Logiques pour l'intelligence artificielle

P. Gribomont

2006-2007

Table des matières

Avant-propos

Du point de vue de son enseignement, la logique formelle élémentaire se trouve dans une situation paradoxale. D'une part, cet enseignement est favorisé par plusieurs facteurs objectifs mais, d'autre part, les résultats obtenus sont souvent décevants. Nous développons ici brièvement ces deux points, et proposons quelques pistes pour améliorer la situation.

Quelques atouts. Citons d'abord trois raisons pour lesquelles un cours d'introduction à la logique formelle devrait être un cours facile à donner, et facile à assimiler.

- *La matière proprement dite est objectivement facile.*
  Analyser une formule propositionnelle, telle $((p \wedge q) \Rightarrow r) \equiv (p \Rightarrow (q \Rightarrow r))$, est plus simple qu'analyser une formule arithmétique ou algébrique telle que $\sqrt{ab} \leq (a+b)/2$. En effet, les propositions ne peuvent être que vraies ou fausses, tandis que les nombres forment un ensemble infini. Cela a pour conséquence que les opérations, les règles et les méthodes de la logique propositionnelle sont moins nombreuses et plus simples que celles de l'algèbre élémentaire. D'une manière analogue, il est plus facile d'analyser une formule du calcul des prédicats, telle que (exemple classique) $\forall x\,(P(x) \Rightarrow Q(x)) \Rightarrow (\forall x\,P(x) \Rightarrow \forall x\,Q(x))$, que de résoudre une équation intégrale, telle que $y(x) = 1 + \int_0^x y(t)\,\mathrm{d}t$. Enfin, comme nous le verrons, presque tous les théorèmes de la logique élémentaire se démontrent de manière quasi systématique, alors que chaque théorème de mathématique élémentaire, fût-il aussi vieux que celui de Pythagore, ressemble à un défi.
- *Les références de bonne qualité, accessibles à l'autodidacte, ne manquent pas.*
  Même si, à l'échelle des mathématiques, la logique formelle est un domaine plutôt jeune, il a quand même plus d'un siècle ; le plus récent résultat que nous verrons, le principe de résolution, date de 1965 (il est même nettement antérieur en ce qui concerne la logique propositionnelle). En mathématique, tous les domaines de base ont fait l'objet de présentations didactiques nombreuses et soignées ; la logique n'échappe pas à la règle. La situation est moins favorable pour des domaines plus récents et moins fondés théoriquement, comme les systèmes experts ou les réseaux neuronaux artificiels.
- *Les mathématiques préparent à la logique.*
  La logique est la science du raisonnement et de l'expression formelle du raisonnement. Tout étudiant est amené à raisonner et à exprimer le fruit de ses cogitations oralement ou par écrit … Bien plus, les écueils traditionnels de la logique élémentaire (implication, démonstration, variables libres et liées, etc.) ont déjà été rencontrés ailleurs, dans des contextes mathématiques souvent plus difficiles. La notion d'implication formalise le lien existant entre l'hypothèse d'un théorème et sa thèse, notion familière aux étudiants qui distinguent condition nécessaire et condition suffisante et qui, plus généralement, ont une certaine expérience des démonstrations. Distinguer les rôles des variables x et y dans la formule $\forall x\,P(x,y)$ n'est pas plus difficile que distinguer les rôles de t et x dans l'intégrale $\int_0^x f(t)\,\mathrm{d}t$, ou ceux de i et j dans $\sum_i A_{ji}x_i$.

Quelques problèmes … Pourquoi alors l'étudiant, reconnaissant rapidement et sans hésitation la validité des formules $((p \wedge q) \Rightarrow r) \equiv (p \Rightarrow (q \Rightarrow r))$, et aussi celle de $\sqrt{ab} \leq (a+b)/2$, hésitera-t-il devant des questions innocentes, telles que

Soient A, B, X et Y des formules quelconques.
On pose $A' =_{def} (X \Rightarrow (A \Rightarrow Y))$ et $B' =_{def} (X \Rightarrow (B \Rightarrow Y))$.
Si $A \Rightarrow B$ est vrai, que peut-on dire de $B \Rightarrow A$, de $A' \Rightarrow B'$ et $B' \Rightarrow A'$?

et

Quel lien logique y a-t-il entre les formules
$\forall x \, \forall y \, (P(x) \Rightarrow Q(y))$ et $\exists x \, P(x) \Rightarrow \forall x \, Q(x)$?

Nous n'avons naturellement pas d'explication définitive à ce problème, et encore moins de remède infaillible, mais on peut néanmoins explorer quelques pistes. Tout d'abord, les trois points cités plus haut, bien qu'objectivement favorables, ne sont pas dépourvus d'effets pervers.

- La facilité de la matière peut susciter trois types de réactions négatives. Tout d'abord, "si c'est trop simple, ce n'est pas utile". Les applications non triviales de la logique sont pourtant nombreuses, mais le temps manque parfois pour les aborder. Ensuite, et cela surtout à propos de la logique propositionnelle, "pourquoi vouloir formaliser et théoriser à propos d'une arithmétique simpliste, limitée à 0 (faux) et 1 (vrai) ?". Enfin, la facilité conduit à l'imprudence, qui elle-même mène à l'erreur ! La logique renferme quand même quelques pièges ...
- Les bons livres existent, sans aucun doute, mais ne correspondent pas toujours aux attentes et besoins du lecteur. Un simple exposé du type "hypothético-déductif" habituellement utilisé en mathématique ne convient pas, même si paradoxalement la logique élémentaire s'y prête très bien. Ce genre d'exposé se lit avec peu d'effort mais conduit seulement à une compréhension passive des concepts, et non à une maîtrise active de l'outil qu'est la logique pour un informaticien. En outre, un tel exposé ne donne pas de justification à l'existence même de la logique mathématique et ne fera qu'amplifier les réactions négatives évoquées plus haut.
- Notons enfin que la maturité mathématique de l'apprenant ne s'accompagne pas toujours d'une motivation pour aborder une nouvelle branche des mathématiques ...

Quelques solutions. Les remèdes existent. Une approche historique et philosophique des concepts [L1] est un excellent moyen de contrer les réactions négatives, en montrant que beaucoup d'efforts ont été nécessaires pour aboutir aux concepts simples et épurés sur lesquels se base la logique moderne. Elle montre aussi que les progrès réalisés au cours des siècles l'ont souvent été à l'occasion de problèmes concrets ; on voit enfin que la formalisation de l'expression des raisonnements a été la voie royale conduisant à une meilleure compréhension de ceux-ci. L'inconvénient de cette approche est qu'elle allonge grandement la taille de l'exposé, surtout si on le complète d'une introduction à des problèmes de nature informatique [L2] auxquels la logique apporte une solution partielle ou complète. Tout en restant persuadé de l'intérêt pédagogique d'une approche historique et philosophique de cette matière, nous devons admettre qu'elle est peu compatible avec la durée maximale de 30 heures prévue au programme (travaux pratiques non compris), surtout pour des auditeurs fortement sollicités par ailleurs.

Un moyen radical de balayer les objections de simplicité et d'inutilité est de dépasser quelque peu la matière reconnue comme indispensable à l'informaticien, et d'aborder quelques grands problèmes tels l'incomplétude et l'indécidabilité de l'arithmétique, ou l'indécidabilité de la logique prédicative, conduisant à de sévères limitations dans les domaines

de l'algorithmique, de la démonstration automatique et des systèmes experts, notamment. On obtient alors un cours de mathématique plutôt volumineux et difficile, dont l'introduction dans un curriculum de sciences appliquées serait malaisée à justifier. (Recommandons néanmoins [CL] et [BM] à l'amateur.) L'effort à fournir par l'étudiant peu habitué à l'algèbre et aux mathématiques abstraites devient alors lourd, même quand l'auteur s'ingénie avec succès à motiver tout résultat et à en donner une bonne intuition avant d'en exposer une démonstration rigoureuse [S].

Il semble donc que les problèmes liés à l'enseignement de la logique se résolvent surtout par des développements supplémentaires, dont le simple volume peut rebuter l'étudiant. Signalons quand même que peu de domaines progressent aussi rapidement que la logique pour l'informatique ; le "Handbook of Logic in Computer Science" comporte six volumes, dont le premier contient plus de 800 pages [AGM].

En dépit de cette inquiétante inflation, on constate que la partie de la logique mathématique *réellement* nécessaire à l'informaticien est plutôt réduite, et peut aisément s'exposer en 30 heures et s'assimiler concrètement et pratiquement en 30 heures supplémentaires ... à condition de respecter une stricte discipline. Une double comparaison va nous permettre de préciser ce point. L'étudiant en sciences appliquées apprend, assimile et utilise une grande quantité de théorèmes d'analyse mathématique, à propos de fonctions réelles et complexes, d'intégrales et d'équations différentielles, de transformées de Laplace et de Fourier, etc. En contrepartie, si l'on peut dire, l'assimilation n'est pas toujours très profonde. On mémorise, ou on sait où retrouver, les formules d'intégration et de transformation de fonctions, mais on s'interroge moins, ou avec moins de succès, sur les conditions de validité de ces formules. Le plus souvent, il n'y a pas de conséquences fâcheuses, mais parfois un résultat aberrant sera admis sans hésitation. A l'opposé, l'étudiant n'utilise que peu de résultats d'arithmétique, et presque exclusivement des résultats élémentaires ; cependant, il est parfaitement "à l'aise" dans ce petit domaine ; en particulier, sa perception intuitive des nombres lui permettra le plus souvent de détecter un résultat aberrant (dû à une erreur de signe, d'un facteur 10, etc.).

Le point crucial est que l'étudiant doit assimiler la logique élémentaire comme l'arithmétique élémentaire ; il doit pouvoir "doubler" le raisonnement méthodique et rigoureux par une compréhension intuitive des formules. Il doit arriver, par exemple, à *rejeter* l'énoncé de logique (incorrect !)

Si $A \Rightarrow B$ est vrai, alors $(X \Rightarrow (A \Rightarrow Y)) \Rightarrow (X \Rightarrow (B \Rightarrow Y))$ est vrai.

aussi rapidement que l'énoncé algébrique (incorrect !)

Si $a \leq b$ est vrai, alors $(y - a) - x \leq (y - b) - x$ est vrai.

En mathématique, il est extrêmement pénible de mémoriser des démonstrations vues comme des textes linéaires dont tous les mots ont la même importance. Il est de loin préférable d'associer à un théorème un objet concret (au sens large) à partir duquel on peut reconstituer aisément la démonstration du théorème.

Le dessin de gauche de la figure 1 comporte deux carrés intérieurs dont les dimensions sont a et b ainsi que deux rectangles de côtés a et b. Ce dessin illustre notamment la formule $(a+b)^2 = a^2 + 2ab + b^2$. Le dessin de droite comporte un carré intérieur de dimension c, ainsi que quatre triangles rectangles de petits côtés a et b et d'hypoténuse c. L'aire totale des deux rectangles étant égale à celle des quatre triangles, l'aire totale $a^2 + b^2$ des deux carrés intérieurs

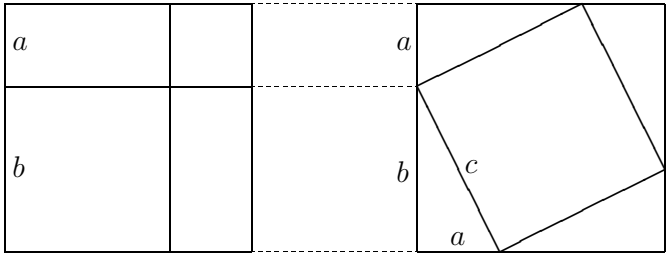


FIG. 1 – Clef du théorème de Pythagore.

à gauche est égale à l'aire c^2 du carré intérieur à droite. Cette dernière égalité est le théorème de Pythagore.

Deux pistes prometteuses ... Ce genre d'objet (ici, une paire de dessins) est naturellement très utile, mais il n'est pas évident de le découvrir. C'est cependant moins difficile en logique formelle qu'en algèbre ou en analyse, et notre principal objectif, en écrivant ce texte, est de *montrer comment ces objets peuvent être découverts et utilisés* ; ce seront souvent des objets bien connus de l'informaticien, tels les tableaux, les listes et les arbres. Nous essayerons aussi *d'arithmétiser la logique*, ce qui permet d'importer en logique l'expérience et l'intuition acquises en arithmétique élémentaire. En particulier, les démonstrations paraîtront simples au lecteur qui observera leur caractère constructif et inductif. Le plus souvent, une démonstration donne lieu à un programme (récursif) construisant l'objet dont le théorème démontré affirme l'existence.

1 Introduction

Dans ce bref chapitre, on présente les objectifs de la logique. On montre certaines analogies existant entre la logique et l'arithmétique. On montre l'utilité de la logique pour une meilleure compréhension des théorèmes et des programmes, et aussi pour l'écriture même de certains programmes.

Qu'est-ce que la logique ? La logique est la science du raisonnement et de l'expression précise du raisonnement. Tout être humain raisonne et est donc concerné par la logique. Raisonner, c'est produire de l'information à partir d'information préexistante et de certains mécanismes de transformation de l'information.

