d'une formule qui sont renommées donnent des formules équivalentes.

3.3 Sémantique de $(\mathcal{L}1\mathcal{O}, \models)$

La sémantique de $(\mathcal{L}1\mathcal{O}, \models)$ décrite ci-dessous s'appuie sur la théorie des modèles. Il faut donc définir ce qu'est une interprétation $\mathcal{I}$ et ce que l'expression $\mathcal{I} \models \varphi$ ($\mathcal{I}$ satisfait φ) signifie. Une remarque importante doit être faite : cette notion de satisfiabilité d'une interprétation par une formule n'a de sens que pour les formules closes.

3.3.1 Interprétations

Une ***interprétation*** $\mathcal{I}$ est un couple $(\Delta_\mathcal{I}, \cdot^\mathcal{I})$ tel que :
— $\Delta_\mathcal{I}$ est un ensemble *non vide* quelconque, appelé le ***domaine d'interprétation*** ;
— $\cdot^\mathcal{I}$ est la ***fonction d'interprétation***, une fonction qui :
  — À une constante a associe $a^\mathcal{I} \in \Delta_\mathcal{I}$;
  — À un prédicat p d'arité n associe une relation $p^\mathcal{I}$ sur $\Delta_\mathcal{I}^n$;
  — À un symbole de fonction f d'arité n associe une fonction de $\Delta_\mathcal{I}^n$ dans $\Delta_\mathcal{I}$.

Par exemple, considérons φ la formule close définie par :

$$\varphi = \forall x\ p(x) \Rightarrow \neg p(f(x)) \tag{16}$$

Pour que cette formule soit syntaxiquement correcte, il faut supposer que p est un prédicat unaire et que f est un symbole de fonction unaire. On peut définir les deux interprétations suivantes s'appuyant sur le vocabulaire apparaissant dans cette formule (à savoir p et f) :
— $\mathcal{I} = (\Delta_\mathcal{I}, \cdot^\mathcal{I})$
  — $\Delta_\mathcal{I} = \mathbb{N}$;
  — $p^\mathcal{I}$ est la relation unaire « être pair » : $p^\mathcal{I} = \{0, 2, 4, \ldots\}$;
  — $f^\mathcal{I}$ est la fonction définie par $f^\mathcal{I}(n) = n + 1$, pour tout $n \in \mathbb{N}$.
— $\mathcal{J} = (\Delta_\mathcal{J}, \cdot^\mathcal{J})$
  — $\Delta_\mathcal{J} = \mathbb{Z}$;
  — $p^\mathcal{J}$ est la relation unaire « être impair » ;
  — $f^\mathcal{J}$ est la fonction définie par $f^\mathcal{J}(n) = n - 3$, pour tout $n \in \mathbb{Z}$.

De façon intuitive, nous allons examiner si ces interprétations sont ou pas des modèles de φ :
— $\mathcal{I} \models \varphi$ signifie intuitivement ceci : « Quel que soit $n \in \mathbb{N}$, si n est pair alors $n + 1$ ne l'est pas. »
— $\mathcal{J} \models \varphi$ signifie intuitivement ceci : « Quel que soit $n \in \mathbb{Z}$, si n est impair alors $n - 3$ ne l'est pas. »

Des connaissances élémentaires en arithmétique indiquent qu'effectivement, $\mathcal{I}$ et $\mathcal{J}$ sont des modèles de φ.

Pour aller plus loin que l'intuition, il faut définir l'interprétation d'une formule et pour cela, on reprend l'ordre utilisé dans la définition syntaxique d'une formule : terme, atome, formule quelconque.

Dans la suite de la section, on considère un vocabulaire donné et on considère un ensemble de variables fini, $\mathcal{V} = \{x_1, x_2, \ldots, x_n\}$. La numérotation des variables (de 1 à n) est fixée pour des raisons pratiques. Soit $\mathcal{I} = (\Delta_\mathcal{I}, \cdot^\mathcal{I})$, une interprétation portant sur le vocabulaire des termes et formules évoquées dans le reste de la section.

Interprétation d'un terme. Soit t un terme. Il peut contenir 0 à n variables. Donc, son interprétation « peut dépendre » de ces n variables ; il sera interprété par une *fonction* à n arguments :

$$\begin{array}{rccc} t^{\mathcal{I}} : & \Delta_{\mathcal{I}}^n & \longrightarrow & \Delta_{\mathcal{I}} \\ & (u_1, u_2, \ldots, u_n) & \longmapsto & t^{\mathcal{I}}(u_1, u_2, \ldots, u_n) \end{array}$$

Si $t = a$, où a est une constante, on assimilera la valeur $a^{\mathcal{I}} \in \Delta_{\mathcal{I}}$ (associée à la définition de l'interprétation) à une *fonction constante* : $a^{\mathcal{I}}$ est la fonction qui à $(u_1, u_2, \ldots, u_n)$ associe $a^{\mathcal{I}}$.

Si $t = x_i$ est la i^{e} variable, alors, $t^{\mathcal{I}}$ est la i^{e} projection :

$$\begin{array}{rccc} (x_i)^{\mathcal{I}} : & \Delta_{\mathcal{I}}^n & \longrightarrow & \Delta_{\mathcal{I}} \\ & (u_1, u_2, \ldots, u_n) & \longmapsto & u_i \end{array}$$

Enfin, si f est un symbole de fonction d'arité k et que t_1, t_2, …, t_k sont k termes, alors le terme $t = f(t_1, t_2, \ldots, t_k)$ est interprété en composant les interprétations :

$$\begin{array}{rccc} (f(t_1, t_2, \ldots, t_k))^{\mathcal{I}} : & \Delta_{\mathcal{I}}^n & \longrightarrow & \Delta_{\mathcal{I}} \\ & (u_1, u_2, \ldots, u_n) & \longmapsto & f^{\mathcal{I}}(t_1^{\mathcal{I}}(u_1, u_2, \ldots, u_n), t_2^{\mathcal{I}}(u_1, u_2, \ldots, u_n), \ldots, t_k^{\mathcal{I}}(u_1, u_2, \ldots, u_n)) \end{array}$$

Par exemple, si f est un symbole de fonction ternaire, a est une constante, qu'il y a 3 variables x_1, x_2 et x_3, que $\mathcal{I}$ est une interprétation dont le domaine est $\Delta_{\mathcal{I}} = \mathbb{N}$ et la fonction d'interprétation est définie par :

$$f^{\mathcal{I}} : (u_1, u_2, u_3) \in \mathbb{N}^3 \mapsto (u_1 + u_2) \times u_3 \in \mathbb{N} \qquad a^{\mathcal{I}} = 2$$

alors, le terme $f(x_1, a, x_2)$ est interprété comme suit :

$$\begin{array}{rccc} (f(x_1, a, x_2))^{\mathcal{I}} : & \mathbb{N}^3 & \longrightarrow & \mathbb{N} \\ & (u_1, u_2, u_3) & \longmapsto & (u_1 + 2) \times u_2 \end{array}$$

Interprétation d'un atome. De façon générale, une formule φ est interprétée par une fonction à n arguments (autant que de variables) qui donne une valeur booléenne en sortie :

$$\begin{array}{rccc} \varphi^{\mathcal{I}} : & \Delta_{\mathcal{I}}^n & \longrightarrow & \{\mathtt{F}, \mathtt{V}\} \\ & (u_1, u_2, \ldots, u_n) & \longmapsto & \varphi^{\mathcal{I}}(u_1, u_2, \ldots, u_n) \end{array}$$

Pour le cas particulier d'une formule atomique $p(t_1, t_2, \ldots, t_k)$ où p est un prédicat d'arité k, pour tout $(u_1, u_2, \ldots, u_n) \in \Delta_{\mathcal{I}}^n$, l'interprétation de cet atome en ce n-uplet vaut vrai si $t_1^{\mathcal{I}}(u_1, u_2, \ldots, u_n)$, $t_2^{\mathcal{I}}(u_1, u_2, \ldots, u_n)$, … $t_k^{\mathcal{I}}(u_1, u_2, \ldots, u_n)$ sont liés par la relation $p^{\mathcal{I}}$ (d'arité k) :

$$\begin{array}{rccc} (p(t_1, t_2, \ldots, t_k))^{\mathcal{I}} : & \Delta_{\mathcal{I}}^n & \longrightarrow & \{\mathtt{F}, \mathtt{V}\} \\ & (u_1, u_2, \ldots, u_n) & \longmapsto & \begin{cases} \mathtt{V} & \text{si } p^{\mathcal{I}}(t_1^{\mathcal{I}}(u_1, u_2, \ldots, u_n), t_2^{\mathcal{I}}(u_1, u_2, \ldots, u_n), \\ & \qquad \ldots, t_k^{\mathcal{I}}(u_1, u_2, \ldots, u_n)) \\ \mathtt{F} & \text{sinon} \end{cases} \end{array}$$

Par exemple, si on poursuit l'exemple précédent en ajoutant l'interprétation du prédicat binaire p par $p^{\mathcal{I}} = >$, on aura l'interprétation de l'atome $p(f(a, x_2, x_3), x_1)$:

$$\begin{array}{rccc} (p(f(a, x_2, x_3), x_1))^{\mathcal{I}} : & \Delta_{\mathcal{I}}^3 & \longrightarrow & \{\mathtt{F}, \mathtt{V}\} \\ & (u_1, u_2, u_3) & \longmapsto & \begin{cases} \mathtt{V} & \text{si } (2 + u_2) \times u_3 > u_1 \\ \mathtt{F} & \text{sinon} \end{cases} \end{array}$$

Interprétation d'une formule non atomique. On considérera d'abord les connecteurs $\neg$, $\wedge$ et $\vee$, auxquels on associe les opérations booléennes correspondantes (**non**, **et** et **ou**).