Raisonnement et calcul. Le raisonnement est proche du calcul, qui lui aussi transforme l'information. Etant donné un triangle dont la base et la hauteur sont de 6 cm (information préexistante), on sait que l'aire du triangle est de 18 cm^2 ("nouvelle" information). Le mécanisme de production est la règle classique $S = (B \times H)/2$.

La logique la plus simple est celle des propositions. Il s'agit bien d'un calcul, dans lequel les objets ne sont pas les nombres et les expressions numériques, mais les valeurs de vérité ("vrai" et "faux") et les propositions et formules susceptibles d'être vraies ou fausses. Voici un exemple typique de ce calcul. L'information préexistante comporte deux énoncés :

> Pour sortir sous la pluie, je prends mon parapluie.
> Je suis dehors sans parapluie.

Le mécanisme de calcul, ou plutôt de déduction, est le suivant

$$\frac{A \Rightarrow B\,,\ \neg B}{\neg A}$$

"Si A implique B est vrai, et si B est faux, alors A est faux." L'information nouvelle que l'on peut obtenir ici est "Il ne pleut pas". On a *instancié* A en "il pleut" et B en "je sors muni d'un parapluie".
On notera l'emploi de la convention habituelle : la ligne horizontale sépare les *prémisses* d'un raisonnement (au-dessus de la ligne) et sa *conclusion* (au-dessous de la ligne).

Logique et arithmétique. Dans le raisonnement précédent, le calcul proprement dit est extrêmement simple. C'est une arithmétique des plus rudimentaires, où on ne dispose que de deux "nombres", notés **F** (faux) et **V** (vrai), ou encore, pour parfaire l'analogie, 0 et 1. Les tables correspondant à l'implication et aux quelques autres opérations logiques seront donc bien plus simples que la table de multiplication par exemple, puisque les tables logiques ne comportent que deux entrées. Une difficulté de la logique par rapport à l'arithmétique est le caractère informel du langage utilisé (le français). L'information représentée par B peut être écrite "je sors muni d'un parapluie" ; on a donc implicitement admis que la phrase "Pour sortir sous la pluie, je prends mon parapluie." est synonyme de "Il pleut implique je sors muni d'un parapluie". On conçoit aisément que, dans des raisonnements plus élaborés, une hypothèse

de ce type peut rapidement devenir douteuse.[1] Toutefois, ce problème est du ressort de la linguistique et nous ne l'aborderons pas ici. Plus précisément, nous ne considérerons pas de raisonnement dont la formalisation ne soit élémentaire, voire même déjà faite. L'objet de la logique mathématique sera donc la représentation et l'analyse des raisonnements formalisés. A titre d'exemple, considérons les tables de la négation et de l'implication :

x	$\neg x$
V	**F**
F	**V**

x	y	$x \Rightarrow y$
V	**V**	**V**
V	**F**	**F**
F	**V**	**V**
F	**F**	**V**

Ces tables permettent d'établir la validité du mécanisme de raisonnement utilisé au paragraphe précédent. Par simple combinaison, on obtient immédiatement la table ci-dessous :

A	B	$A \Rightarrow B$	$\neg B$	$\neg A$
V	**V**	**V**	**F**	**F**
V	**F**	**F**	**V**	**F**
F	**V**	**V**	**F**	**V**
F	**F**	**V**	**V**	**V**

Valider le mécanisme de raisonnement utilisé dans notre exemple (c'est le "Modus Tollens") consiste à vérifier que, dans tous les cas où les deux prémisses $A \Rightarrow B$ et $\neg B$ sont vraies, la conclusion $\neg A$ est également vraie. Dans les trois premières lignes du tableau, l'une des prémisses est fausse. Ces lignes correspondent à des cas où le Modus Tollens ne s'applique pas. La quatrième ligne correspond au cas où les deux prémisses sont vraies, c'est-à-dire au cas où le mécanisme de raisonnement étudié s'applique ; on observe que la conclusion est également vraie, ce qui achève la vérification.

Trop simple, la logique ? On peut se demander à quoi sert la logique en tant que science, puisqu'elle ne recouvre, dans le contexte élémentaire dont nous ne sortirons pas, que des connaissances évidentes. Nous aurons amplement l'occasion de souligner plus loin qu'il est certaines évidences méritant d'être soulignées mais, pour l'informaticien en particulier, l'utilité de la logique est du même ordre que celle de l'arithmétique. Cette utilité est liée à la dimension des problèmes traités, et à l'intérêt, au delà d'une certaine taille très vite atteinte, d'automatiser la résolution de ces problèmes. On "produit" $(6 \times 6)/2 = 18$ sans effort, mais il n'en va pas de même pour $345\,234 \times 765\,864 = 264\,402\,292\,176$; on utilisera ici une calculatrice, ou un ordinateur dont la programmation a requis la mise en œuvre de règles arithmétiques bien formalisées. Plus peut-être que la taille des problèmes, c'est leur généralisation qui requiert l'élaboration d'une science. L'égalité $1 + 2 + \cdots + 10 = 55$ présente un intérêt nettement moindre que l'égalité $\sum_{i=1}^{n} i = (n \times (n+1))/2$. Cette égalité ne se déduit pas seulement de tables arithmétiques, si détaillées soient-elles ; une "science" arithmétique est nécessaire. Il

[1] Même dans notre exemple, certains problèmes surgissent. En particulier, il peut avoir commencé à pleuvoir après que je sois sorti (sans parapluie) ; je peux aussi égarer mon parapluie en cours de route.

en va de même en logique, où l'on formalise des règles générales, telle la classique règle de récurrence :

$$\frac{P(0)\,,\ \forall n\,[P(n) \Rightarrow P(n+1)]}{\forall n\,P(n)}$$

L'effet de dimension est également présent en logique, comme en témoignent deux petites énigmes amusantes.

1. *Les étudiants ayant participé à l'examen de logique diffèrent par le le prénom, la nationalité et le sport favori pratiqué par chacun d'eux. On demande de reconstituer le classement et de déterminer qui est le Français et quel est le sport pratiqué par Richard, sur base des indices suivants.*

1. *Il y a trois étudiants.*
2. *Michel joue au football.*
3. *Michel est mieux classé que l'Américain.*
4. *Simon est Belge.*
5. *Simon a surclassé le joueur de tennis.*
6. *Le nageur s'est classé premier.*

2. *Les occupants de maisons alignées diffèrent par la nationalité, la couleur de la maison, la marque de cigarette favorite, la boisson préférée et l'animal familier. On demande de reconstituer la situation, et en particulier d'identifier le propriétaire du zèbre et le buveur d'eau, sur base des indices suivants.*

1. *Les numéros des maisons sont 1, 2, 3, 4, 5.*
2. *L'Anglais habite la maison verte.*
3. *L'Espagnol possède un chien.*
4. *On boit du café dans la maison rouge.*
5. *On boit du thé chez l'Ukrainien.*
6. *La maison rouge suit la maison blanche.*
7. *Le fumeur de Old Gold élève des escargots.*
8. *On fume des Gauloises dans la maison jaune.*
9. *On boit du lait au numéro 3.*
10. *Le Norvégien habite au numéro 1.*
11. *Le fumeur de Chesterfield et le propriétaire du renard sont voisins.*
12. *Le fumeur de Gauloises habite à côté du propriétaire du cheval.*
13. *Le fumeur de Lucky Strike boit du jus d'orange.*
14. *Le Japonais fume des Gitanes.*
15. *La maison bleue jouxte celle du Norvégien.*

Formaliser ces énigmes, c'est-à-dire les convertir en formules à analyser, est facile. L'analyse proprement dite est tout aussi facile, mais, vu la taille des formules, peut demander un certain temps, et une bonne organisation du travail.[2] Naturellement, on peut programmer un ordinateur pour faire le travail ; nous prêterons une attention particulière aux questions algorithmiques.

Mieux comprendre les théorèmes. La logique formelle a été au départ développée par des mathématiciens, pour éclairer divers problèmes délicats survenant en algèbre, en analyse, en géométrie, etc. Ce point n'a pas tellement d'intérêt pour l'informaticien, mais il importe

[2] Si on n'est pas habitué à résoudre ce type d'énigme, on peut s'attendre à une demi-heure de tâtonnement avant de découvrir la solution de la seconde, même si une minute suffit pour la première …

de reconnaître d'emblée le statut acquis par la logique mathématique : elle contribue au développement d'autres branches des mathématiques et favorise une meilleure compréhension de celles-ci. Inversement, une certaine maturité mathématique favorise l'apprentissage de la logique.

Nous ne donnerons ici qu'un exemple de la symbiose entre logique et mathématique, mais il est capital. Dans n'importe quelle branche des mathématiques, un théorème évoque une catégorie d'objets et affirme que tout objet vérifiant l'hypothèse vérifie aussi la thèse. Ceci est un exemple typique d'évidence qu'il est opportun de souligner. Dans le domaine $\mathbb{Z}$ des entiers relatifs, on a le théorème suivant :

Tout carré est positif.

La paraphrase suivante donne lieu à une traduction immédiate :

Pour tout n, n est un carré implique n est positif.

On peut en effet formaliser l'énoncé en

$$\forall n\,[C(n) \Rightarrow P(n)]\,.$$

Comme souvent en mathématique, on sous-entend une partie de l'information ; dans le cas présent, la formule ne reprend pas (notamment) le fait que n représente un entier relatif et non, par exemple, un nombre complexe. Ce théorème exprime que, des quatre classes d'entiers relatifs que l'on peut a priori former (C-P : carré-positif, C-nP : carré-non-positif, nC-P : non-carré-positif, nC-nP : non-carré-non-positif), la seconde est vide. On a en effet

C-P	0, 1, 4, … , 100, …
C-nP	
nC-P	2, 3, 5, … , 99, 101, …
nC-nP	-1, -2, -3, … , -100, …

Cela illustre la règle logique disant qu'une implication $p \Rightarrow q$ est fausse si et seulement si l'antécédent p est vrai et le conséquent q est faux.[3] En particulier, une implication dont l'antécédent est faux est toujours vraie. Par exemple, l'énoncé "si 2+2=5, alors 2+2=6" est vrai, ce qui ne l'empêche pas d'être sans intérêt pratique.

Mieux comprendre les programmes. La logique est utile à l'informaticien, et en particulier au programmeur. Elle permet de *spécifier* un programme :

$$\{x_0 \in \mathbb{N}\}\ (x,y) := (x_0,1)\,;\ \text{while } x>0 \text{ do } (y,x) := (y{*}x, x{-}1)\,;\ F := y\ \{F = x_0!\}\,.$$

Cette écriture exprime une relation utile entre la donnée x_0 et le résultat F. La logique permet aussi de *donner un argument de conformité* entre le programme et sa spécification, par exemple un invariant de boucle ; cet invariant est ici

$$x, x_0, y \in \mathbb{N} \ \wedge\ 0 \le x \le x_0 \ \wedge\ y * x! = x_0!$$

[3] Le théorème affirme que la seconde des quatre classes est vide ; il n'interdit pas qu'éventuellement une des trois autres classes soit vide aussi.

Enfin, la logique permet de *vérifier la validité de l'argument*. Un élément crucial de cette vérification est le fait que le corps de la boucle, lorsqu'il est exécuté, restaure l'invariant. En appelant cet invariant I, on doit avoir

$$\{I \ \wedge\ x>0\}\ (y,x) := (y{*}x, x{-}1)\ \{I\}\,,$$

ou encore

$$(I \ \wedge\ x>0) \Rightarrow I[y,x\,/\,y{*}x, x{-}1]$$

où $I[y,x\,/\,y{*}x,x{-}1]$ désigne la formule I dans laquelle on a remplacé les occurrences de y et de x par $y*x$ et $x-1$, respectivement. En définitive, pour établir que le programme est correct, il faut prouver que la formule

$$\forall x\,\forall x_0 \forall y\,([0<x\le x_0 \ \wedge\ y*x! = x_0!] \Rightarrow [0\le x{-}1 \le x_0 \ \wedge\ (y{*}x)*(x{-}1)! = x_0!])$$

est vraie, ce qui peut se faire en utilisant les propriétés arithmétiques usuelles, notamment l'associativité de la multiplication et le fait que $n*(n-1)! = n!$ pour tout entier n strictement positif. Le fait que la logique soit plutôt simple rend possible l'automatisation du raisonnement, qui est vu comme un calcul d'un genre particulier. La logique est donc une clef de l'intelligence artificielle et permet en particulier la programmation de systèmes experts, aptes à la résolution automatique d'énigmes comme celle du zèbre ou, d'une manière moins ludique, à l'élaboration de diagnostic de pannes dans les réseaux informatiques, pour ne donner qu'un exemple.

Un algorithme est une recette de calcul permettant de résoudre un problème sans devoir réfléchir. Toute la réflexion nécessaire a été anticipée par l'auteur de l'algorithme, ce qui explique la difficulté potentielle de la tâche du concepteur d'algorithme. Un programme de calcul implique des règles de calcul et la mise en œuvre de ces règles. Dans un programme classique, tel celui calculant la factorielle, ces deux ingrédients sont intimement mélangés, et les règles mathématiques de base qui ont été utilisées (l'associativité de la multiplication, par exemple) n'apparaissent explicitement que lors d'une vérification systématique et détaillée de l'exactitude de l'algorithme. Dans la mesure où calcul et raisonnement ne sont que deux facettes d'un même processus, on peut envisager de séparer les deux ingrédients. Un algorithme de mise en œuvre de règles logico-mathématiques est écrit une fois pour toutes, et cet algorithme est particularisé à un problème particulier par l'adjonction des règles relatives à ce problème. Cette technique de "programmation (par la) logique" est très puissante, notamment dans les cas où la programmation classique est décevante. Dans ce contexte, pour trier un tableau X en un tableau Y, il suffira de donner deux "indices" :

Y est une permutation de X ;
les éléments de Y forment une suite croissante.

Trier de cette manière sera cependant très inefficace. Dans le même contexte de programmation logique, il suffira de donner les indices et la question de l'énigme évoquée plus haut pour résoudre celle-ci. Dans la mesure où on ne connaît pas d'algorithme classique de résolution d'énigme, l'approche logique est ici très attrayante.

Programmer en logique. Puisque la logique est, dans une certaine mesure, un calcul, elle peut donner naissance à un langage de programmation. Le langage PROLOG ("PROgrammer en LOGique") est le plus utilisé des langages basés sur la logique. En dépit de certaines

limitations, il se prête bien à la résolution d'une vaste classe de problèmes. A titre d'exemple, nous donnons à la page suivante un programme PROLOG pour la résolution de l'énigme du zèbre donnée plus haut. Ce programme décrit d'une part ce qu'est une énigme et, d'autre part, les particularités de l'énigme du zèbre. Le système PROLOG calcule la réponse en utilisant les algorithmes de résolution et d'unification, présentés à la fin de ce cours.

```
prc(A,B,[A,B,C,D,E]).  prc(A,C,[A,B,C,D,E]).  prc(A,D,[A,B,C,D,E]).
prc(A,E,[A,B,C,D,E]).  prc(B,C,[A,B,C,D,E]).  prc(B,D,[A,B,C,D,E]).
prc(B,E,[A,B,C,D,E]).  prc(C,D,[A,B,C,D,E]).  prc(C,E,[A,B,C,D,E]).
prc(D,E,[A,B,C,D,E]).

one(A,[A,B,C,D,E]).
three(C,[A,B,C,D,E]).

neighbor(A,B,[A,B,C,D,E]).  neighbor(B,C,[A,B,C,D,E]).
neighbor(C,D,[A,B,C,D,E]).  neighbor(D,E,[A,B,C,D,E]).
neighbor(B,A,[A,B,C,D,E]).  neighbor(C,B,[A,B,C,D,E]).
neighbor(D,C,[A,B,C,D,E]).  neighbor(E,D,[A,B,C,D,E]).

nation(h(N,C,A,B,T),N).
color(h(N,C,A,B,T),C).
animal(h(N,C,A,B,T),A).
drink(h(N,C,A,B,T),B).
tobacco(h(N,C,A,B,T),T).

go(X,Y) :- St = [h(N1,C1,A1,B1,T1),h(N2,C2,A2,B2,T2),
                 h(N3,C3,A3,B3,T3),h(N4,C4,A4,B4,T4),h(N5,C5,A5,B5,T5)],
  member(X2,St), nation(X2,english), color(X2,green),
  member(X3,St), nation(X3,spanish), animal(X3,dog),
  member(X4,St), color(X4,red), drink(X4,coffee),
  member(X5,St), nation(X5,ukrainian), drink(X5,tea),
  neighbor(X6a,X6b,St), prc(X6b,X6a,St), color(X6a,red), color(X6b,white),
  member(X7,St), tobacco(X7,oldgold), animal(X7,snails),
  member(X8,St), color(X8,yellow), tobacco(X8,gauloises),
  three(X9,St), drink(X9,milk),
  one(X10,St), nation(X10,norwegian),
  neighbor(X11a,X11b,St), tobacco(X11a,chesterfield), animal(X11b,fox),
  neighbor(X12a,X12b,St), tobacco(X12a,gauloises), animal(X12b,horse),
  member(X13,St), tobacco(X13,luckystrikes), drink(X13,orangejuice),
  member(X14,St), nation(X14,japanese), tobacco(X14,gitanes),
  neighbor(X15a,X15b,St), nation(X15a,norwegian), color(X15b,blue),
  member(Q,St), animal(Q,zebra), nation(Q,X),
  member(R,St), drink(R,water), nation(R,Y).
```

On observe que ce texte ressemble plus à une variante de l'énoncé du problème qu'à un programme pour résoudre le problème. Il n'est en fait qu'une donnée pour la version Prolog des algorithmes de résolution et d'unification.

Le texte est composé de *clauses* qui décrivent des prédicats. Il y a deux sortes de clauses :

```
a.
a :- b,c,d.
```


La première exprime un fait (axiome, postulat, définition). Elle peut se lire "On a `a`" ou "`a` est (toujours) vrai". La seconde clause exprime une règle, la possibilité d'obtenir le fait `a` à partir des faits `b`, `c` et `d`. On peut lire "Si `b`, `c` et `d` sont vrais, alors `a` est vrai", ou encore "Pour avoir (établir) `a`, il suffit d'avoir (d'établir) `b`, `c` et `d`".

Les clauses préliminaires constituent des définitions auxiliaires, pour les notions de précédence (une maison précède une autre si le numéro de la première est plus petit que celui de la seconde), de première maison, de maison du milieu et de maisons voisines. On note par exemple que, dans une structure de cinq éléments `[A,B,C,D,E]`, les maisons `A` et `B` sont mitoyennes, de même que `B` et `A`, `B` et `C`, ... et enfin `D` et `E`. On précise aussi qu'une maison est décrite par cinq attributs qui sont, dans l'ordre, la nationalité du propriétaire, la couleur de la façade, l'animal familier, la boisson favorite et la marque de tabac.

La clause principale définit le prédicat `go`. Les cinq premières lignes correspondent à l'indice 1 ; les lignes suivantes correspondent aux quatorze autres indices, sauf les deux dernières lignes qui correspondent aux deux questions. A titre d'exemple, voici une paraphrase du dernier indice : "la structure `St` comporte deux maisons mitoyennes `X15a` et `X15b` telles que l'occupant de `X15a` est de nationalité norvégienne, et que `X15b` est de couleur bleue". La première question se traduit en "la structure `St` comporte une maison (inconnue) `Q`, dont l'occupant est de nationalité `X` et possède un zèbre".

L'exécution de ce programme est représentée ci-dessous.

```
?- go(ZebraOwner,WaterDrinker).
ZebraOwner = japanese
WaterDrinker = norwegian ? ;
no
```

Le "`no`" indique l'absence d'une seconde solution ; sur base des indices, il est donc certain que le Japonais possède le zèbre et que le Norvégien boit de l'eau.

Notre but n'est pas ici de présenter Prolog, mais de montrer que les algorithmes logiques que nous étudierons sont suffisamment puissants pour prendre en charge la résolution d'un problème non trivial. Ces algorithmes sont préprogrammés efficacement et une fois pour toutes dans le système Prolog, mais peuvent naturellement être programmés dans n'importe quel langage ; nous verrons d'ailleurs comment, aux chapitres trois et cinq.

Nous terminons ce chapitre par une très brève introduction aux deux algorithmes utilisés par Prolog. Considérons le petit programme suivant.

```
a.
b.
c :- a.
d :- a,f.
d :- b,c.
e :- c,f.
```

Le logicien écrira plutôt

$$\{A\,,\ B\,,\ A \Rightarrow C\,,\ (A \wedge F) \Rightarrow D\,,\ (B \wedge C) \Rightarrow D\,,\ (C \wedge F) \Rightarrow E\}\,.$$

La dernière formule, par exemple, signifie que si C et F sont vrais, alors E est vrai. Essayons, sur base de nos six clauses, de voir si D est vrai, et si E est vrai. Pour avoir D, d'après la cinquième clause,[4] il suffit d'avoir B et C. On a B (deuxième clause) et, pour avoir C, il suffit (troisième clause) d'avoir A, que l'on a par la première clause. La réponse à la première question est donc "oui". D'autre part, pour avoir E, il suffit d'avoir C et F. On a bien C (on a vu comment), mais aucune clause ne permet d'espérer obtenir F. La réponse à la seconde question est donc "non". Ces raisonnements, illustrés à la figure 2, sont des exemples typiques de ce qu'étudie la *logique des propositions*, abordée aux chapitres deux et trois. Les numéros des nœuds des arbres de la figure 2 indiquent l'ordre dans lequel ces nœuds sont créés et exploités.

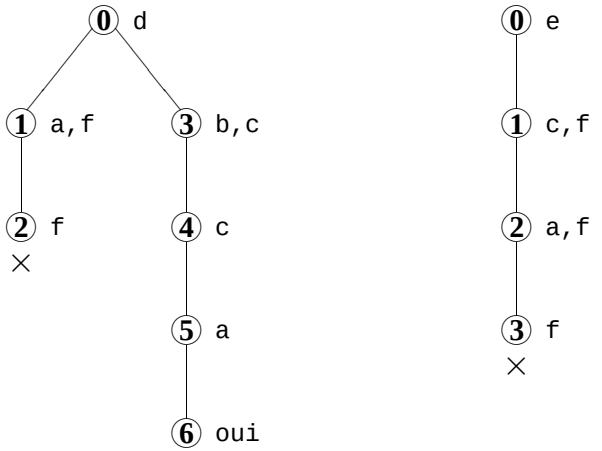


FIG. 2 – Arbres de vérification systématique de faits.

Considérons maintenant un autre programme.

```
append([],Ys,Ys).
append([X|Xs],Ys,[X|Zs]) :- append(Xs,Ys,Zs).
```

La notation `append(Xs,Ys,Zs)` signifie "la concaténation de la liste `Xs` et de la liste `Ys` est la liste `Zs`". Le programme ci-dessus correspond donc à la définition récursive classique de l'opération de concaténation de deux listes. (L'écriture `[X|Xs]` représente la liste dont le premier élément est `X` et dont le reste est la liste `Xs`.) On peut demander à Prolog

```
? append(Xs,Ys,[a,b]).
```

Cela signifie "Existe-t-il des listes `Xs` et `Ys` dont la concaténation donne la liste `[a,b]` ?". Prolog cherche une preuve constructive de l'énoncé, en pratiquant l'*unification* des termes. En particulier, il établit que l'on a bien `append(Xs,Ys,[a,b])`, si l'on choisit `Xs` = `[a]` et `Ys` = `[b]`. Cette recherche systématique produit trois solutions ; elle est illustrée à la figure 3. C'est une application typique de la *logique des prédicats*, abordée aux chapitres quatre et cinq.

[4] Utiliser la quatrième clause conduit à une impasse, comme on on le voit à la figure 2.

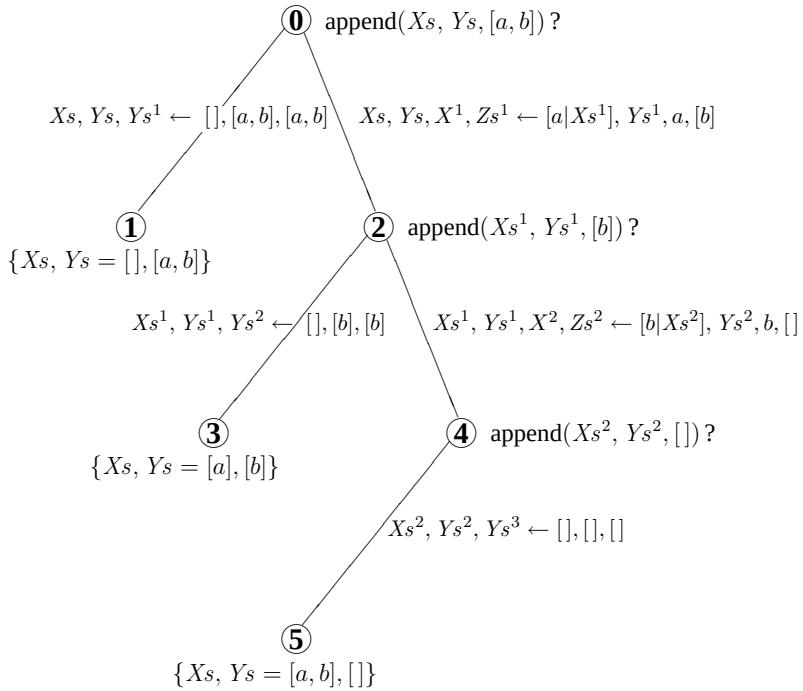


FIG. 3 – Arbre de recherche systématique des solutions.

2 Logique propositionnelle : syntaxe et sémantique

2.1 Introduction

L'objet de la logique propositionnelle est l'étude des propositions et de certaines opérations qui permettent de les combiner. Nous montrons dans cette section d'où proviennent ces objets et ce qu'ils sont.

2.1.1 Généralités sur les propositions

Une *proposition* est une phrase susceptible d'être vraie ou fausse. En français, on parle souvent de "phrase énonciative". Voici quelques exemples de propositions, écrites en langage naturel, en formalisme mathématique ... ou dans un mélange des deux.