Soit $\varphi, \varphi_1, \varphi_2 \in \mathcal{L}1\mathcal{O}$. Les formules $\neg\varphi$, $\varphi_1 \wedge \varphi_2$ et $\varphi_1 \vee \varphi_2$ sont interprétées par les fonctions $(\neg\varphi)^{\mathcal{I}}$, $(\varphi_1 \wedge \varphi_2)^{\mathcal{I}}$ et $(\varphi_1 \vee \varphi_2)^{\mathcal{I}}$ définies, pour $(u_1, u_2, \ldots, u_n) \in \Delta_{\mathcal{I}}^n$, par

$$\begin{aligned}(\neg\varphi)^{\mathcal{I}}(u_1, u_2, \ldots, u_n) &= \textbf{non}(\varphi^{\mathcal{I}}(u_1, u_2, \ldots, u_n))\\ (\varphi_1 \wedge \varphi_2)^{\mathcal{I}}(u_1, u_2, \ldots, u_n) &= \varphi_1^{\mathcal{I}}(u_1, u_2, \ldots, u_n) \textbf{ et } \varphi_2^{\mathcal{I}}(u_1, u_2, \ldots, u_n)\\ (\varphi_1 \vee \varphi_2)^{\mathcal{I}}(u_1, u_2, \ldots, u_n) &= \varphi_1^{\mathcal{I}}(u_1, u_2, \ldots, u_n) \textbf{ ou } \varphi_2^{\mathcal{I}}(u_1, u_2, \ldots, u_n)\end{aligned}$$

Pour poursuivre l'exemple précédent, on a :

$$(\neg p(a, x_1) \wedge p(f(a, x_2, x_3), a))^{\mathcal{I}}(u_1, u_2, u_3) = \textbf{non}(2 > u_1) \textbf{ et } (2 + u_2) \times u_3 > 2$$

Le connecteur $\top$, d'arité 0, peut être défini par $\top \equiv \varphi \vee \neg\varphi$ où φ est une formule quelconque. Par conséquent, $\top^{\mathcal{I}}$ est la fonction constante donnant la valeur `V` (puisque α **ou non** α = `V`, pour tout $\alpha \in \{\mathtt{F}, \mathtt{V}\}$). De même $\bot^{\mathcal{I}}$ est la fonction constante à `F` : on définit $\bot$ par $\varphi \wedge \neg\varphi$.

On peut définir de la même façon les autres connecteurs binaires comme étant des raccourcis de notation de la même façon qu'en logique propositionnelle. Par exemple, $\varphi_1 \Rightarrow \varphi_2$, peut être considéré comme un raccourci pour $\neg\varphi_1 \vee \varphi_2$, d'où :

$$(\varphi_1 \Rightarrow \varphi_2)^{\mathcal{I}}(u_1, u_2, \ldots, u_n) = \textbf{non } \varphi_1^{\mathcal{I}}(u_1, u_2, \ldots, u_n) \textbf{ ou } \varphi_2^{\mathcal{I}}(u_1, u_2, \ldots, u_n)$$

Enfin, considérons les formules Qx_i où Q est un quantificateur. Les quantificateurs *lient* les variables : $(Qx_i\ \varphi)^{\mathcal{I}}$ est une fonction dont la valeur ne change pas en changeant le i^{e} argument : $(Qx_i\ \varphi)^{\mathcal{I}}(u_1, \ldots, u_i, \ldots, u_n) = (Qx_i\ \varphi)^{\mathcal{I}}(u_1, \ldots, u_i', \ldots, u_n)$, quels que soient $u_1, \ldots, u_i, \ldots, u_n, u_i' \in \Delta_{\mathcal{I}}$. La conséquence de cela est qu'une formule sans variable libre (i.e., close) est interprétée par une fonction constante (valant toujours `F` ou toujours `V`).

La formule $\forall x_i\ \varphi$ s'interprète comme suit. $(\forall x_i\ \varphi)^{\mathcal{I}}(u_1, \ldots, u_i, \ldots, u_n)$ vaudra `V` si, quelle que soit la valeur $u_i' \in \Delta_{\mathcal{I}}$, $\varphi^{\mathcal{I}}(u_1, \ldots, u_i', \ldots, u_n) = \mathtt{V}$.

La formule $\exists x_i\ \varphi$ s'interprète comme suit. $(\forall x_i\ \varphi)^{\mathcal{I}}(u_1, \ldots, u_i, \ldots, u_n)$ vaudra `V` s'il existe au moins un $u_i' \in \Delta_{\mathcal{I}}$ tel que $\varphi^{\mathcal{I}}(u_1, \ldots, u_i', \ldots, u_n) = \mathtt{V}$.

Pour poursuivre l'exemple suivi, on a :

$$(\forall x_1 \neg p(a, f(a, x_1, x_2)))^{\mathcal{I}}(u_1, u_2, u_3) = \begin{cases}\mathtt{V} & \text{si pour tout } u_1' \in \mathbb{N},\ 2 \leqslant (2 + u_1') \times u_2\\ \mathtt{F} & \text{sinon}\end{cases}$$

$$(\exists x_2 \forall x_1 \neg p(a, f(a, x_1, x_2)))^{\mathcal{I}}(u_1, u_2, u_3) = \mathtt{V} \text{ car, en prenant par exemple } u_2 = 1,\ 2 \leqslant (2 + u_1') \times u_2 \text{ quel que soit } u_1' \in \mathbb{N}$$

Question 5 *Soit φ, une formule close et $\mathcal{I}$, une interprétation. $\varphi^{\mathcal{I}}$ est donc une fonction constante de valeur* `V` *ou de valeur* `F`. *Si $\mathcal{I} \not\models \varphi$ alors $\mathcal{I} \models \neg\varphi$. Pourquoi ?*

3.3.2 Satisfaction d'une formule close par une interprétation et relation de conséquence logique

Satisfaction d'une formule close par une interprétation. Soit $\mathcal{I}$ une interprétation. Une formule close φ s'interprète donc comme une fonction *constante* $\varphi^{\mathcal{I}}$. $\mathcal{I}$ satisfait φ — noté $\mathcal{I} \models \varphi$ — si cette constante est `V`.

Relation $\models$ entre formules closes. D'où la relation $\varphi \models \psi$ entre deux formules closes ($\varphi \models \psi$ si $\mathcal{I} \models \varphi$ entraîne $\mathcal{I} \models \psi$), la relation $B \models \varphi$ entre une base de connaissances ne contenant que des formules closes et une formule close, la satisfiabilité d'une formule close, l'équivalence entre formules closes $\varphi \equiv \psi$ (si $\varphi \models \psi$ et $\psi \models \varphi$, i.e., $\mathcal{I}$ est un modèle de φ ssi $\mathcal{I}$ est un modèle de ψ).

La suite de ce paragraphe concerne les formules quelconques, contenant d'éventuelles variables libres.

Relation $\models$ entre formules quelconques. Les notions de conséquence logique et d'équivalence logique ($\models$ et $\equiv$) s'étendent à toutes formules de la façon suivante. Soit $\varphi, \psi \in \mathcal{L}$, n étant toujours le nombre de variables. Intuitivement, $\varphi \models \psi$ si « à chaque fois que φ est vérifié, ψ est vérifié ». On peut formaliser cela comme suit :

$$\varphi \models \psi \quad \text{si} \quad \left| \begin{array}{l} \text{Pour toute interprétation } \mathcal{I} \text{ et tout } (u_1, u_2, \ldots, u_n) \in \Delta_{\mathcal{I}}^n, \\ \text{si } \varphi^{\mathcal{I}}(u_1, u_2, \ldots, u_n) = \mathtt{V} \text{ alors } \psi^{\mathcal{I}}(u_1, u_2, \ldots, u_n) = \mathtt{V} \end{array} \right.$$

On dit alors que $\varphi \equiv \psi$ si $\varphi \models \psi$ et $\psi \models \varphi$, ce qui revient à dire que les deux fonctions $\varphi^{\mathcal{I}}$ et $\psi^{\mathcal{I}}$ sont égales.

Par exemple, une ***tautologie*** est une formule qui s'interprète comme la formule vraie en tout $(u_1, u_2, \ldots, u_n)$. Il existe des tautologies qui ne sont pas des formules closes, comme, par exemple $p(x, y) \vee \neg p(x, y)$.

Remarque 8 Cette définition étend bien la précédente puisque si φ et ψ sont des formules closes, $\varphi \models \psi$ a le même sens dans les deux définitions (celle restreinte aux formules closes et celle pour toute formule).

Par exemple, on a $\varphi \models \varphi \vee \psi$. En effet, si $\mathcal{I}$ est une interprétation, $(u_1, u_2, \ldots, u_n) \in \Delta_{\mathcal{I}}^n$ et que $\varphi^{\mathcal{I}}(u_1, u_2, \ldots, u_n) = \mathtt{V}$, alors $(\varphi \vee \psi)^{\mathcal{I}}(u_1, u_2, \ldots, u_n) = \varphi^{\mathcal{I}}(u_1, u_2, \ldots, u_n)$ **ou** $\psi^{\mathcal{I}}(u_1, u_2, \ldots, u_n) = \mathtt{V}$ **ou** $\psi^{\mathcal{I}}(u_1, u_2, \ldots, u_n) = \mathtt{V}$.

Remarque 9 Une autre façon de présenter cela consisterait à considérer qu'une interprétation est un couple $\mathcal{I} = (\Delta_{\mathcal{I}}, \cdot^{\mathcal{I}})$ comme précédemment, sauf qu'on y ajouterait l'interprétation des variables *libres* par des valeurs de $\Delta_{\mathcal{I}}$.

Clôture universelle et conséquence logique. Une autre façon de considérer la relation $\models$ entre formules s'appuie sur la notion de ***clôture universelle***. Étant donné une formule φ dont les variables libres sont y_1, y_2, …, y_p, la clôture universelle de φ est la formule close $\forall y_1 \forall y_2 \ldots \forall y_p \, \varphi$ ([11]). Notons $\overrightarrow{\forall}\varphi$ la clôture universelle de φ.

On peut montrer que si $\varphi \models \psi$ alors $\overrightarrow{\forall}\varphi \models \overrightarrow{\forall}\psi$. Par conséquent, si $\varphi \equiv \psi$ alors $\overrightarrow{\forall}\varphi \equiv \overrightarrow{\forall}\psi$. En revanche, la réciproque est fausse, par exemple :

$$p(x, y) \not\models p(y, x) \qquad \text{alors que} \qquad \forall x \forall y \, p(x, y) \models \forall x \forall y \, p(y, x)$$

Or, il arrive souvent qu'une formule ouverte φ soit implicitement considérée comme quantifiée universellement sur ses variables libres (cf. section 3.7.5). Par conséquent, il convient de bien savoir si, quand on a une formule ouverte, il faut considérer sa clôture universelle à sa place ou s'il faut la considérer telle qu'elle.

3.3.3 Exemples

Reprenons les exemples d'interprétations $\mathcal{I}$ et $\mathcal{J}$ pour la formule φ introduite par l'équation (16) : lesquelles de ces interprétations satisfont φ ?

Pour l'interprétation $\mathcal{I}$ et $u \in \mathbb{N}$, on a :

$$\begin{aligned}
x^{\mathcal{I}}(u) &= u \\
(p(x))^{\mathcal{I}}(u) &= u \text{ est pair} \\
(f(x))^{\mathcal{I}}(u) &= u + 1 \\
(p(f(x)))^{\mathcal{I}}(u) &= u + 1 \text{ est pair} \\
(\neg p(f(x)))^{\mathcal{I}}(u) &= \textbf{non}(u + 1 \text{ est pair}) = u + 1 \text{ est impair} \\
(p(x) \Rightarrow \neg p(f(x)))^{\mathcal{I}}(u) &= \textbf{non}(u \text{ est pair}) \textbf{ ou } u + 1 \text{ est impair} = u \text{ est impair } \textbf{ou } u + 1 \text{ est impair} \\
(\forall x \, p(x) \Rightarrow \neg p(f(x)))^{\mathcal{I}}(u) &= \begin{cases} \mathtt{V} & \text{si, pour tout } u \in \mathbb{N}, u \text{ est impair } \textbf{ou } u + 1 \text{ est impair} \\ \mathtt{F} & \text{sinon} \end{cases} = \mathtt{V}
\end{aligned}$$

11. On devrait dire *une* clôture universelle de φ puisque la formule résultante dépend de l'ordre des variables. Cependant, on verra que changer cet ordre donne une formule équivalente : cf. équivalence (19) un peu plus loin.

Donc $\mathcal{I} \models \varphi$.

Pour l'interprétation $\mathcal{J}$ et $u \in \mathbb{Z}$, on a :

$$(\forall x\ p(x) \Rightarrow \neg p(f(x)))^{\mathcal{J}}(u) = \begin{cases} \mathtt{V} & \text{si, pour tout } u \in \mathbb{Z},\ u \text{ est pair } \textbf{ou } u-3 \text{ est pair} \\ \mathtt{F} & \text{sinon} \end{cases} = \mathtt{V}$$

Donc $\mathcal{J} \models \varphi$.

Exercice 29 *Pour chacune des formules closes φ suivantes, donnez un modèle de φ (si φ est satisfiable) et un modèle de $\neg\varphi$ (si φ n'est pas une tautologie) :*

$$\forall x \exists y\ p(x,y)$$
$$(\forall x\ p(x)\ \Rightarrow p(f(x))) \Rightarrow (\forall x\ p(x) \Rightarrow p(f(f(x))))$$
$$\exists x\ \forall y\ \forall z\ q(f(x,y),z)$$
$$(\forall x\ p(x)) \Rightarrow (\exists x\ p(f(x)))$$

Exercice 30 *Pour chaque couple (φ, ψ) de formules ouvertes ci-dessous, a-t-on $\varphi \models \psi$? Justifiez une réponse positive par une preuve et une réponse négative par un contre-exemple.*

$\varphi = p(x,y) \wedge p(y,a)$	$\psi = p(x,y)$
$\varphi = \neg p(x,y)$	$\psi = \neg p(a,y)$
$\varphi = \exists x\ p(x,y)$	$\psi = \exists y\ p(x,y)$

3.4 Quelques résultats utiles

Les résultats suivants du chapitre sur la logique propositionnelle s'appliquent de la même façon en logique du premier ordre :

- Le théorème de la déduction et son corollaire ;
- Le lien entre tautologie et satisfiabilité et son corollaire ;
- Le fait qu'on puisse assimiler une base de connaissances à la formule constituée d'une conjonction des éléments de cette base ;
- Les équivalences (1) à (15) où φ, ψ et χ sont des formules ouvertes ou closes de la logique du premier ordre.

Les résultats suivants sont liés aux quantificateurs :

$$\neg\forall x\ \varphi \quad \equiv \quad \exists x\ \neg\varphi \tag{17}$$
$$\neg\exists x\ \varphi \quad \equiv \quad \forall x\ \neg\varphi \tag{18}$$
$$\forall x \forall y\ \varphi \quad \equiv \quad \forall y \forall x\ \varphi \tag{19}$$
$$\exists x \exists y\ \varphi \quad \equiv \quad \exists y \exists x\ \varphi \tag{20}$$
$$\forall x\ (\varphi \wedge \psi) \quad \equiv \quad (\forall x\ \varphi) \wedge (\forall x\ \psi) \tag{21}$$
$$\exists x\ (\varphi \vee \psi) \quad \equiv \quad (\exists x\ \varphi) \vee (\exists x\ \psi) \tag{22}$$
$$\forall x\ \varphi \quad \models \quad \exists x\ \varphi \tag{23}$$

Les équivalences (17) et (18) montrent qu'on peut se passer d'un des deux quantificateurs sans perdre en expressivité.

Les équivalences (19) et (20) montrent qu'on peut inverser l'ordre entre quantificateurs de même type qui se suivent. En revanche, en général, on ne peut pas inverser deux quantificateurs de types différents sans briser l'équivalence ! À titre de contre-exemple, considérons les formules :

$$\varphi = \forall x \exists y\ p(x,y) \qquad \psi = \exists y \forall x\ p(x,y)$$

et l'interprétation

$$\mathcal{I} = (\Delta_\mathcal{I}, \cdot^\mathcal{I}) \qquad \text{avec } \Delta_\mathcal{I} = \mathbb{N} \quad \text{et } p^\mathcal{I} : (u,v) \in \mathbb{N}^2 \mapsto (u = v) \in \{\mathtt{F}, \mathtt{V}\}$$

On a $\mathcal{I} \models \varphi$ et $\mathcal{I} \not\models \psi$. En revanche, on a :

$$\exists x \forall y\ \varphi \qquad \models \qquad \forall y \exists x\ \varphi \tag{24}$$

Les équivalences (21) et (22) se retiennent si on interprète un $\forall$ comme un $\wedge$ itéré et un $\exists$ comme un $\vee$ itéré. En effet, si $\mathcal{I}$ est une interprétation dans le domaine est fini — $\Delta_\mathcal{I} = \{u_1, u_2, \ldots, u_c\}$ — et que pour chaque constante a_k ($1 \leqslant k \leqslant c$), $a_k^\mathcal{I} = u_k$, alors :

$$\begin{array}{lll} \mathcal{I} \models \forall x\ \varphi & \text{ssi} & \mathcal{I} \models \varphi[x\backslash a_1] \wedge \varphi[x\backslash a_2] \wedge \ldots \wedge \varphi[x\backslash a_c] \\ \mathcal{I} \models \exists x\ \varphi & \text{ssi} & \mathcal{I} \models \varphi[x\backslash a_1] \vee \varphi[x\backslash a_2] \vee \ldots \vee \varphi[x\backslash a_c] \end{array}$$

La conséquence logique (23) semble naturelle : si φ est vérifiée quelle que soit la valeur de x, elle le sera aussi pour une valeur particulière. Notons cependant que ce résultat s'appuie sur le fait que le domaine d'interprétation *doit* être non vide.

À titre d'exemple, montrons (24) dans l'hypothèse où φ n'a pas d'autres variables libres que x et y (c'est vrai aussi sinon mais un brin plus fastidieux à montrer). Soit $\mathcal{I} = (\Delta_\mathcal{I}, \cdot^\mathcal{I})$, un modèle de $\exists x \forall y\ \varphi$. Il existe donc $u_1^0 \in \Delta_\mathcal{I}$ tel que pour tout $u_2 \in \Delta_\mathcal{I}$ on a $\varphi^\mathcal{I}(u_1^0, u_2) = \mathtt{V}$. Pour $u_2 \in \Delta_\mathcal{I}$, il existe donc $u_1 \in \Delta_\mathcal{I}$ tel que $\varphi^\mathcal{I}(u_1, u_2) = \mathtt{V}$: il suffit de prendre $u_1 = u_1^0$. Donc, $(\forall y \exists x\ \varphi)^\mathcal{I}(u_1, u_2) = \mathtt{V}$ pour tout $(u_1, u_2) \in \Delta_\mathcal{I}^2$. D'où $\mathcal{I} \models \forall y \exists x\ \varphi$, ce qui finit la preuve.

Exercice 31 *Choisissez au hasard au moins* 2 *résultats à montrer entre (17) et (23).*

En appliquant le même principe que dans l'exemple ci-dessus, montrez ces résultats dans l'hypothèse où les formules de part et d'autre des signes $\equiv$ *et* $\models$ *sont closes.*

3.5 Un système formel correct et complet pour $(\mathcal{L}1\mathcal{O}, \models)$

Les fbf sont les formules de la logique du premier ordre avec les restrictions syntaxiques suivantes :

— Les seuls connecteurs autorisés sont $\neg$ et $\Rightarrow$ (on ne perd pas d'expressivité car les autres connecteurs peuvent être réécrits en utilisant ces connecteurs, par exemple $\alpha \wedge \beta \equiv \neg(\alpha \Rightarrow \neg\beta)$).

— Le seul quantificateur autorisé est $\forall$ (on ne perd pas d'expressivité car les formules $\exists x\ \alpha$ peuvent être remplacées par les formules logiquement équivalentes $\neg\forall x\ \neg\alpha$).

Les axiomes sont définis par les cinq schémas d'axiomes suivants (où α, β, γ sont des fbf, δ est une fbf dont x n'est pas variable libre et t est un terme) :

$$\alpha \Rightarrow (\beta \Rightarrow \alpha) \tag{SA1}$$

$$(\alpha \Rightarrow (\beta \Rightarrow \gamma)) \Rightarrow ((\alpha \Rightarrow \beta) \Rightarrow (\alpha \Rightarrow \gamma)) \tag{SA2}$$

$$(\neg\alpha \Rightarrow \neg\beta) \Rightarrow (\beta \Rightarrow \alpha) \tag{SA3}$$

$$(\forall x\ \alpha) \Rightarrow \alpha[x\backslash t] \tag{SA4}$$

$$(\delta \Rightarrow \alpha) \Rightarrow (\delta \Rightarrow \forall x\ \alpha) \tag{SA5}$$

On notera que les trois premiers schémas d'axiomes sont ceux du système formel de la logique propositionnelle présenté au chapitre précédent.