1. Un plus deux égalent trois.
2. 1+1 = 3.

3. $\pi > e$.
4. Il n'existe que cinq polyèdres réguliers convexes.
5. Il existe une infinité de nombres premiers x tels que $x + 2$ est aussi premier.
6. Le procompsognathus est un deutérostomien anamniote.
7. Il fera beau à Liège le 29 avril de l'an 2021.
8. $x^2 + y^2 = z^2$.
9. Il pleut.
10. Je donne cours de logique le mardi.
11. Je_donne_cours(matière, jour).
12. Un plus deux égalent trois et la terre tourne autour du soleil.
13. Un plus deux égalent trois parce que la terre tourne autour du soleil.
14. Cette phrase est fausse.
15. La phrase suivante est vraie.
16. La phrase précédente est fausse.
17. Ceci n'est pas une phrase.
18. Cette phrase n'est pas une proposition.

On observe immédiatement que la nature propositionnelle d'une phrase n'implique pas la connaissance automatique de la valeur de vérité de cette phrase. Les trois premiers exemples paraissent non problématiques, mais on pourrait hésiter à leur attribuer une valeur de vérité par méconnaissance du français ou de l'arithmétique élémentaire ; on pourrait aussi ignorer ou refuser les conventions habituelles des mathématiciens concernant les constantes numériques importantes, telles $\pi = 3.14159...$ et $e = 2.71828...$. Les exemples 4 à 7 montrent que l'ignorance est une cause excusable et même inévitable de non-attribution. On peut savoir ce qu'est un polyèdre régulier convexe, mais le non-mathématicien ignore généralement leur nombre. L'exemple 5 a un sens clair, mais il s'agit d'une conjecture de l'arithmétique : les spécialistes pensent qu'elle est vraie, mais n'ont pas réussi à la démontrer. Le sens de la phrase 6 et a fortiori sa valeur de vérité échapperont au non-biologiste. Enfin, faire une prévision météorologique à très long terme est complètement irréaliste.

Les exemples 8 à 11 posent une difficulté d'une autre nature. Les phrases sont claires et élémentaires, mais leur valeur de vérité dépend du contexte. La pertinence de ce contexte peut apparaître explicitement, via des paramètres tels "x", "y", "z", "matière", "jour", ou de manière implicite : qui est "Je" ? Où et quand est prononcée la phrase "Il pleut" ? On ne peut attribuer une valeur de vérité à ces exemples sans connaître leur contexte.

Les exemples 1 à 11 étaient des propositions *atomiques* ; les exemples 12 et 13 sont des propositions *composées* ; les deux composants sont les propositions atomiques "Un plus deux égalent trois" et "La terre tourne autour du soleil". Ces composants sont *connectés* par "et" et par "parce que".

Les exemples 14 à 16 sont des *paradoxes* ; on les comprend, aucune culture et contexte particuliers ne sont nécessaires à leur analyse, mais il est néanmoins impossible de leur attribuer une valeur de vérité sans aboutir à une contradiction (le cauchemar du logicien !).

Ce phénomène est lié à l'*autoréférence* : ces phrases parlent d'elles-mêmes. Les exemples 17 et 18 montrent que l'autoréférence n'implique pas toujours le paradoxe : il s'agit bien de deux propositions, toutes deux fausses.

Dans l'approche formelle de la logique propositionnelle, nous voulons nous affranchir de tous les problèmes non liés au raisonnement proprement dit.[5] Un moyen radical d'y parvenir est de restreindre le sens d'une proposition à sa valeur de vérité, exactement comme, en arithmétique, le sens d'une multiplicité est réduit à sa taille. On évoque le nombre "13" par exemple, sans se soucier d'évoquer une multiplicité concrète comportant 13 éléments. En fait, l'arithmétique ne nous apprend rien sur les nombres eux-mêmes, mais a pour objet les relations qui existent entre les nombres, dont l'existence est tout simplement postulée et admise. Le lexique de l'arithmétique se limite donc aux notations identifiant les nombres (suites de chiffres), aux variables représentant les nombres (souvent des lettres, x, y, a, b, etc.) et aux symboles représentant les opérations par lesquelles on peut combiner et comparer les nombres ($+$, $=$, $<$, etc.). L'écriture $x^2 + y^2 = z^2$ est un énoncé de l'arithmétique ; c'est aussi une proposition. Pour attribuer une valeur de vérité à cette proposition, il faut connaître les valeurs (numériques) de x, y et z, mais rien d'autre (inutile, par exemple, de savoir si les nombres en questions représentent des longueurs, ou des vitesses, ou des nombres de pommes contenues dans des paniers).

En arithmétique, on distingue les énoncés *valides*, qui sont toujours vrais, les énoncés *inconsistants*, ou *contradictoires*, qui sont toujours faux, et les énoncés *contingents*, ou *simplement consistants*, qui sont vrais ou faux selon le contexte, c'est-à-dire selon les valeurs que l'on attribue aux variables qu'ils contiennent. Un énoncé valide peut contenir des variables, tel $x^2 + y^2 \geq 2xy$ ou n'évoquer que des constantes, tel $2 + 3 = 5$. Il en va de même pour les énoncés inconsistants ($x^2+y^2 < 2xy$, $2+3 = 6$). En revanche, un énoncé contingent comporte toujours une variable au moins ($x < 3$).

La même classification sera adoptée en logique propositionnelle. Les énoncés *true* et $false \Rightarrow p$ sont valides ; seul le second comporte une variable propositionnelle. L'énoncé $p \Rightarrow q$ est contingent, tandis que l'énoncé $true \Rightarrow false$ est contradictoire.

Deux différences essentielles existent entre la logique et l'arithmétique. Tout d'abord, il n'existe que deux valeurs en logique, contre une infinité en arithmétique ; de plus, la notion de proposition existe en arithmétique (les énoncés arithmétiques sont des propositions, au même titre que les énoncés de mécanique des fluides, par exemple), alors que la notion de nombre n'apparaît pas en logique propositionnelle. En fait, la logique "précède" l'arithmétique, car on ne peut pas faire d'arithmétique sans faire, consciemment ou non, de la logique. En contrepartie, l'arithmétique est "plus riche" que la logique ; on pourra identifier le calcul des propositions à un calcul numérique particulier, mais on ne pourra pas identifier le calcul sur les nombres à une logique propositionnelle particulière.

[5] D'après Larousse, la logique est la "science du raisonnement en lui-même, abstraction faite de la matière à laquelle il s'applique et de tout processus psychologique".

2.1.2 Généralités sur les connecteurs

Les opérateurs combinant les propositions sont appelés *connecteurs*. La logique des propositions est en fait la logique des connecteurs, comme l'arithmétique est plus la science des opérations (addition et multiplication surtout) que celle des nombres proprement dits.

Les opérations de l'arithmétique (au sens large) sont des fonctions dont les arguments (en général, un ou deux) prennent des valeurs numériques ; la valeur du résultat est numérique, ou une valeur de vérité. Voici quelques opérations courantes :

- L'addition : $+ : \mathbb{Z} \times \mathbb{Z} \longrightarrow \mathbb{Z} : (x, y) \mapsto x + y.$
- Le passage à l'opposé : $- : \mathbb{Z} \longrightarrow \mathbb{Z} : x \mapsto -x.$
- La divisibilité : $| : \mathbb{Z} \times \mathbb{Z}_0 \longrightarrow \{\mathbf{V}, \mathbf{F}\} : (x, y) \mapsto x|y.$
- La primarité : $\mathrm{Pr} : \mathbb{N}_{0,1} \longrightarrow \{\mathbf{V}, \mathbf{F}\} : x \mapsto \mathrm{Pr}(x).$
  Rappelons que, si x est un nombre entier plus grand que 1, $\mathrm{Pr}(x)$ est vrai si x est premier, c'est-à-dire n'est divisible que par lui-même et par 1.

Il existe une infinité d'opérations arithmétiques, et le mathématicien s'autorisera à en créer une nouvelle, qu'il nommera et notera de manière appropriée, dès que le besoin s'en fera sentir. Toutefois, quand on étudie l'arithmétique, on se limite généralement à une demi-douzaine d'opérations. On retient, d'une part, celles dont l'intérêt pratique est évident et, d'autre part, celles dont les propriétés sont les plus attrayantes et les plus élégantes. Ces deux critères sont souvent concordants ; de plus, les opérations non retenues comme primitives peuvent souvent se dériver des opérations primitives … au moyen de la logique. On définit par exemple la divisibilité (ne pas confondre avec la division) à partir de l'égalité et de la multiplication :

$$x|y \ =_{def} \ \exists z \, (x.z = y) \, .$$

On se sert de cette nouvelle opération pour définir la primarité :

$$\mathrm{Pr}(x) \ =_{def} \ x > 1 \ \wedge \ \forall y \, [y|x \ \equiv \ (y = 1 \vee y = x)] \, .$$

En logique propositionnelle aussi, nous aurons des connecteurs fondamentaux et des connecteurs dérivés.

Notons aussi que certaines "opérations" arithmétiques ne sont pas considérées comme telles par les mathématiciens, parce qu'elles dépendent non seulement des nombres eux-mêmes mais aussi de points annexes, par exemple le formalisme utilisé pour les représenter. Ainsi, la "longueur" d'un nombre n'est pas une véritable opération arithmétique, puisqu'elle n'est pas "numérifonctionnelle" :

$$\begin{array}{l} \ell(13) = 2 \, , \\ \ell(6 + 7) = 3 \, , \\ \ell(78/6) = 4 \, , \\ \ell(\mathsf{1101}) = 4 \, , \\ \ell(\mathsf{treize}) = 6 \, , \\ \ell(\mathsf{thirteen}) = 8 \, , \\ \ell(\mathsf{XIII}) = 4 \, . \end{array}$$

En logique aussi, les connecteurs non "vérifonctionnels" seront éliminés.

Les connecteurs propositionnels sont nombreux dans la langue française ; nous en avons rencontré deux exemples :

- Un plus deux égalent trois *et* la terre tourne autour du soleil.
- Un plus deux égalent trois *parce que* la terre tourne autour du soleil.

Le connecteur *et* est vérifonctionnel : la proposition "A et B" sera vraie si et seulement si les propositions "A" et "B" sont toutes deux vraies. En revanche, il n'est pas évident d'établir un éventuel lien de cause à effet entre deux faits, et connaître les valeurs de vérité de "A" et de "B" ne permet généralement pas de connaître la valeur de vérité de "A parce que B". Les propositions composées suivantes, dont nous supposons les composantes vraies, montrent que le connecteur "parce que" n'est pas vérifonctionnel :

- La voiture dérape *parce que* la route est mouillée.
- La route est mouillée *parce que* la voiture dérape.

Dans un contexte où une voiture a dérapé sur une route mouillée, la première proposition composée semble vraie mais la seconde est quasi certainement fausse. Or, les deux propositions simples contenues dans les propositions composées sont vraies ; cela montre que la valeur de vérité d'une proposition composée avec "parce que" ne dépend pas uniquement des valeurs de vérité des composantes.

Voici quelques exemples d'emploi des connecteurs vérifonctionnels les plus fréquemment utilisés en français.

- J'irai au théâtre *ou bien* j'irai au cinéma.
- Il pleut *ou* il vente.
- *S*'il pleut, *alors* la route est mouillée.
- Le ciel est bleu *et* la neige est blanche.
- Il *n'*est *pas* bête.
- *Si* c'est pile *alors* je gagne *sinon* tu perds !
- Elle réussit *si* elle travaille.
- Elle réussit *seulement si* elle travaille.
- Elle réussit *si et seulement si* elle travaille.
- Elle travaille, *donc* elle réussit.
- [Ils n'ont] *ni* Dieu, *ni* maître !

Dans ces exemples, la valeur de vérité de la proposition composée se déduit aisément de la valeur de vérité des composants ; les connecteurs sont donc bien vérifonctionnels. Dans ce cadre, il est possible de rendre compte de la validité de certains raisonnements, tel le suivant :

1. Tous les hommes sont mortels.
2. Si tous les hommes sont mortels et si Socrate est un homme,
   alors Socrate est mortel.
3. Socrate est un homme.
4. Donc, Socrate est mortel.

Ce raisonnement est une *instance*, c'est-à-dire un exemple, du schéma

$$\frac{A \, , \ (A \wedge B) \Rightarrow C \, , \ B}{C}$$

et nous verrons plus loin que toutes les instances de ce schéma (qui comporte trois prémisses et une conclusion) sont valides. On observe cependant que, en bonne logique (informelle) la prémisse

2. Si tous les hommes sont mortels et si Socrate est un homme,
   alors Socrate est mortel.

semble redondante, car elle exprime une tautologie, c'est-à-dire une évidence. Comme nous l'avons signalé plus haut, le problème est que la validité du raisonnement

1. Tous les hommes sont mortels.
3. Socrate est un homme.
4. Donc, Socrate est mortel.

ne peut pas être établie dans le cadre du calcul des propositions mais seulement dans celui, plus puissant, du calcul des prédicats. Notons enfin que, ici aussi, des curiosités linguistiques peuvent compliquer l'emploi de la logique en langage naturel. Voici un exemple classique de raisonnement qui, formellement, pourrait sembler valide mais qui, clairement, ne l'est pas.