La première règle d'inférence est le *modus ponens* (comme en logique propositionnelle) :

$$\frac{\alpha \qquad \alpha \Rightarrow \beta}{\beta} mp \qquad \text{pour } \alpha, \beta, \text{ deux fbf}$$

La seconde règle d'inférence est appelée *généralisation* :

$$\frac{\alpha}{\forall x\ \alpha} g \qquad \text{pour } \alpha \text{ une fbf et } x \text{ une variable}$$

Par exemple, on peut prouver $\forall x \forall y\ p(x,y) \vdash^S \forall z\ p(z,z)$ de la façon suivante, où S est le système formel décrit ci-dessus (repris du livre de J.-P. Delahaye cité en introduction) :

Nom de la fbf	fbf	explications
f_1	$\forall x \forall y\ p(x,y)$	hypothèse
f_2	$(\forall x \forall y\ p(x,y)) \Rightarrow \forall y\ p(z,y)$	axiome (cf. (SA4))
f_3	$\forall y\ p(z,y)$	*modus ponens*
f_4	$(\forall y\ p(z,y)) \Rightarrow p(z,z)$	axiome (cf. (SA4))
f_5	$p(z,z)$	*modus ponens*
f_6	$\forall z\ p(z,z)$	généralisation

3.6 Note sur la déduction naturelle en logique du premier ordre

La déduction naturelle pour la logique propositionnelle décrite en section 2.6 s'étend à la logique du premier ordre, mais ne sera pas décrite dans ce document.

3.7 Aspects algorithmiques

3.7.1 Indécidabilité, semi-décidabilité

Un ***problème de décision*** est un problème algorithmique dont la sortie est un booléen. Un tel problème est ***décidable*** s'il existe un algorithme permettant, quelles que soient les valeurs des paramètres d'entrée, de donner une réponse correcte en un temps fini. Par exemple le problème de décision « Un tableau d'entier est-il trié ? » est décidable.

Il existe des problèmes indécidables : il n'existe aucun algorithme pour les résoudre en temps fini (et pas seulement aucun algorithme connu). L'un d'eux est connu sous le nom du problème d'arrêt des machines de Turing. On peut le reformuler ainsi : « Étant donné un algorithme et des paramètres de cet algorithme, est-ce que l'exécution de cet algorithme prendra un temps fini ? » Ce problème est indécidable : aucun algorithme ne peut effectuer ce test.

Parmi les problèmes indécidables, il y a les ***problèmes semi-décidables*** : ce sont les problèmes pour lesquels il existe un algorithme qui, si la réponse est positive (`V`) donne un résultat dans un temps fini (mais peuvent « boucler » indéfiniment dans le cas contraire).

Question 6 *Le problème d'arrêt des machines de Turing est-il semi-décidable ? Pourquoi ?*

3.7.2 La logique du premier ordre est semi-décidable

Par extension, on dit qu'une logique est décidable/semi-décidable/indécidable si l'inférence $B \models \varphi$ l'est. Par exemple, la logique propositionnelle est décidable. En revanche, la logique du premier ordre est semi-décidable.

Une preuve de ce résultat s'inspire de la méthode de la diagonale de Cantor (cf. section 2.1.2).

Question 7 *Que peut-on proposer comme algorithme qui, à la base de connaissances B de $(\mathcal{L}1\mathcal{O}, \models)$ et à $\varphi \in \mathcal{L}1\mathcal{O}$ associe au bout d'un temps fini la valeur* `V` *si $B \models \varphi$ et boucle indéfiniment dans le cas contraire ?*

L'algorithme en question n'est pas tenu d'être efficace ; il peut même être terriblement lent !

Les algorithmes pour réaliser les inférences en logique du premier ordre :
— Soit ne terminent pas toujours ;
— Soit sont incomplets (peuvent finir sur un résultat qui s'interprète comme « Je ne sais pas. ») ;
— Soit s'appliquent sur des ***fragments décidables*** de $(\mathcal{L}1\mathcal{O}, \models)$.

3.7.3 Transformations de formules

Littéraux, FNN, FNC, FND. Un ***littéral*** est une formule de la forme A (littéral positif) ou $\neg A$ (littéral négatif), où A est un atome.

Les mises sous forme normale négative, forme normale conjonctive et forme normale disjonctive sont similaires à celles effectuées en logique propositionnelle pour les formules sans quantificateurs : la différence est que la notion de littéral n'est pas la même.

Pour les formules avec quantificateurs, on peut d'abord les mettre sous forme prénexe (avec les quantificateurs en tête : voir ci-dessous) puis traiter la sous-formule obtenue en enlevant les quantificateurs pour la mettre, selon les besoins, en FNN, FNC ou FND.

Mise sous forme prénexe. Une formule est sous forme prénexe si les quantificateurs apparaissent tous en tête, autrement écrit, φ est sous forme prénexe si $\varphi = Q_1 x_1 Q_2 x_2 \dots Q_p x_p \ \psi$ où $Q_i \in \{\forall, \exists\}$, x_i est une variable $(1 \leqslant i \leqslant n)$ et ψ est une expression sans quantificateur.

Pour mettre une formule sous forme prénexe, on ne considérera que les connecteurs $\neg$, $\wedge$, $\vee$ et $\Rightarrow$ (on aurait d'ailleurs pu se contenter d'un ensemble plus petit, par exemple, $\{\neg, \Rightarrow\}$) et on utilise, de gauche à droite les équivalences suivantes, où x est une variable, φ est une formule et ψ est une formule pour laquelle x n'est pas une variable libre :

$$\begin{array}{rcl@{\qquad}rcl}
\neg\forall x\ \varphi & \equiv & \exists x\ \neg\varphi & \neg\exists x\ \varphi & \equiv & \forall x\ \neg\varphi \\
(\forall x\ \varphi) \wedge \psi & \equiv & \forall x\ (\varphi \wedge \psi) & (\exists x\ \varphi) \wedge \psi & \equiv & \exists x\ (\varphi \wedge \psi) \\
(\forall x\ \varphi) \vee \psi & \equiv & \forall x\ (\varphi \vee \psi) & (\exists x\ \varphi) \vee \psi & \equiv & \exists x\ (\varphi \vee \psi) \\
(\forall x\ \varphi) \Rightarrow \psi & \equiv & \exists x\ (\varphi \Rightarrow \psi) & \psi \Rightarrow (\forall x\ \varphi) & \equiv & \forall x\ (\psi \Rightarrow \varphi) \\
(\exists x\ \varphi) \Rightarrow \psi & \equiv & \forall x\ (\varphi \Rightarrow \psi) & \psi \Rightarrow (\exists x\ \varphi) & \equiv & \exists x\ (\psi \Rightarrow \varphi)
\end{array}$$

Par ailleurs, on utilisera la commutativité de $\wedge$ et $\vee$ modulo l'équivalence.

Si on a, par exemple, une formule $(\forall x\ \varphi) \wedge \psi$ dans laquelle x *est* une variable libre, on renomme d'abord le x du quantificateur en une variable qui n'apparaît pas dans la formule ψ et on fait la substitution de x par cette variable dans φ. Ainsi :

$$(\forall x\ p(x, y)) \wedge q(x) \equiv (\forall z\ p(z, y)) \wedge q(x) \equiv \forall z\ (p(z, y) \wedge q(x))$$

Exercice 32 *Pour chacune des formules suivantes, mettez-les sous forme prénexe puis, mettez la partie hors quantificateur sous FNC puis sous FND :*

$$(\forall x\ p(x, y)) \Rightarrow (\exists z\ q(z))$$
$$p(x) \wedge (\forall x\ (\neg r(x, y, z) \vee \exists z\ s(x, z)))$$

Exercice 33 *Soit la formule sous forme prénexe $\varphi = Q_1 x_1 Q_2 x_2 \dots Q_p x_p\ \psi$ où les Q_i sont des quantificateurs et ψ est une formule sans quantificateur.*

Comment mettre $\neg\varphi$ sous forme prénexe ?

Skolémisation d'une formule. L'***équi-satisfiabilité*** de deux formules closes est la relation disant que si l'une est satisfiable, l'autre l'est aussi et réciproquement. Si deux formules φ et ψ sont équivalentes, elles sont équi-satisfiables, mais la réciproque est fausse. Considérons par exemple :

$$\varphi = \forall x\ p(f(x)) \qquad \psi = \forall x\ p(g(x))$$

φ et ψ ne sont pas équivalentes. En revanche, si $\mathcal{I}$ est un modèle de φ, on peut construire un modèle $\mathcal{J}$ de ψ en remplaçant dans la définition de $\mathcal{I}$ f par g. Et on peut faire l'opération récriproque à partir d'un modèle de ψ. Ces deux formules non équivalentes sont donc équi-satisfiables.

Question 8 *Quel modèle de φ peut-on donner ? Comment transformer ce modèle en un modèle de ψ ?*

L'équi-satisfiabilité peut être utile dans une ***preuve réfutationnelle***. Une telle preuve consiste à remplacer un test « $\models \varphi$? » par un test « $\neg\varphi$ est-elle insatisfiable ? ». Si $\neg\varphi$ et une formule χ sont équi-satisfiables, la preuve de $\models \varphi$ peut se faire par la preuve de l'instatisfiabilité de χ (laquelle peut se ramener à « $\models \neg\chi$? »).

La ***skolémisation*** est une transformation d'une formule *prénexe* φ en une formule *prénexe* φ' équi-satisfiable (mais, généralement, pas équivalente) et ne contenant pas de quantificateur existentiel. Cette transformation s'accompagne de nouveaux symboles de fonction appelés les *fonctions de Skolem* [12] : une fonction de Skolem est introduite à chaque disparition d'un $\exists$. L'exemple ci-dessous a pour but de faire passer l'intuition :

$$\forall x\ \exists y\ q(x,y) \wedge p(y) \qquad \text{est skolémisée en} \qquad \forall x\ q(x,s(x)) \wedge p(s(x))$$

Dans la seconde formule, s est une fonction de Skolem d'arité 1. On observe que la quantification $\exists y$ est supprimée et que toutes les occurrences de y dans la portée de ce quantificateur sont remplacées par $s(x)$. Quand on a, comme ci-dessus, un $\exists y$ dans la portée d'un $\forall x$, l'idée est que ce y est « dépendant » du x et cette dépendance se traduit dans la skolémisation par le remplacement de y par une fonction de x.

De façon plus générale, la skolémisation d'une formule prénexe φ consiste, pour chaque quantification existentielle $\exists z$ de la formule :

— À identifier les k quantifications universelles $\forall x_1, \ldots, \forall x_k$ qui *précèdent* $\exists z$;
— À introduire un nouveau symbole de fonction [13] s d'arité k ;
— À remplacer toutes les occurrences de z par $s(x_1, \ldots, x_k)$;
— À supprimer la quantification $\exists z$.

On considérera ici qu'une constante est un symbole de fonction d'arité 0.