1. Jacques est un personnage intelligent.
2. Un personnage intelligent a découvert la relativité.
3. Donc, Jacques a découvert la relativité.

En voici un autre :

1. Tout ce qui est rare est cher.
2. Une Rolls-Royce bon marché est rare.
3. Donc, une Rolls-Royce bon marché est chère.

Le calcul des prédicats classique ne pourra pas rendre compte de ces problèmes, qui sont plus du ressort de la linguistique que de la logique.

2.1.3 Les connecteurs vérifonctionnels

Le nombre d'opérations arithmétiques à n arguments est infini et, de plus, il n'est pas possible de donner une table explicite exhaustive pour les opérations arithmétiques, car les opérandes peuvent prendre une infinité de valeurs.[6] Ces limitations n'existent pas en calcul des propositions, puisqu'il n'y a pour les opérandes que deux valeurs possibles. Une table de connecteur sera explicite et exhaustive : on énumère simplement tous les cas possibles. Chaque opérande peut prendre deux valeurs ; pour un opérateur à n arguments, la table comportera donc 2^n lignes. Chaque ligne peut correspondre à un résultat vrai ou à un résultat faux ; il y aura donc 2^{2^n} connecteurs (vérifonctionnels) à n arguments. En particulier, il y a 2 constantes (opérateurs sans argument), 4 connecteurs unaires et 16 connecteurs binaires, dont les tables sont reprises aux figures 4 et 5.

Dans la suite, nous n'utiliserons que des connecteurs vérifonctionnels, simplement qualifiés de connecteurs.

[6] La table de multiplication, par exemple, ne concerne que les nombres de 1 à 10 ; des règles supplémentaires sont utilisées pour traiter les cas où les valeurs des opérandes n'appartiennent pas à cet intervalle.

x	$\circ_1$	$\circ_2$	$\circ_3$	$\circ_4$
V	**V**	**V**	**F**	**F**
F	**V**	**F**	**V**	**F**

FIG. 4 – *Les quatre connecteurs unaires.*

x	y	$\circ_1$	$\circ_2$	$\circ_3$	$\circ_4$	$\circ_5$	$\circ_6$	$\circ_7$	$\circ_8$	$\circ_9$	$\circ_{10}$	$\circ_{11}$	$\circ_{12}$	$\circ_{13}$	$\circ_{14}$	$\circ_{15}$	$\circ_{16}$
V	**V**	**V**	**V**	**V**	**V**	**V**	**V**	**V**	**V**	**F**	**F**	**F**	**F**	**F**	**F**	**F**	**F**
V	**F**	**V**	**V**	**V**	**V**	**F**	**F**	**F**	**F**	**V**	**V**	**V**	**V**	**F**	**F**	**F**	**F**
F	**V**	**V**	**V**	**F**	**F**	**V**	**V**	**F**	**F**	**V**	**V**	**F**	**F**	**V**	**V**	**F**	**F**
F	**F**	**V**	**F**	**V**	**F**	**V**	**F**	**V**	**F**	**V**	**F**	**V**	**F**	**V**	**F**	**V**	**F**

FIG. 5 – *Les seize connecteurs binaires.*

2.1.4 Les connecteurs usuels

On voit immédiatement que, des quatre connecteurs unaires, seul le troisième sera vraiment utile ; on l'appelle la *négation*. (Les autres sont les deux constantes et l'identité.) La moitié des connecteurs binaires ont reçu un nom, et un ou plusieurs symboles. Ils sont repris à la figure 6.

op.	nom	symbole	se lit
$\circ_2$	disjonction	$\vee$	ou
$\circ_3$	implication inverse	$\Leftarrow$	est impliqué par
$\circ_5$	implication	$\Rightarrow$	implique
$\circ_7$	équivalence	$\equiv$	est équivalent à
$\circ_8$	conjonction	$\wedge$	et
$\circ_9$		$\uparrow$	*nand* (en électronique)
$\circ_{10}$	ou exclusif	$\oplus$	*xor*
$\circ_{15}$		$\downarrow$	*nor* (en électronique)

FIG. 6 – *Les connecteurs binaires usuels.*

Les *tables de vérité* des connecteurs importants, les plus fréquemment utilisés, sont reprises à la figure 7.
Remarque. Les symboles utilisés pour représenter les connecteurs peuvent différer d'un ouvrage à l'autre. Nous avons adopté les notations les plus courantes ; on notera cependant que "$\supset$" est souvent utilisé au lieu de "$\Rightarrow$" ; en Prolog, le symbole "`:-`" est employé à la place de "$\Leftarrow$" et la virgule remplace la conjonction.

Disposer de nombreux connecteurs permet une expression facile et concise des propositions composées, mais rend le formalisme plus complexe, et son étude plus fastidieuse. Avec la négation et un connecteur binaire bien choisi, il est possible de tout exprimer. Supposons par exemple, comme le font souvent les mathématiciens, que les connecteurs "primitifs" sont la

x	$\neg x$
V	**F**
F	**V**

x	y	$\wedge$	$\vee$	$\equiv$	$\oplus$	$\Rightarrow$
V	**V**	**V**	**V**	**V**	**F**	**V**
V	**F**	**F**	**V**	**F**	**V**	**F**
F	**V**	**F**	**V**	**F**	**V**	**V**
F	**F**	**F**	**F**	**V**	**F**	**V**

FIG. 7 – *Les connecteurs importants.*

négation et l'implication. On peut alors introduire les autres connecteurs comme suit :

$$(a \vee b) \;=_{def}\; (\neg a \Rightarrow b)\,,\; (a \wedge b) \;=_{def}\; \neg(a \Rightarrow \neg b)\,, \ldots$$

On montre facilement que $\{\neg, \vee\}$ et $\{\neg, \wedge\}$ constituent aussi des "paires primitives" acceptables, au contraire de $\{\neg, \equiv\}$ et $\{\neg, \oplus\}$. Curieusement, le connecteur binaire $\uparrow$ permet à lui seul de définir tous les autres ; on a par exemple $\neg a \;=_{def}\; (a \uparrow a)$ et $(a \wedge b) \;=_{def}\; ((a \uparrow b) \uparrow (a \uparrow b))$. L'opérateur $\downarrow$ est le seul autre connecteur binaire jouissant de cette propriété.

En français, l'un des rares connecteurs ternaires d'usage courant est "si-alors-sinon". La proposition "si A alors B sinon C" a la valeur de B si A est vrai, et celle de C si A est faux. Ce connecteur permet lui aussi de dériver tous les autres, si on lui adjoint les constantes de base *true* et *false*.[7] Il peut lui-même s'exprimer en termes de connecteurs binaires et de la négation : "si A alors B sinon C" a même valeur de vérité que $((A \wedge B) \vee (\neg A \wedge C))$, ou encore que $((A \Rightarrow B) \wedge (\neg A \Rightarrow C))$.

On a aussi le résultat suivant.

Théorème. Tout opérateur n-aire ($n > 2$) peut se réduire à une combinaison d'opérateurs binaires et de négations.

Remarque. En logique, les théorèmes affirmant l'existence d'un certain objet se démontrent souvent de façon *constructive* ; la preuve du théorème est une méthode (un algorithme) de construction de l'objet en question. En outre, la preuve se fait souvent par *récurrence* ; cela revient à dire que l'algorithme de construction est récursif. Enfin, une fois que l'on sait cela, il suffit de mémoriser une simple ligne pour reconstituer le détail de la preuve. Dans le cas présent, cette ligne peut être

$$M(p_1, \ldots, p_n) \;\equiv\; [(p_n \wedge M(p_1, \ldots, p_{n-1}, \mathit{true})) \;\vee\; (\neg p_n \wedge M(p_1, \ldots, p_{n-1}, \mathit{false}))]\,.$$

Cette ligne est en fait une preuve du résultat suivant :

Lemme. Tout opérateur n-aire ($n > 2$) peut se réduire à une combinaison de deux opérateurs $(n-1)$-aires, d'opérateurs binaires et de négations.

Corollaire. Les connecteurs n-aires ($n > 2$) peuvent être ignorés.

Corollaire. Toute la logique propositionnelle peut être développée sur base du seul connecteur binaire $\uparrow$ (ou de son dual $\downarrow$).

[7] On a par exemple $(a \vee b) \;=_{def}\;$ [si a alors *true* sinon b] .

2.2 Syntaxe du calcul des propositions

2.2.1 Les règles de base

Soit $\Pi = \{p, q, r, \ldots\}$, un *lexique propositionnel*, c'est-à-dire un ensemble de symboles arbitraires appelés *propositions atomiques* ou *atomes*. La notation $p \in \Pi$ signifie que p *appartient à* Π, ou *est un élément* de Π. Signalons aussi que l'ensemble vide, celui qui ne contient aucun élément, est noté $\emptyset$.

Définition. Une *formule* du calcul des propositions est une chaîne de symboles générée par la *grammaire*

$$\begin{array}{lcl} \mathit{formula} & ::= & p\,, \text{ pour tout } p \in \Pi \\ \mathit{formula} & ::= & \mathit{true} \mid \mathit{false} \\ \mathit{formula} & ::= & \neg \mathit{formula} \\ \mathit{formula} & ::= & (\textit{formula op formula}) \\ \mathit{op} & ::= & \vee \mid \wedge \mid \Rightarrow \mid \equiv \mid \Leftarrow \end{array}$$

Chaque ligne s'interprète simplement. Par exemple, si nous savons déjà que "$\wedge$" est un opérateur (connecteur) et que "$(p \Rightarrow q)$" et "$\neg r$" sont des formules, la quatrième ligne nous permet de conclure que "$(p \Rightarrow q) \wedge \neg r$" est une formule. On appelle *dérivation* un développement détaillé montrant qu'un assemblage de symboles est une formule.[8] Voici deux exemples de dérivations, montrant que $(p \wedge q)$ et $((p \Rightarrow q) \equiv (\neg q \Rightarrow \neg p))$ sont des formules :

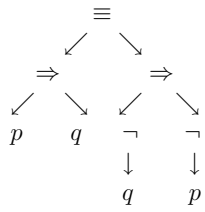
1. *formula*
2. (*formula op formula*)
3. (*formula* $\wedge$ *formula*)
4. ($p \wedge$ *formula*)
5. $(p \wedge q)$

1. $\mathit{formula}$
2. $(\mathit{formula} \equiv \mathit{formula})$
3. $((\mathit{formula} \Rightarrow \mathit{formula}) \equiv \mathit{formula})$
4. $((p \Rightarrow \mathit{formula}) \equiv \mathit{formula})$
5. $((p \Rightarrow q) \equiv \mathit{formula})$
6. $((p \Rightarrow q) \equiv (\mathit{formula} \Rightarrow \mathit{formula}))$
7. $((p \Rightarrow q) \equiv (\neg \mathit{formula} \Rightarrow \mathit{formula}))$
8. $((p \Rightarrow q) \equiv (\neg q \Rightarrow \mathit{formula}))$
9. $((p \Rightarrow q) \equiv (\neg q \Rightarrow \neg \mathit{formula}))$
10. $((p \Rightarrow q) \equiv (\neg q \Rightarrow \neg p))$

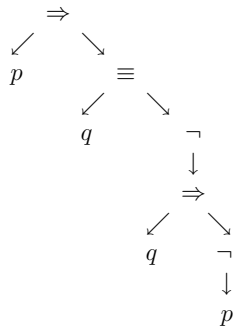
L'ordre des dérivations n'est pas total mais partiel ; il est donc naturel de représenter une dérivation par un arbre. Les arbres de dérivation de la figure 8 montrent l'importance des parenthèses. Deux formules peuvent ne différer que par les positions des parenthèses et avoir des sens très différents.[9]

[8] Formellement, cette grammaire comporte deux symboles non terminaux *formula* et *op*. Une formule est donc un mot du langage engendré par la grammaire, dépourvu de symboles non terminaux. C'est le dernier terme d'une dérivation dont le premier terme est le symbole non terminal distingué *formula*.

[9] Le même phénomène se produit en arithmétique ; les expressions arithmétiques $a * (b + c)$ et $(a * b) + c$ ont généralement des valeurs différentes.



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



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

FIG. 8 – Deux arbres de dérivation.

2.2.2 Les règles simplificatrices

Les règles simplificatrices de la logique propositionnelle sont analogues à celles de l'arithmétique. Elles sont destinées à abréger l'écriture des formules et à en faciliter la lecture. Voici d'abord deux règles d'un emploi habituel.

- Omission des parenthèses extérieures :
  on écrit $p \wedge q$ au lieu de $(p \wedge q)$, et $q \equiv \neg(q \Rightarrow \neg q)$ au lieu de $(q \equiv \neg(q \Rightarrow \neg q))$.
- Utilisation de l'associativité des opérateurs $\wedge$ et $\vee$:[10]
  on écrit $p \vee q \vee r$ au lieu de $(p \vee q) \vee r$ ou $p \vee (q \vee r)$.