Voici ce que cela donne sur un exemple :

$$\exists x \forall y \exists z \forall u \forall v \exists w\ p(x,y) \wedge p(y,z) \wedge r(z,u,v) \wedge r(x,y,w)$$
$$\text{est skolémisée en}$$
$$\forall y \forall u \forall v\ p(s_1,y) \wedge p(y,s_2(y)) \wedge r(s_2(y),u,v) \wedge r(s_1,y,s_3(y,u,v))$$

(ici s_1 est une constante).

Exercice 34 *Skolémisez les formules de l'exercice 32.*

Soit φ une formule sous forme prénexe et σ une formule obtenue par skolémisation de φ. On a les résultats suivants :

— φ et σ sont équi-satisfiables.

12. Thoralf Skolem, mathématicien et logicien norvégien (1887-1963).
13. « Nouveau » signifie ici un symbole non utilisée par ailleurs dans l'énoncé (ni dans la formule courante, ni dans aucune autre utilisée dans l'énoncé, p. ex., dans aucune formule d'une base de connaissances utilisée dans l'énoncé).

— Si $\mathcal{I} \models \sigma$ alors $\mathcal{I} \models \varphi$ (et donc, $\sigma \models \varphi$).
— Si $\mathcal{I} \models \varphi$ alors il existe une interprétation $\mathcal{I}'$ telle que $\mathcal{I}' \models \sigma$.

Donnons un exemple du dernier résultat. Soit :

$$\varphi = \forall x \exists y\ p(x, y) \qquad \sigma = \forall x\ p(x, s(x))$$

σ est obtenue par skolémisation de φ. Soit $\mathcal{I} = (\Delta_{\mathcal{I}}, \cdot^{\mathcal{I}})$, l'interprétation définie par :

$$\Delta_{\mathcal{I}} = \mathbb{N} \qquad p^{\mathcal{I}} : (u, v) \in \mathbb{N}^2 \mapsto \begin{cases} \mathtt{V} & \text{si } u < v \\ \mathtt{F} & \text{sinon} \end{cases}$$

$\mathcal{I}$ satisfait φ. En effet, pour tout $u \in \mathbb{N}$, on peut toujours trouver un $v \in \mathbb{N}$ qui soit strictement plus grand que u, par exemple $v = u + 1$ (pour prendre le plus petit). Soit alors la fonction $f : u \in \mathbb{N} \mapsto u + 1 \in \mathbb{N}$: f associe à u un v tel que $p^{\mathcal{I}}(u, v) = \mathtt{V}$. Cela permet de décrire l'interprétation $\mathcal{I}' = (\Delta_{\mathcal{I}'}, \cdot^{\mathcal{I}'})$ qui satisfait σ en reprenant la définition de $\mathcal{I}$ et en ajoutant l'interprétation de la fonction de Skolem :

$$\Delta_{\mathcal{I}'} = \Delta_{\mathcal{I}} = \mathbb{N} \qquad p^{\mathcal{I}'} = p^{\mathcal{I}} \qquad s^{\mathcal{I}'} = f$$

Pour prouver qu'une formule φ est une tautologie, il suffit de prouver que $\neg\varphi$ est insatisfiable. Si σ est obtenue par skolémisation de $\neg\varphi$, prouver $\models \varphi$ revient à prouver que σ est insatisfiable.

3.7.4 La méthode des tableaux sémantiques en logique du premier ordre

⚠Cette section n'est qu'un aperçu de la méthode des tableaux sémantiques en logique du premier ordre.

Cette méthode s'apparente à celle de même nom en logique propositionnelle. Elle permet de tester si un ensemble de formules B de $(\mathcal{L}1\mathcal{O}, \models)$ est satisfiable. Il est important de noter qu'elle ne constitue *pas* un algorithme : il faut dans certains cas « deviner » (voir la règle $\rightarrow_\forall$). En effet, avoir un algorithme qui termine dans tous les cas avec une réponse correcte serait en contradiction avec l'indécidabilité de cette logique.

La description ci-dessous de cette méthode se fera à travers l'exemple suivi de la base de connaissances suivante :

$$B = \{\forall x\ p(x) \Rightarrow \exists y\ q(x, y), \qquad \neg\exists x\ q(a, x), \qquad p(a)\}$$

On rappelle que les autres problèmes de décision peuvent se ramener à un problème de satisfiabilité, ainsi $\{\varphi_1, \varphi_2, \varphi_3\} \models \psi$ ssi $\{\varphi_1, \varphi_2, \varphi_3, \neg\psi\}$ est insatisfiable.

Mise sous forme normale négative. La première étape consiste à remplacer dans B chaque formule φ par une formule φ' sous forme normale négative (le connecteur $\neg$ n'apparaît que devant les atomes) dans laquelle on ne considère que les connecteurs $\neg$, $\wedge$ et $\vee$. On obtient alors la base de connaissance B' qui est équivalente (et, donc, équisatisfiable) à B. Pour ce faire, on s'appuie sur les équivalences logiques permettant d'éliminer les autres connecteurs (p. ex., $\alpha \Rightarrow \beta \quad \equiv \quad \neg\alpha \vee \beta$) et celles qui permettent de faire « descendre » le conneteur $\neg$ plus profondément dans la formule (p. ex., $\neg\exists x\ \alpha \equiv \forall x\ \neg\alpha$).

Sur l'exemple, cela donne :

$$B' = \{\forall x\ \neg p(x) \vee \exists y\ q(x, y), \qquad \forall x\ \neg q(a, x), \qquad p(a)\}$$

Développement de l'arbre. On crée un arbre en mettant initialement toutes les formules de φ' l'une en-dessous de l'autre (racine, enfant de la racine, petit-enfant de la racine, etc.), dans un ordre quelconque. Puis, on utilise les règles de transformation suivantes :

— $\rightarrow_\wedge$ et $\rightarrow_\vee$ comme pour la méthode des tableaux en logique propositionnelle ;

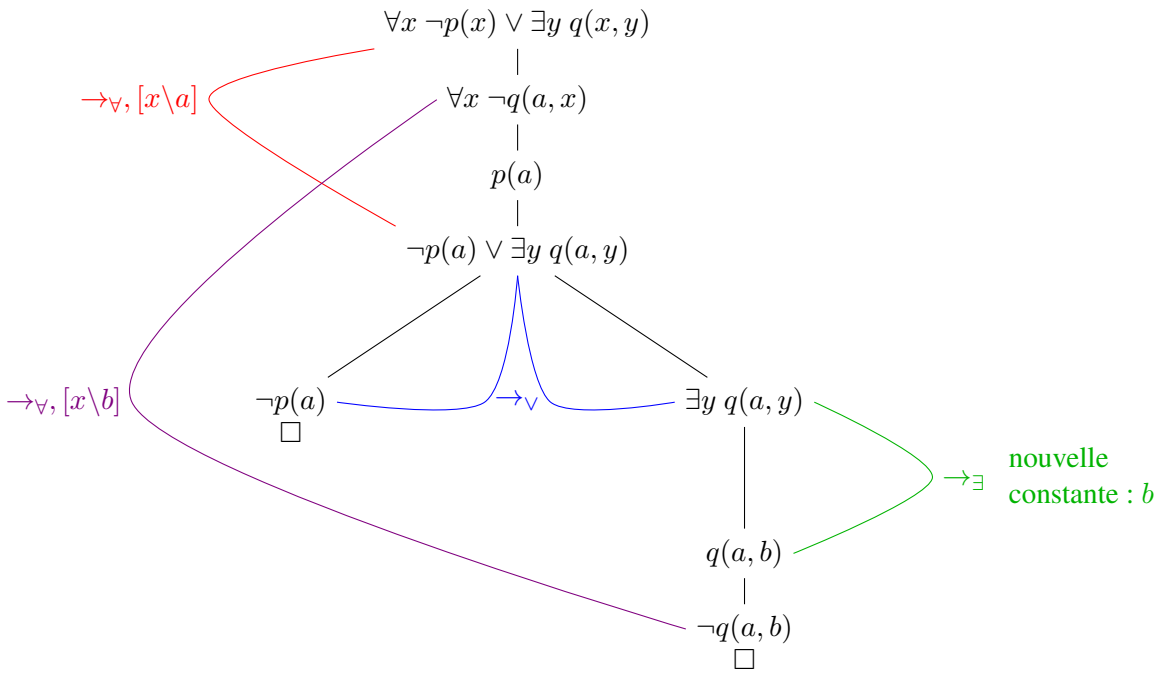


FIGURE 8 – Méthode des tableaux sémantiques en logique du premier ordre sur un exemple.

— $\rightarrow_\exists$ s'applique sur une formule de la forme $\exists x\ \varphi$. Elle produit, en prolongeant l'arbre, une formule $\varphi[x\backslash a]$ (i.e., la formule obtenue en substituant toutes les occurrences libres de x dans φ par a) où a est une *nouvelle* constante, c'est-à-dire une constante n'apparaissant pas dans les formules précédentes.
On notera que les formules $\exists x\ \varphi$ et $\varphi[x\backslash a]$ ne sont pas équivalentes, mais qu'elles sont équi-satisfiables : comme on veut faire un test de satisfiabilité, c'est suffisant. Cela peut également être considéré comme une étape de skolémisation de $\exists x\ \varphi$, où a est une fonction de Skolem d'arité nulle.

— $\rightarrow_\forall$ s'applique sur une formule de la forme $\forall x\ \varphi$. Elle produit, en prolongeant l'arbre, une formule $\varphi[x\backslash t]$ où t est un terme à *choisir* : t peut être construit à partir de n'importe quel symbole de fonction, variable ou constante, mais si le choix n'est pas adéquat, on peut ne pas terminer.
On peut noter que $\rightarrow_\forall$ peut s'appliquer plusieurs fois dans la même branche sur une même formule $\forall x\ \varphi$, avec des substitutions vers des termes différents.

Les conflits considérés sont de la forme $\{A, \neg A\}$ (sur une même branche) où A est un atome.
Sur l'exemple précédent, on obtient l'arbre de la figure 8.

Conclusion. Si on arrive à terminer un arbre par un conflit sur toutes les branches, alors cela entraîne que B' (et donc B) est insatisfiable.

On peut noter que si dans une branche il y a une formule de la forme $\forall x\ \varphi$, on peut appliquer indéfiniment la règle $\rightarrow_\forall$ à partir de cette formule (en utilisant autant de termes à subsituter à x qu'on le désire). Donc, si B' est satisfiable et qu'il y a un quantificateur $\forall$ dans une formule de B', il est très fréquent qu'on ne puisse pas terminer une branche.