Certains ouvrages utilisent en outre les règles suivantes :

- Groupement à gauche des opérateurs non associatifs :
  on écrit parfois $p \Rightarrow q \Rightarrow r$ au lieu de $(p \Rightarrow q) \Rightarrow r$.
- Priorité des opérateurs :
  en arithmétique, on écrit souvent $a + b * c$ pour $a + (b * c)$; de même on convient par exemple que $p \vee q \wedge r$ équivaut à $p \vee (q \wedge r)$ et non à $(p \vee q) \wedge r$.
  La suite $\neg$, $\wedge$, $\vee$, $\Rightarrow$, $\Leftarrow$, $\equiv$ reprend les connecteurs logiques, par ordre décroissant de priorité.

Dans la suite, seules les deux premières règles de simplification seront utilisées. On s'autorisera aussi, pour faciliter la lecture, à remplacer certaines paires de parenthèses par des paires de crochets.

2.2.3 Les notations polonaises

Les logiciens polonais ont proposé des notations permettant de se passer complètement des parenthèses. La notation polonaise directe, ou préfixée, consiste à écrire l'opérateur avant ses opérandes ; la notation polonaise inverse, ou postfixée, consiste à écrire l'opérateur après ses opérandes. La notation habituelle est dite infixée. Changer de notation revient à changer l'ordre de parcours des nœuds dans l'arbre de dérivation. A titre d'exemple, les trois parcours possibles pour les arbres de dérivation de la figure 8 sont représentés à la figure 9.

Notation infixée	$((p \Rightarrow q) \equiv (\neg q \Rightarrow \neg p))$	$(p \Rightarrow (q \equiv \neg(q \Rightarrow \neg p)))$
Notation simplifiée	$(p \Rightarrow q) \equiv (\neg q \Rightarrow \neg p)$	$p \Rightarrow (q \equiv \neg(q \Rightarrow \neg p))$
Notation préfixée	$\equiv \Rightarrow p\ q \Rightarrow \neg\ q \neg\ p$	$\Rightarrow p \equiv q \neg \Rightarrow q \neg\ p$
Notation postfixée	$p\ q \Rightarrow q \neg\ p \neg \Rightarrow \equiv$	$p\ q\ q\ p \neg \Rightarrow \neg \equiv \Rightarrow$

FIG. 9 – Les notations polonaises.

Les calculatrices de poche Hewlett-Packard proposent ou imposent l'usage de la notation postfixée, ce qui a contribué à rendre cette notation familière aux non-logiciens.[11]

2.2.4 Formules et sous-formules

La formule A est une *sous-formule* de B si l'arbre syntaxique (l'arbre de dérivation) de A est un sous-arbre de l'arbre syntaxique de B ; la formule A est une *sous-formule propre* de B si A est une sous-formule de B, mais A n'est pas identique à B.
Exemples. $p \Rightarrow q$ est une sous-formule propre de $(p \Rightarrow q) \equiv (\neg q \Rightarrow \neg p)$; en revanche, $p \Rightarrow q$ est une sous-formule impropre de $p \Rightarrow q$; enfin, $q \equiv \neg q$ n'est pas une sous-formule de $(p \Rightarrow q) \equiv (\neg q \Rightarrow \neg p)$.

On notera aussi qu'une sous-formule peut avoir plusieurs occurrences dans une formule. Par exemple, la (sous-)formule $\neg p$ a deux occurrences dans l'équivalence $\neg p \equiv \neg(p \Leftarrow \neg p)$; la (sous-)formule p a trois occurrences dans l'implication $p \Rightarrow (p \vee \neg p)$.

2.2.5 Exemples de récurrence non numérique

L'ensemble des formules du calcul des propositions est *inductif* en ce sens qu'il admet un principe de récurrence :

> *Si une propriété P est vraie de toute proposition élémentaire,*
> *et si, chaque fois qu'elle est vraie des formules A et B,*
> *elle l'est aussi des formules $\neg A$, $(A \wedge B)$, $(A \vee B)$, $(A \Rightarrow B)$ et $(A \equiv B)$,*
> *alors elle est vraie de toute formule.*

On appelle souvent *principe d'induction* tout principe de récurrence non numérique.

Ce principe peut être utilisé pour démontrer des propriétés variées. Un exemple simple consiste à démontrer que toute formule (en notation infixée, non simplifiée) comporte un nombre pair de parenthèses. D'une part, c'est vrai pour les propositions élémentaires qui comportent zéro parenthèse. D'autre part, si A comporte α parenthèses, si B comporte β parenthèses et si α et β sont des nombres pairs, alors $\neg A$ comporte α parenthèses (nombre

[10] On verra plus loin comment démontrer cette propriété.

[11] Un bon exercice de programmation consiste à écrire les procédures permettant de passer d'une notation à l'autre.

pair) et $(A \wedge B)$, $(A \vee B)$, $(A \Rightarrow B)$ et $(A \equiv B)$ comportent $\alpha + \beta + 2$ parenthèses (nombre pair), d'où la conclusion.

Un propriété plus intéressante affirme que toute formule en notation infixée peut s'écrire en notation préfixée et en notation postfixée. C'est évident pour les propositions élémentaires. D'autre part, si les versions préfixées de A et B sont α et β, alors les versions préfixées de $\neg A$, $(A \wedge B)$, $(A \vee B)$, $(A \Rightarrow B)$ et $(A \equiv B)$ sont respectivement $\neg\alpha$, $\wedge\alpha\beta$, $\vee\alpha\beta$, $\Rightarrow\alpha\beta$ et $\equiv\alpha\beta$. Le principe d'induction permet de conclure. On peut aussi démontrer que la traduction est unique.

2.3 Sémantique du calcul des propositions

2.3.1 Définitions

Une sémantique est une fonction qui donne un sens aux objets d'un langage. Une sémantique est compositionnelle si le sens d'une entité composée au moyen d'un mécanisme donné dépend uniquement du sens des composants. Dans le cadre de la logique des propositions, compositionnel signifie vérifonctionnel. On sait notamment que la formule conjonctive $(A \wedge B)$ est vraie si et seulement si les formules A et B sont toutes les deux vraies. Le caractère non compositionnel des sémantiques des langages naturels est la première cause de leur complexité (et de leur richesse).

Une sémantique compositionnelle est non seulement facile à utiliser mais aussi facile à décrire. Il suffit en fait d'enrichir les règles syntaxiques de construction des formules de règles sémantiques, permettant d'en construire le sens. Aux règles syntaxiques rappelées ci-dessous

$$\begin{array}{lll} \mathit{formula} & ::= & p\,, \text{ pour tout } p \in \Pi \\ \mathit{formula} & ::= & \mathit{true} \mid \mathit{false} \\ \mathit{formula} & ::= & \neg \mathit{formula} \\ \mathit{formula} & ::= & (\textit{formula op formula}) \\ \mathit{op} & ::= & \vee \mid \wedge \mid \Rightarrow \mid \equiv \mid \Leftarrow \end{array}$$

correspondent des règles sémantiques, permettant d'interpréter, c'est-à-dire de donner un sens à chaque formule.

Une *interprétation* propositionnelle I est basée sur une fonction $v : \Pi \rightarrow \{\mathbf{V}, \mathbf{F}\}$ qui attribue une *valeur de vérité* quelconque à chaque atome. L'interprétation I, dont le domaine est l'ensemble des formules construites au moyen du lexique Π, est l'unique fonction respectant les conditions suivantes :

$$\begin{array}{l} I(p) = v(p)\,, \text{ pour tout } p \in \Pi\,, \\ I(\mathit{true}) = \mathbf{V}\,,\ I(\mathit{false}) = \mathbf{F}\,, \\ I(\neg\varphi) = \mathbf{V} \text{ si } I(\varphi) = \mathbf{F}\,,\ I(\neg\varphi) = \mathbf{F} \text{ si } I(\varphi) = \mathbf{V}\,, \\ I(\varphi\ \mathit{op}\ \psi) = \mathcal{S}(\mathit{op}, I(\varphi), I(\psi))\,, \end{array}$$

la dernière condition est la définition de la fonction sémantique $\mathcal{S}$, résumée à la figure 10.
Remarque. Conceptuellement, la formule *true* (objet syntaxique) et la valeur de vérité $\mathbf{V}$ (objet sémantique) sont des objets très différents. L'usage d'une seule notation pour les deux objets est cependant fréquent, et peu gênant en pratique.[12]

[12] En arithmétique, il est inhabituel de distinguer l'expression arithmétique réduite au seul terme 13 (objet syntaxique) et sa valeur (objet sémantique) qui est le nombre représenté par 13.

A	$I(A_1)$	$I(A_2)$	$I(A)$
$A_1 \vee A_2$	**F**	**F**	**F**
$A_1 \vee A_2$	sinon		**V**
$A_1 \wedge A_2$	**V**	**V**	**V**
$A_1 \wedge A_2$	sinon		**F**
$A_1 \Rightarrow A_2$	**V**	**F**	**F**
$A_1 \Rightarrow A_2$	sinon		**V**
$A_1 \Leftarrow A_2$	**F**	**V**	**F**
$A_1 \Leftarrow A_2$	sinon		**V**
$A_1 \equiv A_2$	$I(A_1) = I(A_2)$		**V**
$A_1 \equiv A_2$	$I(A_1) \neq I(A_2)$		**F**

FIG. 10 – Fonction sémantique pour les opérateurs binaires.

Théorème. La fonction d'interprétation v se prolonge en une et une seule interprétation I. C'est une conséquence immédiate de l'unicité de l'arbre syntaxique (arbre de dérivation) d'une formule.

Ceci signifie seulement que, si on fixe l'interprétation des propositions élémentaires que contient une formule, on fixe ipso facto l'interprétation de la formule elle-même. La réciproque n'est généralement pas vraie. Si on impose que, par exemple, $p \wedge \neg q$ soit vrai, alors nécessairement p est vrai et q est faux. En revanche, si on impose que $p \wedge \neg q$ soit faux, on ne peut pas déduire avec certitude les valeurs de vérité attachées à p et q.

Remarques. Cet énoncé est effectivement immédiat mais on peut néanmoins le démontrer. Cette démonstration (il en existe plusieurs variantes) met en évidence les caractéristiques des démonstrations en logique : construction de l'objet qu'évoque l'énoncé, raisonnement par récurrence ou par induction. Les règles de la figure 10 permettent de calculer $I(\neg\phi)$ et $I(\phi\ \mathit{op}\ \psi)$ à partir de $I(\phi)$ et $I(\psi)$. On voit donc que si $I(\alpha)$ existe et est unique pour toute formule α comportant (strictement) moins de n connecteurs, alors $I(\alpha)$ existe et est unique pour toute formule α comportant exactement n connecteurs. D'autre part, pour une formule α comportant 0 connecteur, c'est-à-dire une proposition, on a $I(\alpha) = v(\alpha)$ par définition. Cela montre l'existence et l'unicité de l'interprétation I, prolongement sur le domaine des formules de lexique Π de la fonction d'interprétation v (de domaine Π).
On observera aussi que la figure 10 est en fait un *algorithme* (récursif), pour le calcul de $I(\alpha)$. La démonstration ci-dessus est une preuve d'exactitude pour cet algorithme. Notons enfin que, I étant univoquement déterminé à partir de v, il n'est pas indispensable d'utiliser deux notations différentes, ni de distinguer interprétation et fonction d'interprétation. Dans la suite, nous parlerons uniquement d'interprétation, et utiliserons indifféremment I et v pour noter une interprétation quelconque.

La mise en œuvre de la sémantique propositionnelle est élémentaire ; on attribue d'abord une valeur de vérité à toutes les propositions intervenant dans la formule à interpréter, c'est-à-dire à toutes les feuilles de l'arbre syntaxique correspondant à la formule. On "remonte" ensuite dans l'arbre, la valeur d'une sous-formule (correspondant à un nœud interne de l'arbre) dépendant

uniquement des valeurs attribuées à ses composants directs. L'arbre syntaxique étant fini, on termine en attribuant une valeur à la formule elle-même, correspondant à la racine de l'arbre.

Exemple. La fonction d'interprétation $v = \{(p, \mathbf{V}), (q, \mathbf{F}), (r, \mathbf{V}), (s, \mathbf{V})\}$ se prolonge en une interprétation I unique sur l'ensemble de toutes les formules basées sur le lexique $\Pi = \{p, q, r, s\}$. Le cas de $(p \vee s) \equiv (s \wedge q)$ est traité à la figure 11.

$$
\begin{aligned}
&I(p) = \mathbf{V}\,,\\
&I(s) = \mathbf{V}\,,\\
&I(p \vee s) = \mathbf{V}\,,\\
&I(q) = \mathbf{F}\,,\\
&I(s \wedge q) = \mathbf{F}\,,\\
&I((p \vee s) \equiv (s \wedge q)) = \mathbf{F}
\end{aligned}
$$

FIG. 11 – Interprétation d'une formule composée.

Remarque. Le fait que toute fonction d'interprétation v se prolonge en une et une seule interprétation I nous autorise, en pratique, à confondre les deux notions.

Remarque. L'examen exhaustif des interprétations d'une formule composée se fait souvent au moyen d'une *table de vérité* ; cette notion sera approfondie plus loin, mais nous en donnons déjà un exemple à la figure 12. Cette table montre que la formule $(p \Rightarrow q) \Rightarrow (\neg q \Rightarrow \neg p)$ est vraie pour chacune des quatre interprétations possibles.

p	q	$p \Rightarrow q$	$\neg q \Rightarrow \neg p$	$(p \Rightarrow q) \Rightarrow (\neg q \Rightarrow \neg p)$
V	**V**	**V**	**V**	**V**
V	**F**	**F**	**F**	**V**
F	**V**	**V**	**V**	**V**
F	**F**	**V**	**V**	**V**

FIG. 12 – Table de vérité de $(p \Rightarrow q) \Rightarrow (\neg q \Rightarrow \neg p)$.

2.3.2 Les connecteurs naturels

Les connecteurs vérifonctionnels utilisés en logique sont des versions simplifiées et idéalisées de leurs homologues naturels. Par exemple, le connecteur "et" de la langue française n'est pas toujours compositionnel, comme le montrent les exemples suivants :

- Il eut peur et il tua l'intrus.
- Il tua l'intrus et il eut peur.

Dans un contexte où les deux composants sont vrais, un procès d'assises par exemple, les deux phrases ont des sens nettement différents. Néanmoins, dans la plupart des cas, le connecteur "et" s'emploie de manière vérifonctionnelle, avec parfois une surcharge de sens,

indiquant une nuance temporelle ou causale. Il en va de même des versions naturelles de la négation, de la disjonction et de l'équivalence.

Le connecteur d'implication pose toutefois un problème. Des phrases telles que

- Si $2 + 2 = 4$, alors la terre tourne autour du soleil.
- Si la terre tourne autour du soleil, alors $2 + 2 = 4$.
- Si $2 + 2 = 5$, alors $2 + 2 = 6$

sont vraies selon les règles sémantiques de la logique propositionnelle,[13] elles pourraient bien être tenues pour fausses (ou "absurdes") par le non-logicien. Il se fait que, la plupart du temps, la notion d'implication n'est pas vérifonctionnelle.

Nous avons déjà noté dans le chapitre introductif que l'implication logique correspondait exactement à l'implication mathématique. Il n'empêche que, dans les théorèmes "utiles", le lien entre l'hypothèse (ou la conjonction des hypothèses) et la thèse n'est pas seulement vérifonctionnel ; il s'y ajoute un autre lien, l'existence d'une démonstration permettant de "passer" de l'hypothèse à la thèse. Il n'en est pas moins vrai qu'un énoncé tel que "si $2+2=5$ alors $2 + 2 = 6$" est un théorème de l'arithmétique, aussi valide qu'inutile.

On peut aborder le problème autrement. Il n'existe que 16 connecteurs binaires, et donc 16 possibilités de définir une approximation vérifonctionnelle de l'implication. En outre, certains choix sont d'office exclus. En particulier, on admet aisément que la valeur de vérité d'une implication $(p \Rightarrow q)$ ne peut dépendre du seul antécédent p, ou du seul conséquent q ; on admet de même que les rôles de l'antécédent et du conséquent ne sont pas interchangeables. Si l'on se réfère à la figure 5, les seuls candidats possibles sont $\circ_3$, $\circ_5$, $\circ_{12}$ et $\circ_{14}$. Si on admet en outre que, quand l'antécédent est vrai, l'implication a la valeur du conséquent, il ne reste que $\circ_5$, c'est-à-dire l'implication logique telle qu'elle a été définie plus haut.

Une comparaison plus poussée entre la logique et l'arithmétique fournira une autre "justification" de la notion d'implication (Fig. 13).

2.3.3 Formalisation d'un texte en langage naturel

Comment formaliser le raisonnement ci-dessous en logique propositionnelle ?

> *Si le récit biblique de la création est strictement exact, le soleil n'a pas été créé avant le quatrième jour. Et si le soleil n'a pas été créé avant le quatrième jour, il ne peut avoir été la cause de l'alternance de lumière et d'obscurité pendant les trois premiers jours. Mais soit le sens du mot "jour" dans la bible est différent du sens habituel, soit le soleil a bien été la cause de l'alternance de lumière et d'obscurité pendant les trois premiers jours. Il s'en suit DONC que le récit biblique de la création n'est pas strictement exact ou que le sens du mot "jour" dans la bible est différent du sens habituel.*

Introduisons les propositions suivantes :

- E : le récit biblique de la création est strictement Exact ;
- Q : le soleil a été créé avant le Quatrième jour ;

[13] On admet que les propositions élémentaires contenues dans ces phrases ont les valeurs de vérité que leur assignent les mathématiques et l'astronomie …

– A : le soleil a été la cause de l'Alternance de lumière et d'obscurité pendant les trois premiers jours ;
– D : le sens du mot "jour" dans la bible est Différent du sens habituel.

Le raisonnement se formalise en

$$\frac{E \Rightarrow \neg Q\,,\ \neg Q \Rightarrow \neg A\,,\ D \vee A}{\neg E \vee D}$$

On peut vérifier que toute interprétation rendant vraies les trois prémisses rend vraie la conclusion. En effet, si la conclusion est fausse pour l'interprétation v, on a nécessairement $v(E) = \mathbf{V}$ et $v(D) = \mathbf{F}$. Les valeurs de vérité des trois prémisses sont alors

1. $v(E \Rightarrow \neg Q) = v(\neg Q)$;
2. $v(\neg Q \Rightarrow \neg A)$;
3. $v(D \vee A) = v(A)$.

On vérifie immédiatement qu'aucune des quatre manières d'attribuer des valeurs de vérité à Q et A ne permet de rendre simultanément vraies les formules $\neg Q$, $(\neg Q \Rightarrow \neg A)$ et A. En anticipant sur la section suivante, on peut donc affirmer que le raisonnement est correct.

Remarque. Notre analyse ne permet évidemment pas de porter un jugement sur la véracité des trois prémisses. Une simple lecture de la Genèse permet de vérifier la véracité de la première. Pour admettre la seconde, il faut admettre notamment que la cause doit précéder (temporellement) l'effet. L'analyse de chacune des prémisses et de la conclusion requiert naturellement une bonne connaissance du français. En fait, il est difficile d'inventorier avec précision les connaissances, élémentaires mais nombreuses, nécessaires à la validation de ce raisonnement.

2.3.4 Logique et arithmétique

Si on assimile les valeurs de vérité **F** et **V** aux nombres 0 et 1, respectivement, les connecteurs logiques deviennent les restrictions au domaine $\{0,1\}$ d'opérations arithmétiques. Naturellement, les opérations arithmétiques dont les connecteurs logiques sont des restrictions ne sont pas univoquement déterminées. Par exemple, la fonction identité sur $\{0,1\}$ est la restriction non seulement de la fonction identité sur $\mathbb{N}$ mais aussi, entre autres, de la fonction carré puisque $0^2 = 0$ et $1^2 = 1$. Il est intéressant de considérer que les connecteurs sont les restrictions d'opérations arithmétiques aussi élémentaires et utiles que possible. Nous proposons les rapprochements repris à la figure 13.
Remarque. L'expression $a \bmod n$, dans laquelle a est un entier et n un entier strictement positif, désigne le reste de la division de a par n, c'est-à-dire l'unique entier $r \in \{0, \ldots, n-1\}$ tel que l'équation $a = nq + r$ admette une solution (nécessairement unique et appelée le quotient).

Le rôle de l'implication en logique est analogue à celui, crucial, de la relation d'ordre numérique en arithmétique. Comme il n'existe que deux valeurs de vérité, l'implication jouit de propriétés particulières. Par exemple, si on considère une formule $A(p)$ comportant une seule occurrence d'une proposition p comme une fonction de p, cette fonction est croissante, si $A(false) \Rightarrow A(true)$ est valide (voir paragraphe suivant), ou décroissante (si $A(true) \Rightarrow A(false)$ est valide) ou les deux à la fois (si $A(false) \equiv A(true)$ est valide). Cette propriété, qui n'est pas vérifiée en arithmétique, permettra de faciliter l'analyse de certaines formules.

$a \equiv b$	$a = b$
$a \Rightarrow b$	$a \leq b$
$a \wedge b$	$\min(a, b)$ ou encore $a * b$
$a \vee b$	$\max(a, b)$
$a \oplus b$	$a \neq b$ ou encore $(a + b) \bmod 2$
$\neg a$	$1 - a$

FIG. 13 – Interprétation arithmétique des connecteurs.

2.4 Relation de conséquence logique

Un raisonnement est un mécanisme, ou un algorithme, permettant de dériver une proposition, la *conclusion*, à partir d'un ensemble de propositions données, les *prémisses*. Un raisonnement est *correct*, ou *valide*, si dans tout contexte où les prémisses sont vraies, la conclusion est vraie aussi. On dit alors que la conclusion est *conséquence logique* de l'ensemble des prémisses. Dans cette section, nous abordons le problème central de la logique, qui est de déterminer si une formule est ou n'est pas conséquence logique d'un ensemble donné de formules.

Un problème plus simple est de déterminer si une formule donnée (ou un ensemble de formules) est vrai pour toutes les interprétations possibles, pour au moins une, ou pour aucune. C'est ce problème que nous allons aborder en premier lieu.

2.4.1 Consistance et validité

Consistance et validité d'une formule isolée. Soit A une formule propositionnelle.

– Une interprétation v de A est un *modèle* de A si $v(A) = \mathbf{V}$.
– A est *satisfaisable* ou *consistante* si A a au moins un modèle.
– A est *valide*, ou A est une *tautologie*, si $v(A) = \mathbf{V}$ pour toute interprétation v. La notation $\models A$ exprime que A est valide.
– A est *insatisfaisable* ou *inconsistante* si A n'est pas satisfaisable, c'est-à-dire si, pour toute interprétation v, on a $v(A) = \mathbf{F}$.

Remarque. "(In)satisfaisabilité" est le terme propre ; "(in)consistance" est souvent préféré par euphonie.

Théorème. Une formule A est valide si et seulement si sa négation $\neg A$ est insatisfaisable.
Démonstration. Les quatre énoncés suivants sont équivalents.

– A est valide ;
– $v(A) = \mathbf{V}$, pour toute interprétation v ;
– $v(\neg A) = \mathbf{F}$, pour toute interprétation v ;
– $\neg A$ est insatisfaisable.

Consistance et validité d'un ensemble de formules. Soit E un ensemble de formules.

– Le *lexique* Π de E est la réunion des lexiques des éléments de E.

– Une *interprétation* de E est une fonction v de Π dans $\{\mathbf{V}, \mathbf{F}\}$; elle admet un prolongement unique permettant d'interpréter toutes les formules dont le lexique est inclus dans Π et, en particulier, toutes les formules de E.
– Une interprétation v de E est un *modèle* de E si elle est un modèle de tous les éléments de E, c'est-à-dire si $v(A) = \mathbf{V}$ pour toute formule $A \in E$.
– E est *satisfaisable* ou *consistant* si E a au moins un modèle.
– E est *insatisfaisable* ou *inconsistant* si E n'est pas satisfaisable, c'est-à-dire si, pour toute interprétation v, on a $v(A) = \mathbf{F}$, pour au moins une formule $A \in E$.

Voici trois conséquences immédiates de ces définitions.

– Toute interprétation est un modèle de l'ensemble vide $\emptyset$.
– Les modèles du singleton $\{A\}$ sont les modèles de la formule A.
– Les modèles de l'ensemble fini $\{A_1, \ldots, A_n\}$ sont les modèles de la conjonction $A_1 \wedge \cdots \wedge A_n$.

Remarques. On peut définir la validité d'un ensemble E comme le fait que toute interprétation est un modèle de E. Cela n'est guère intéressant car un ensemble valide n'est qu'un ensemble de formules valides. La situation est différente en ce qui concerne la consistance. Il est clair que les éléments d'un ensemble consistant sont des formules consistantes, mais un ensemble de formules consistantes peut être inconsistant ; c'est par exemple le cas de la paire $\{p, \neg p\}$.
Une interprétation v d'un ensemble *fini* $E = \{A_1, \ldots, A_n\}$ est un modèle de E si et seulement si c'est un modèle de la formule $A_1 \wedge \cdots \wedge A_n$; c'est pourquoi on parle parfois d'ensemble "conjonctif" de formules. Notons enfin que les notions d'interprétation et de consistance restent pertinentes dans le cas d'un ensemble infini de formules. En revanche, la notion de "conjonction infinie" ou de "formule infinie" n'existe pas dans notre contexte, parce qu'elle correspondrait à un arbre sémantique infini, objet (informatique) difficilement manipulable.

Théorème. Si $E' \subset E$, tout modèle de E est un modèle de E'.
Corollaire. Tout sous-ensemble d'un ensemble consistant est consistant.
Corollaire. Tout sur-ensemble d'un ensemble inconsistant est inconsistant.
Remarque. La notation $E' \subset E$ représente l'énoncé "tout élément de E' est un élément de E".

2.4.2 Conséquence logique, équivalence logique

Une formule A est *conséquence logique* d'un ensemble E de formules si tout modèle de E est un modèle de A. La notation $E \models A$ exprime que A est conséquence logique de E.