Exercice 35 *Montrez le résultat suivant en utilisant la méthode des tableaux sémantiques :*

$$\{p(a),\quad \forall x\ p(x) \Rightarrow p(f(x))\} \qquad \models \qquad p(f(f(a)))$$

Exercice 36 *Traduire en logique du premier ordre les phrases suivantes en utilisant le prédicat binaire* `ami_de` *et les constantes* `jacques`, `léonne` *et* `claude` *:*

(P1) Les amis des amis de x sont des amis de x (quel que soit x).

(P2) Si x est ami de y alors y est ami de x (quels que soient x et y).

(P3) Jacques est ami de Léonne et Léonne est amie de Claude.

(P4) Claude n'est pas ami(e) de Jacques.

Montrez par la méthode des tableaux sémantiques que l'ensemble de ces quatre phrases est incohérent. Vous utilisez les abréviations $p =$ `ami_de`, $a =$ `jacques`, $b =$ `léonne` et $c =$ `claude`.

3.7.5 Les clauses de Horn du premier ordre : un fragment décidable de $(\mathcal{L}1\mathcal{O}, \models)$

⚠Cette section ne contient que des éléments basiques liés aux clauses de Horn et au langage Prolog.

Les clauses de $(\mathcal{L}1\mathcal{O}, \models)$. La définition des clauses en logique du premier ordre est la même qu'en logique propositionnelle, sauf que la notion de littéral est différente (variable propositionnelle ou négation de variable propositionnelle dans $(\mathcal{LP}, \models)$, atome ou négation d'atome dans $(\mathcal{L}1\mathcal{O}, \models)$).

Les clauses de la logique du premier ordre sont souvent considérées comme étant quantifiées universellement de façon implicite. Ainsi :

La clause	$p(x, a) \vee \neg q(y) \vee r(x, y, b)$
se lit	$\forall x \forall y\ p(x, a) \vee \neg q(y) \vee r(x, y, b)$
ou encore	$\forall y \forall x\ p(x, a) \vee \neg q(y) \vee r(x, y, b)$

(rappelons qu'on peut échanger l'ordre des quantificateurs universels). Plus généralement, φ se lit $\overrightarrow{\forall}\varphi$ (cf. notion de clôture universelle introduite en section 3.3.2).

Question 9 *Pour toute formule de $(\mathcal{L}1\mathcal{O}, \models)$, il existe une formule équi-satisfiable qui peut s'exprimer comme une conjonction de clauses implicitement quantifiées. Comment peut-on montrer cela en passant par la mise sous forme prénexe, la skolémisation, la mise sous FNC et l'équivalence (21) ?*

Les clauses de Horn. Comme dans le cadre propositionnel, une clause de Horn est une clause ne contenant qu'un seul littéral positif. Si elle contient au moins deux littéraux, elle s'écrit comme une implication et on parle de *règle*. Si elle contient un seul littéral, c'est un atome (on parlera de *fait* si cet atome ne contient pas de variable).

On notera $\mathcal{LH}1\mathcal{O}$ l'ensemble des clauses de Horn (sur un vocabulaire donné). À titre d'exemple, les formules suivantes forment une base de connaissances de $(\mathcal{LH}1\mathcal{O}, \models)$ (c'est un exemple simple dans lequel il n'y a pas de symbole de fonction) :

$$\texttt{père}(x, y) \Rightarrow \texttt{parent}(x, y) \quad (r_1)$$
$$\texttt{mère}(x, y) \Rightarrow \texttt{parent}(x, y) \quad (r_2)$$
$$\texttt{père}(x, y) \wedge \texttt{parent}(y, z) \Rightarrow \texttt{grand-père}(x, z) \quad (r_3)$$
$$\texttt{mère}(x, y) \wedge \texttt{parent}(y, z) \Rightarrow \texttt{grand-mère}(x, z) \quad (r_4)$$
$$\texttt{parent}(x, y) \Rightarrow \texttt{ascendant}(x, y) \quad (r_5)$$
$$\texttt{parent}(x, y) \wedge \texttt{ascendant}(y, z) \Rightarrow \texttt{ascendant}(x, z) \quad (r_6)$$

$$\texttt{père}(\texttt{Marcel}, \texttt{Maurice}) \quad (f_1)$$
$$\texttt{mère}(\texttt{Henriette}, \texttt{Maurice}) \quad (f_2)$$
$$\texttt{père}(\texttt{Maurice}, \texttt{Juliette}) \quad (f_3)$$
$$\texttt{père}(\texttt{Maurice}, \texttt{Léon}) \quad (f_4)$$
$$\texttt{mère}(\texttt{Cunégonde}, \texttt{Léon}) \quad (f_5)$$

La déduction dans $(\mathcal{LH}1\mathcal{O}, \models)$ est décidable. Un algorithme pour y parvenir s'appuie sur le chaînage arrière, comme pour les clauses de Horn propositionnelles. La différence est qu'alors que pour que deux littéraux de clauses de Horn puissent être comparées, il faut qu'ils soient égaux, alors que l'utilisation de variables en logique du premier ordre fait que les littéraux doivent être *unifiés*.

L'***unification*** consiste, étant donné deux termes avec des variables à trouver des substitutions de chaque variable pour que ces termes coïncident. Par exemple, une unification des termes $f(x, y)$ et $f(a, g(x))$ est composée des substitutions $x\backslash a$ et $y\backslash g(a)$. En effet, $f(x, y)[x\backslash a][y\backslash g(a)] = f(a, g(x))[x\backslash a][y\backslash g(a)] = f(a, g(a))$.

L'algorithmique de la déduction dans cette logique ne fait pas l'objet de ce cours.

Prolog. Prolog est un langage de programmation s'appuyant sur les clauses de Horn. Il est équivalent, en terme de puissance de calcul, à n'importe quel langage de programmation classique, mais nous nous contenterons ici de la partie purement logique de Prolog (et pas dans sa totalité).

Il y a des variantes syntaxiques de Prolog mais, classiquement, on distingue les variables et les constantes par le fait que les premières commencent par une lettre majuscule. De plus, toute formule est finie par un point. Le $\wedge$ est remplacé par une virgule. L'implication est notée « à l'envers » (la conclusion à gauche et les prémisses à droite) et par le symbole `:-`.

Ainsi, la base de connaissances ci-dessus peut s'écrire ainsi en Prolog :

```
% Base de regles
parent(X, Y) :- pere(X, Y).
parent(X, Y) :- mere(X, Y).
grand_pere(X, Z) :-
  pere(X, Y),
  parent(Y, Z).
grand_mere(X, Z) :-
  mere(X, Y),
  parent(Y, Z).
ascendant(X, Y) :-
  parent(X, Y).
ascendant(X, Z) :-
  parent(X, Y),
  ascendant(Y, Z).

% Base de faits
pere(marcel, maurice).
mere(henriette, maurice).
pere(maurice, juliette).
pere(maurice, leon).
mere(cunegonde, leon).
```

Une requête en Prolog est un terme (avec ou sans variable). Par exemple, la requête

```
?- grand_pere (marcel, leon).
```

donnera la réponse `Yes`. En revanche, la requête

```
?- grand_pere (edouard, henriette).
```

donnera la réponse `No`, ce qui signifie que la base de connaissances ne permet pas d'impliquer qu'`edouard` est grand-père d'`henriette`. C'est peut-être le cas, mais ce n'est pas déductible de la base de connaissances (on parle parfois d'hypothèse du monde clos à ce sujet). La requête suivante :

```
?- ascendant (henriette, X).
```

donnera les `X` tels qu'on peut déduire qu'`henriette` est ascendant(e) de `X` (les descendants connus d'Henriette) :

```
  X = maurice ;
  X = juliette ;
  X = leon ;
  No
```

Le point-virgule peut se lire « donne-moi une autre solution » et le `No` peut se lire « pas d'autre solution ».

Exercice 37 *En effectuant un chaînage arrière et des unifications, décrivez le processus qui a permis de répondre à la requête* `ascendant (henriette, X).`*, jusquà l'obtention d'un résultat* `X = ...`.

Exercice 38 *Reprenez l'exercice 35 en utilisant le chaînage arrière. Il vous faudra dans un premier temps traduire la base de connaissances en Prolog.*

Exercice 39 *On considère la base de connaissances en Prolog suivante :*

```
% Base de faits                          % Base de règles
p(a, b).   % (f1)                       p(Y, X) :- p(X, Y).          % (r1)
p(b, c).   % (f2)                       p(X, Z) :- p(X, Y), p(Y, Z)  % (r2)
```

On veut montrer que cette base entraîne le fait `p(c, a)`.

Expliquez pourquoi cela permet de répondre à la question de cohérence de l'exercice 36.

Répondez-y en utilisant un chaînage arrière avec la précaution supplémentaire suivante : quand un atome est généré dans une branche de l'arbre dans laquelle il a déjà été généré, on arrête le développement de cette branche par un échec. Cette précaution permet à l'exécution de ce chaînage arrière de terminer.

On notera que l'ordre des lignes est important et a été choisi pour rendre le chaînage arrière plus rapide.

3.8 Logique du premier ordre avec prédicat d'égalité

La logique du premier ordre telle qu'on l'a définie ne contient pas de prédicat d'égalité. Ce prédicat binaire est noté $=$ et utilisé de façon infixée : les atomes correspondant sont de la forme $t_1 = t_2$ où t_1 et t_2 sont deux termes. Soit n le nombre de variables considérées. Si $\mathcal{I} = (\Delta_\mathcal{I}, \cdot^\mathcal{I})$ est une interprétation, alors

$$(t_1 = t_2)^\mathcal{I}(u_1, u_2, \ldots, u_n) = \begin{cases} \mathtt{V} & \text{si } t_1^\mathcal{I}(u_1, u_2, \ldots, u_n) = t_2^\mathcal{I}(u_1, u_2, \ldots, u_n) \\ \mathtt{F} & \text{sinon} \end{cases}$$

L'égalité sur un ensemble est une relation d'équivalence. Elle est donc réflexive, symétrique et transitive, d'où :

$$\begin{aligned} &\models \forall x\ x = x \\ &\models \forall x \forall y\ x = y \Rightarrow y = x \\ &\models \forall x \forall y \forall z\ x = y \wedge y = z \Rightarrow x = z \end{aligned}$$

Par ailleurs, si on remplace un terme t_1 par un terme t_2 dans une formule dans un contexte dans lequel $t_1 = t_2$ s'interprète comme vrai, alors la vérité de la première formule coïncide avec la vérité de la seconde :

$$\models \forall x \forall y\ x = y \Rightarrow (\varphi \Leftrightarrow \varphi[x \backslash y])$$

Question 10 *Dans la formule précédente, on peut remplacer le $\Leftrightarrow$ par un $\Rightarrow$. Pourquoi ?*

Ce prédicat d'égalité permet d'introduire la notation $\exists!$ (« il existe un unique »). Par exemple

$$\forall x\ \mathtt{humain}(x) \Rightarrow \exists! y\ \mathtt{nez}(y, x)$$

exprime que toute personne a exactement un nez. Cette notation peut être considérée comme une abréviation :

$$\exists! x\ \varphi \qquad \exists x\ (\varphi \wedge \forall y\ \varphi[x \backslash y] \Rightarrow y = x)$$

Par exemple, la formule sur le nez unique s'écrira :

$$\forall x\ \mathtt{humain}(x) \Rightarrow \exists y\ (\mathtt{nez}(y, x) \wedge \forall z\ (\mathtt{nez}(z, x) \Rightarrow z = y))$$

Une fois qu'on est capable d'exprimer que deux termes t_1 et t_2 sont égaux ($t_1 = t_2$) ou différents ($\neg(t_1 = t_2)$), on est capable de compter. Par exemple, pour exprimer que Marc a au moins deux enfants, on peut écrire :

$$\exists x_1 \exists x_2\ \mathtt{parent}(\mathtt{marc}, x_1) \wedge \mathtt{parent}(\mathtt{marc}, x_2) \wedge \neg(x_1 = x_2)$$

3.9 Représenter des connaissances en logique du premier ordre

Pour représenter des connaissances dans une logique, en partant de phrases écrites dans un langage naturel, il faut d'abord définir le vocabulaire. En logique du premier ordre, ce vocabulaire sera constitué par les constantes, les symboles de fonction et les prédicats.

Exercice 40 *« Traduisez » en logique du premier ordre les phrases suivantes :*

- *— À père avare, fils prodigue.*
- *— Aucun avare n'est altruiste.*
- *— Personne, excepté les avares, ne conserve les coquilles d'œufs.*
- *— Aucune personne altruiste ne conserve les coquilles d'œufs.*

Exercice 41 *Les trois dernières phrases de l'exercice précédant sont dues à Lewis Carroll. On se pose la question « La troisième phrase peut elle se déduire des deux premières ? » Formalisez cette question.*

Exercice 42 *Soit* `parent`, `féminin` *et* `masculin` *les prédicats tels que* $\texttt{parent}(x, y)$ *se lit « x est un parent (au sens père ou mère) de y »,* $\texttt{féminin}(x)$ *(resp.,* $\texttt{masculin}(x)$*) se lit « x est de sexe féminin (resp., masculin) ». On peut, en s'appuyant sur ces prédicats, « définir » d'autres prédicats par des axiomes. Par exemple, on peut définir le prédicat* `mère` *par :*

$$\forall x \forall y \; \texttt{mère}(x, y) \Leftrightarrow (\texttt{parent}(x, y) \wedge \texttt{féminin}(x))$$

Définissez de façon similaire les prédicats binaires `père`, `grand-parent`, `grand-mère`, `grand-père`, `grand-mère-paternelle`, `sœur`, `frère`, `demi-sœur`, `demi-frère`, `tante`, `oncle`, `nièce`, `neveu`, `cousine`, `cousin` *(en se limitant pour les cousines et les cousins aux enfants des oncles et tantes),* `ascendant` *et* `descendant`.

Vous pourrez, lors de la définition du prédicat p, vous appuyer sur un ou des prédicats déjà définis précédemment.

Vous pourrez aussi, quand ce sera nécessaire, utiliser le prédicat d'égalité.

Exercice 43 *En fin de section 3.8, la représentation en logique du premier ordre avec égalité du fait que Marc a au moins deux enfants est donnée.*

Dans ce même formalisme, exprimez le fait que Marc a en fait exactement deux enfants.

Par ailleurs, Marc est enfant unique (il n'a ni frère, ni sœur, ni demi-frère, ni demi-sœur). Représentez cela dans ce même formalisme.

Exercice 44 *La logique du premier ordre est plus expressive que la logique propositionnelle au sens suivant :*

(a) Tout ce qui peut s'exprimer en logique propositionnelle peut s'exprimer en logique du premier ordre.

(b) Il existe des phrases qui se traduisent « naturellement » en logique du premier ordre mais pas en logique propositionnelle.

Pour illustrer le point (a), traduisez la phrase « Le chat est un félin. » dans les deux formalismes. De façon générale, comment traduire un énoncé en logique propositionnelle vers la logique du premier ordre ?

Trouvez un exemple qui illustre le point (b).

*****

4 Conclusion

Ce cours a pour ambition de donner un premier aperçu du vaste domaine de la logique, à travers deux formalismes très importants : la logique propositionnelle (qui est décidable) et la logique du premier ordre (qui est semi-décidable).

Il y aurait évidemment énormément à dire en plus sur ces deux formalismes, en particulier sur les preuves des différents résultats, sur d'autres approches algorithmiques et sur d'autres inférences dans ces formalismes.

Par ailleurs, il existe bien d'autres logiques que ces deux-là. La section 4.1 en évoque quelques unes. Enfin, d'autres inférences que les inférences déductives sont étudiées. La section 4.2 en présente rapidement.

4.1 D'autres logiques

La logique du deuxième ordre. De façon générale, ce cours présuppose qu'il existe une théorie des ensembles sur laquelle on s'appuie. En logique du premier ordre, on considère deux « niveaux » : les éléments des ensembles ($u \in \Delta_{\mathcal{I}}$, interprétant les termes) et les relations entre eux ($R \subseteq \Delta_{\mathcal{I}}^n$, interprétant les prédicats). On ne peut pas représenter de façon adéquate une notion d'« ensemble d'ensembles ». Par exemple, si on veut représenter l'ensemble des espèces, dont un élément est une espèce, laquelle peut être modélisée par l'ensemble des individus qui la contient ($a \in E$ et $E \in M$ où a est l'individu Toto, E est l'espèce des ornithorynques et M est l'ensemble des espèces), on a besoin d'une logique plus expressive. La logique du deuxième ordre permet de faire cela.

Cette logique permet d'exprimer, par exemple, le principe de la preuve par récurrence sous la forme suivante :

$$\forall p\ p(\mathtt{0}) \wedge \forall x\ p(x) \Rightarrow p(\mathtt{S}(x)) \Rightarrow \forall x\ p(x)$$

où `0` est la constante s'interprétant comme l'entier 0 et `S` est le symbole de fonction s'interprétant comme la fonction $\mathtt{S}^{\mathcal{I}} : n \in \mathbb{N} \mapsto n+1$. On notera que la variable p varie dans l'ensemble des prédicats (plus précisément, dans l'ensemble des prédicats unaires) : c'est une variable du deuxième ordre.

Pour l'étude d'une telle logique, il paraît naturel de chercher à définir une sémantique ($\models$) et un système formel S qui soit correct et complet. Kurt Gödel avait prouvé en 1928 qu'un tel système formel existait pour la logique du premier ordre (on parle du théorème de complétude de Gödel). En 1931, il a montré que ce n'était pas possible en logique du deuxième ordre [14] : il n'existe pas de système formel correct et complet pour cette logique (on parle du théorème d'incomplétude de Gödel). Sa preuve s'appuie sur la puissance même du formalisme dans lequel on peut « encoder » la notion de prouvabilité. Elle utilise aussi le principe de la diagonale de Cantor.

On peut, au-delà de la logique du deuxième ordre, définir la logique du troisième ordre (dans laquelle on peut « manipuler » des ensembles d'ensembles d'ensembles), du quatrième ordre, etc.

Les logiques non monotones. Les logique $(\mathcal{L}, \models)$ vues dans ce cours sont monotones, ce qui signifie ceci :

$$\text{si } B_1 \subseteq B_2 \text{ et } B_1 \models \varphi \text{ alors } B_2 \models \varphi$$

Cela peut s'exprimer par les fonctions `Cn` et $\mathcal{M}$ et l'ensemble $\mathcal{B}$ des bases de connaissances :

— `Cn` est une fonction croissante de $(\mathcal{B}, \subseteq)$ dans $(\mathcal{L}, \subseteq)$;

— $\mathcal{M}$ est une fonction décroissante de $(\mathcal{B}, \subseteq)$ dans $(\Omega, \subseteq)$ ([15]).

Une logique non monotone $(\mathcal{L}, \mid\!\sim)$ est donc une logique dans laquelle il existe des conséquences « défaisables » : partant d'une base de connaissances B_1, on peut conclure φ, mais si on enrichit B_1 en B_2, il est possible que B_2 n'entraîne pas φ.

Une telle logique est utile en particulier pour représenter des connaissances par défaut, telles que « Les oiseaux volent, sauf exception. » Ainsi, si on a un individu dont on sait qu'il est un oiseau, mais pas plus, on inférera qu'il vole. Si on apprend ensuite que c'est une autruche et qu'on sait que les autruches ne volent pas, alors on retirera la conclusion disant que cet oiseau vole.

14. Plus précisément, il a montré que ce n'était pas possible pour une théorie couvrant les théorèmes de base de l'arithmétique : une telle théorie peut s'exprimer en logique du deuxième ordre mais pas en logique du premier ordre.

15. Où Ω est l'ensemble des interprétations, ce qui a un sens précis en logique propositionnelle mais peut être sujet à caution en logique du premier ordre...

Un tel mécanisme existe dans les formalismes de représentation des connaissances par objets (formalismes proches de la programmation par objets), grâce au mécanisme d'exception à l'héritage. Mais il existe bien d'autres formalismes, notamment, la logique des défauts de Reiter, dans laquelle on peut exprimer des règles par défaut telles que :

$$\frac{\texttt{oiseau} \ : \ \texttt{vole}}{\texttt{vole}}$$

qui se comprend comme : si on a un oiseau et qu'il n'est pas contradictoire avec ce qu'on sait déjà qu'il vole, alors on en conclura qu'il vole.

Les logiques intuitionnistes. Une logique intuitionniste est une logique dans laquelle une formule φ se lit « φ est prouvable », plutôt que « φ est vrai » en logique classique. Une conséquence de cela est que la formule $a \vee \neg a$ qui est une tautologie dans le cadre classique [16], ne l'est pas dans un cadre intuitionniste : on peut avoir a et $\neg a$ tous les deux non prouvables.

Dans une telle logique, une preuve est *constructive* : si on veut prouver $a \vee \neg a$, il faut prouver a ou prouver $\neg a$. De même, si on veut prouver (dans une logique intuitionniste du premier ordre) $\exists x\, p(x)$, une preuve dans cette logique doit consister à « construire » un x tel que $p(x)$.

Une façon d'illustrer cette idée de façon intuitive est liée à la notion de plan : si on prouve de façon intuitionniste qu'il existe un plan permettant de réaliser un but (par exemple, trouver un chemin entre deux points), ce plan doit pouvoir se trouver dans la preuve.

Les logiques modales. Seules les logiques modales s'appuyant sur la logique propositionnelle sont évoquées dans cette section.

Une logique modale est une logique dans laquelle on a des opérateurs modaux qui spécifient dans quels cas une proposition est vraie. On peut étudier les logiques modales en tant que telles ou leurs applications à différents domaines. Voici deux applications :

— En logique modale temporelle, φ se comprend comme la vérité de φ à l'instant présent et $\Box\varphi$ se comprend comme la vérité de φ à tout instant présent ou futur. Par conséquent, $\neg\Box\neg\varphi$ — qu'on abrège en $\Diamond\varphi$ — se comprend comme φ est vrai au moins une fois à partir de l'instant présent. Par exemple, la formule $\Box\Diamond$`il_pleut` signifie qu'à tout instant t, il existe un instant coïncidant avec t ou ultérieur à t durant lequel il pleuvra.

— En logique épistémique, on a une modalité K_a, paramétrée par un agent a : $K_a\varphi$ signifie que l'agent a sait que φ est vrai. Le principe d'introspection positive s'écrit : $K_a\varphi \Rightarrow K_aK_a\varphi$ (si a sait que φ est vraie, alors a sait que a sait que φ est vraie).

La sémantique usuelle dans les logiques modales est la sémantique de Kripke qui est une sémantique en théorie des modèles des logiques modales.

Les logiques multi-valuées. La sémantique des logiques classiques s'appuie sur deux valeurs de vérité : `V` et `F`. Dans les logiques multi-valuées (ou floues), on considère des valeurs intermédiaires. On utilise souvent comme ensemble de valeurs l'intervalle $[0;1]$ où 0 est assimilé à `F`, 1, à `V` et pour $0 < a < b < 1$, les valeur de vérité a et b sont « ni `V` ni `F` » mais entre les deux, quoique a soit « moins vrai » que b.

Par exemple, des termes linguistiques tels que « petit », « jeune », etc., ne s'accomodent pas toujours de frontières nettes : si on a une pomme de 4 (resp., 8) centimètres de diamètre, on pourra la considérer comme une petite pomme (resp., une pomme non petite). Mais pour une pomme de 6 centimètres de diamètre, cela se discute : l'ensemble des petites pommes peut être modélisé par un *sous-ensemble flou* PP de l'univers U des pommes, caractérisé par un degré d'appartenance : $\mu_{PP} : x \in U \mapsto \mu_{PP}(x) \in [0;1]$ avec $\mu_{PP}x = \begin{cases} 1 & \text{si } D(x) \leqslant 5 \\ \frac{7-D(x)}{7-5} & \text{si } 5 \leqslant D(x) \leqslant 7, \\ 0 & \text{si } D(x) \geqslant 7 \end{cases}$

où $D(x)$ est le diamètre de la pomme x en centimètres.

16. Dans le cas classique, $a \vee \neg a$ est appelé « le tiers exclus » : a est vrai ou a est non vrai, il n'y a pas de troisième possibilité.

Ce genre de formalisme est utilisé pour représenter des *connaissances imprécises* ou vagues (« environ 2 heures », etc.).

Les logiques possibilistes. Les logiques possibilistes sont utiles pour la représentation de *connaissances incertaines* (à ne pas confondre avec les connaissances imprécises) : « Il y aura de la neige demain sur le département de la Meurthe-et-Moselle » est une affirmation soit vraie (si au moins un flocon tombe sur le sol du département) soit fausse. En revanche, aucune des deux possibilités n'est certaine.

Pour modéliser l'incertain, plusieurs théories existent, la plus connue d'entre elles est la théorie des probabilités. La théorie des possibilités en est une autre, qui a l'avantage de se « marier » plus facilement avec la logique. Les notions d'événements sont communs aux deux théories. En théorie des possibilités, on associe le *degré de possibilité* $\Pi(E)$ et le *degré de nécessité* $N(E)$ à un événement $E \subseteq U$, avec $N(E) = 1 - \Pi(\overline{E})$.

Sur un lancer de pièce, s'il est parfaitement possible que la pièce tombe sur pile ou sur face ($\Pi(p) = \Pi(f) = 1$). La possibilité qu'elle tombe sur la tranche est non nulle mais faible, mettons $\Pi(t) = 0{,}1$. La possibilité d'une union étant le maximum des possibilités, si l'univers est $U = \{p, f, t\}$, la nécessité qu'une pièce tombe sur pile ou sur face est de $N(\{p, f\}) = 1 - \Pi(\overline{\{p, f\}}) = 1 - \Pi(\{t\}) = 0, 9$.

Dans le cadre de la logique propositionnelle possibiliste, les formules sont $(\varphi, \Pi\alpha)$ et $(\varphi, N\alpha)$ où $\varphi \in \mathcal{LP}$. $(\varphi, \Pi\alpha)$ (resp., $(\varphi, N\alpha)$) représente le fait que la possibilité (resp., la nécessité) que φ se réalise est supérieure ou égale à α.

Autres logiques. Il existe bien d'autres logiques. Certaines sont des combinaisons des logiques précédentes. Le choix d'une logique dépend de son usage. Par exemple, si on a des connaissances à la fois imprécises et incertaines, on peut envisager d'utiliser une logique possibiliste multi-valuée.

4.2 D'autres inférences

De façon générale, une inférence (ou un raisonnement) consiste à produire de nouvelles connaissances (les *conclusions*) à partir de connaissances disponibles (les *prémisses*). La relation $\models$ pour la logique propositionnelle et la logique du premier ordre représente une inférence ***déductive*** au sens suivant : si les prémisses sont sûres, les conclusions seront également sûres. Par la contraposée, si on remet en cause les conclusions d'un raisonnement déductif, on doit aussi remettre en cause ses prémisses (au moins une de ses prémisses). Il existe des raisonnements non déductifs, aussi appelés raisonnements ***hypothétiques*** : pour un tel raisonnement, même si on est sûr des prémisses, les conclusions ne sont que des hypothèses.

La généralisation inductive [17] est un raisonnement hypothétique dont les prémisses sont des formules φ_1, φ_2, …, φ_n et la conclusion est une formule Φ qui *généralise* les φ_i au sens où $\varphi_i \models \Phi$.

C'est un raisonnement couramment utilisé en apprentissage artificiel : partant de faits particuliers (les φ_i) il va au général (Φ). Par exemple, si on observe qu'il ne pleut pas à Nancy du premier au 5 juillet ($\neg$`il_pleut_le(1_juillet)`, $\neg$`il_pleut_le(2_juillet)`, …, $\neg$`il_pleut_le(5_juillet)`) on peut inférer qu'il ne pleut jamais à Nancy ($\forall x$ `est_un_jour`$(x) \Rightarrow \neg$`il_pleut_le`(x)).

Plus on a de φ_i, plus la conclusion est renforcée, mais elle ne sera jamais certaine : tant qu'on n'avait pas rencontré d'ornithorynque ni d'échidné, on pouvait penser qu'aucun mammifère n'était ovipare !

L'abduction est un raisonnement souvent utilisé dans des diagnostics, que ce soit en médecine, en mécanique, en informatique (p. ex., quand on débogue un programme), etc. C'est un raisonnement dont les prémisses sont des conséquences et les conclusions des (hypothèses de) causes.

En médecine, l'observation de symptômes conduit à des diagnostics expliquant ces symptômes : si on observe sur le patient une fièvre et des frissons, on peut inférer par abduction que c'est une grippe, mais ce raisonnement n'est pas certain. En effet, d'autres maladies ont ces symptômes et même si on a des symptômes caractéristiques

17. Parfois appelée « induction », mais ce terme est problématique parce qu'il est aussi utilisé comme un type particulier de raisonnement déductif !

d'une seule maladie connue, cela n'entraîne pas nécessairement que le patient a cette maladie : il peut s'agir d'une nouvelle maladie.

Le raisonnement par analogie s'appuie sur des relations entre quatre termes : $A : B :: C : D$, qui se lit « A est à B ce que C est à D. » En général, les prémisses d'un raisonnement par analogie sont trois des quatre termes et la conclusion, le quatrième terme. À l'origine, il était utilisé dans le cadre de problèmes de proportions : si $\frac{A}{B} = \frac{C}{D}$ avec $A = 3$, $B = 6$, $C = 5$, que vaut D ? Il s'est étendu par la suite à d'autres domaines. Par exemple, Georges Clémenceau a écrit « La justice militaire est à la justice ce que la musique militaire est à la musique. » (que le lecteur interprétera comme il l'entend).

Une application du raisonnement par analogie est le raisonnement à partir de cas, qui permet de résoudre des problèmes sur la base de problèmes déjà résolus. Par exemple, si mon problème est la confection d'une tarte aux poires et que je sais comment préparer une tarte aux pommes, il me suffit d'adapter la recette de la tarte au pommes pour tenir compte du changement des pommes en poires (remplacement des ingrédients, ajustement de la quantité de sucre, etc.).

La révision des croyances a pour objectif de faire évoluer une base de croyances [18] lors de l'arrivée de nouvelles croyances, supposées prioritaires. Par exemple, initialement, je pense que tous les oiseaux volent et que les autruches sont des oiseaux. Puis, on me présente une autruche qui est incapable de voler. Il y a donc incohérence de l'ensemble de mes croyances. Une façon de faire pour restaurer la croyance en conservant les nouvelles croyances consiste à supprimer la croyance « Tous les oiseaux volent. » Une autre consiste à supprimer la croyance « Les autruches sont des oiseaux. »

Il existe des liens entre révision des croyances et logiques non monotones. Il y a d'autres opérations de changements de croyances et c'est un domaine de recherches très actif.

18. Le terme « croyance » est une traduction du terme anglais *belief* qui n'a pas de connotation religieuse. L'ensemble des croyances d'un agent est ce qu'il pense être vrai. On les distingue parfois de ses connaissances de la façon suivante : les croyances sont susceptibles d'évoluer, les connaissances non.