Remarque. Si E est valide, et en particulier si E est vide, A est conséquence logique de E si et seulement si A est valide. On peut donc voir la notation

$$\models A$$

comme une abréviation de la notation

$$\emptyset \models A\,.$$

Autrement dit, une formule est valide si et seulement si elle est conséquence logique de l'ensemble vide. On notera aussi qu'une formule est valide si et seulement si elle est

conséquence logique de tout ensemble. Dans le même ordre d'idée, la notation

$$E \models \mathit{false}$$

exprime que l'ensemble E est inconsistant. En effet, le seul moyen que tout modèle de E soit un modèle de false est que l'ensemble E n'admette aucun modèle.

Nous introduisons maintenant un résultat, immédiat mais important, permettant de ramener la question "A est-elle conséquence logique de E ?" à la question "$E \cup \{\neg A\}$ est-il inconsistant ?".
Théorème de la déduction (cas fini). Soit A une formule et soit $U = \{A_1, \ldots, A_n\}$ un ensemble fini de formules. Les trois conditions suivantes sont équivalentes :

– A est une conséquence logique de U ; $U \models A$;
– l'ensemble $U \cup \{\neg A\}$ est inconsistant ; $U \cup \{\neg A\} \models \mathit{false}$;
– l'implication $(A_1 \wedge \ldots \wedge A_n) \Rightarrow A$ est valide ; $\models (A_1 \wedge \ldots \wedge A_n) \Rightarrow A$.

Théorème de la déduction (cas général). Soit A une formule et soit E un ensemble de formules. Les deux conditions suivantes sont équivalentes :

– A est une conséquence logique de E ; $E \models A$;
– l'ensemble $E \cup \{\neg A\}$ est inconsistant ; $E \cup \{\neg A\} \models \mathit{false}$.

La *théorie* d'un ensemble E de formules est l'ensemble des conséquences logiques de E, soit $\mathcal{T}(E) = \{A : E \models A\}$; les éléments de E sont les *axiomes* ou les *postulats* et les éléments de $\mathcal{T}(E)$ sont les *théorèmes*. Cette notion est surtout employée dans le cadre de la logique des prédicats.

Théorème. Soit E un ensemble de formules et soit U un ensemble de conséquences logiques de E. Les ensembles E et $E \cup U$ admettent exactement les mêmes modèles.
Corollaire. On préserve la consistance d'un ensemble de formules par suppression de formules (quelconques) et aussi par adjonction de conséquences logiques ; on préserve l'inconsistance d'un ensemble par adjonction de formules (quelconques) et aussi par suppression de conséquences logiques (de ce qui n'est pas supprimé !).

Deux formules propositionnelles A_1 et A_2 sont dites *logiquement équivalentes* (ce qui se note $A_1 \leftrightarrow A_2$, ou parfois $A_1 \simeq A_2$) si elles ont les mêmes modèles, c'est-à-dire si $v(A_1) = v(A_2)$ pour toute interprétation v.
En pratique, on peut vérifier simplement l'équivalence logique de deux formules, en passant en revue toutes les interprétations définies sur la réunion des lexiques des deux formules. On établit par exemple

$$p \vee q \;\leftrightarrow\; q \vee p$$

en dressant le tableau suivant :

p	q	$v(p \vee q)$	$v(q \vee p)$
V	**V**	**V**	**V**
V	**F**	**V**	**V**
F	**V**	**V**	**V**
F	**F**	**F**	**F**

Remarque. Le mot “équivalence” est employé dans trois cas bien distincts, que nous allons énumérer.

1. Ce mot désigne des objets du *langage formel* qu’est la logique propositionnelle. On peut écrire
   – Le symbole $\equiv$ représente le connecteur d’équivalence.
   – La formule $(p \vee q) \equiv (q \vee p)$ est une équivalence valide ;
   – La formule $p \equiv q$ est une équivalence contingente ;
   – La formule $p \equiv \neg p$ est une équivalence inconsistante.
2. Ce mot intervient aussi dans le *métalangage*, c’est-à-dire le formalisme, comportant des notations spécifiques, qui permet de parler des objets logiques. On peut écrire
   – Le symbole $\leftrightarrow$ représente la relation d’équivalence logique.
   – L’expression $(p \vee q) \leftrightarrow (q \vee p)$ n’est pas une formule, mais un énoncé du métalangage ; cet énoncé est vrai et exprime que les formules $(p \vee q)$ et $(q \vee p)$ sont logiquement équivalentes.
   – L’énoncé $p \leftrightarrow q$ appartient au métalangage ; il est faux parce que les formules p et q ne sont pas logiquement équivalentes.
3. Enfin, le mot “équivalence” est employé en français, la langue qui nous permet d’écrire ce texte, et d’évoquer les objets du langage et du métalangage. On peut écrire
   – Les expressions $\models (A \equiv B)$ et $A \leftrightarrow B$ appartiennent toutes les deux au métalangage ; ce sont des énoncés interchangeables, ou *équivalents*, parce qu’ils sont tous les deux vrais, ou tous les deux faux, selon les formules que les variables (du métalangage) A et B représentent.
   – Les énoncés “tout sous-ensemble d’un ensemble consistant est consistant” et “tout sur-ensemble d’un ensemble inconsistant est inconsistant” appartiennent à la langue française (et non au métalangage) ; ils sont *équivalents*, parce qu’ils sont interchangeables ; un raisonnement (informel et élémentaire) permet de déduire un énoncé de l’autre.
   – La phrase française “Les énoncés $\models (A \equiv B)$ et $A \leftrightarrow B$ sont équivalents” n’appartient pas au métalangage, mais exprime un fait (vrai) relatif à deux énoncés du métalangage.

L’usage d’un même mot pour désigner des concepts différents se justifie (ou au moins s’explique) par les liens étroits existant entre ces concepts. Ces liens apparaissent par exemple dans le théorème suivant, qui exprime l’équivalence (au sens 3) entre deux énoncés du métalangage.

Pour toutes formules A_1 et A_2, on a $A_1 \leftrightarrow A_2$ si et seulement si on a $\models (A_1 \equiv A_2)$.
Ce théorème, dont la démonstration élémentaire est laissée au lecteur, exprime que deux formules sont logiquement équivalentes (sens 2) si et seulement si l’équivalence (sens 1) dont elles sont les deux termes est valide.[14]

Selon les formules représentées par A_1 et A_2, les quatre énoncés

[14] Pour “chicaner” un peu plus, signalons que la locution “si et seulement si” est utilisée, en français, pour exprimer l’équivalence (au sens 3) entre deux énoncés du métalangage … et parfois aussi entre deux énoncés du français, c’est-à-dire entre deux phrases énonciatives quelconques. (Rassurons le lecteur qui nous a suivi jusqu’ici : c’est fini sur ce point !)

– $A_1 \leftrightarrow A_2$.
– $\models (A_1 \equiv A_2)$.
– $\models (A_1 \Rightarrow A_2)$ et $\models (A_2 \Rightarrow A_1)$.
– $\{A_1\} \models A_2$ et $\{A_2\} \models A_1$.

sont, ou bien tous vrais, ou bien tous faux.
Remarques. On écrit souvent $A \models B$ au lieu de $\{A\} \models B$, et $E, A, B \models C$ au lieu de $E \cup \{A, B\} \models C$. De plus, certains auteurs écrivent $v \models A$ au lieu de $v(A) = \mathbf{V}$. Cela vient de ce que l’on assimile parfois l’interprétation v à l’ensemble des formules dont v est un modèle. Mieux vaut éviter cette surcharge de sens pour une notation importante.

2.4.3 Echange et substitution uniforme

En arithmétique et en algèbre, on est souvent amené à remplacer une variable ou une sous-expression d’une expression ou d’une équation donnée par une autre expression. Ce remplacement est intéressant s’il respecte certaines propriétés de l’expression ou de l’équation initiale. Deux cas particuliers sont d’usage fréquent. Tout d’abord, si une expression α contient une sous-expression β égale à une troisième expression γ, on peut remplacer dans α une ou plusieurs occurrences de β par γ sans changer la valeur de α. Supposons par exemple

$$\alpha =_{def} 2\beta x + 3(\beta+\delta)^2(y-1)^\beta \text{ et } \beta = \gamma\,.$$

On a, entre autres, les égalités suivantes :

$$\alpha = 2\beta x + 3(\gamma+\delta)^2(y-1)^\beta = 2\beta x + 3(\beta+\delta)^2(y-1)^\gamma = 2\beta x + 3(\gamma+\delta)^2(y-1)^\gamma\,.$$

Un autre type de remplacement est souvent employé dans les équations. De l’égalité bien connue

$$(x+y)^2 = x^2 + 2xy + y^2\,,$$

on tire par exemple

$$(ab+3c)^2 = (ab)^2 + 2ab3c + (3c)^2\,.$$

Notons deux différences importantes entre les deux types de remplacements :
– Dans le premier cas on exige l’égalité du terme remplaçant et du terme remplacé, mais pas dans le second.
– Dans le second cas on exige le remplacement uniforme, de toutes les occurrences du terme remplacé, mais pas dans le premier.

Ces deux résultats paraissent évidents mais on doit se méfier, pour au moins trois raisons. La première est que les mathématiques fourmillent de “résultats évidents” … mais faux. La propriété d’associativité de l’addition, souvent résumée par la formule

$$a + (b + c) \;=\; (a + b) + c$$

permet, dans un enchaînement d’additions, de former arbitrairement des résultats intermédiaires, et de calculer indiffemment $(a + (b + c)) + d$ ou $(a + b) + (c + d)$, les résultats

étant égaux. Cette propriété est valable pour toute somme finie, mais pas pour toute somme infinie (série), comme le montre l'exemple suivant :[15]

$$1 = 1 + [(-1)+1] + \cdots + [(-1)+1] + \cdots \stackrel{?}{=} [1+(-1)] + \cdots + [1+(-1)] + \cdots = 0\,.$$

La deuxième raison est que les résultats susmentionnés, si évidents qu'ils paraissent, deviennent faux dans certains contextes particuliers. Par exemple, on apprend en astronomie que "l'étoile du berger" est en fait une planète, Vénus ; on apprend aussi que les planètes, au contraire des étoiles dites "fixes", tournent autour du soleil. Un imprudent remplacement conduirait à confondre les phrases

Jacques sait que Vénus tourne autour du Soleil.

et

Jacques sait que l'étoile du berger tourne autour du Soleil.

alors que pour beaucoup de gens la première phase est vraie mais pas la seconde. Voici un autre exemple :

L'expression $(x+y)^3$ s'écrit en moins de 10 caractères,
donc l'expression $x^3+3x^2y+3xy^2+y^3$ s'écrit en moins de 10 caractères.

La troisième raison pour laquelle il n'est pas inutile de démontrer des "résultats évidents" est que les preuves sont parfois aussi instructives que les théorèmes correspondants. En particulier, la validité du premier principe de remplacement évoqué plus haut tient à une propriété essentielle des opérateurs mathématico-logiques ; laquelle ?

Lemme de remplacement. La propriété cruciale des opérateurs mathématico-logiques est que la valeur de la forme $f(e_1,\ldots,e_n)$ ne dépend que de l'opérateur f et de la *valeur* des opérandes $e_1,\ldots,e_n$; en particulier, si $e_i = e'_i$, alors $f(e_1,\ldots,e_n) = f(e'_1,\ldots,e'_n)$. On dit que les opérateurs mathématico-logiques sont compositionnels, ou encore dénotationnels ; en logique propositionnelle, on a déjà noté que "compositionnel" signifie "vérifonctionnel". L'opérateur "Je sais" n'est pas vérifonctionnel ; il n'est donc pas étonnant qu'il mette en défaut les principes évoqués plus haut.

Formellement, le lien entre vérifonctionnalité et remplacement s'exprime comme suit.

Lemme (remplacement). Soient A, B, C des formules et v une interprétation. On suppose que A apparaît au moins une fois comme sous-formule de C, et de plus que $v(A) = v(B)$. Si D est le résultat du remplacement d'une occurrence de A par B dans C, alors $v(D) = v(C)$.

Première démonstration du lemme. On considère l'arbre syntaxique relatif à la formule C. Chaque nœud n de l'arbre correspond à une sous-formule φ_n de C ; il peut être étiqueté par la valeur de vérité $v(\varphi_n)$. Le fait que les connecteurs soient vérifonctionnels signifie que la valeur attachée à un nœud intérieur n ne dépend que du connecteur principal de φ_n et des valeurs associées aux descendants directs de n. Le remplacement d'une occurrence de A par B correspond au remplacement d'un sous-arbre par un autre mais, comme $v(A) = v(B)$, ceci ne change ni l'étiquette de la racine du sous-arbre, ni les étiquettes des ancêtres, dont la racine de l'arbre. Cela implique $v(C) = v(D)$.

[15] Ce fait nous paraît maintenant évident, mais a été dans le passé à l'origine d'erreurs graves.

Remarque. Cette démonstration se résume très bien en un dessin, que nous suggérons au lecteur de tracer. La démonstration qui suit, plus dans le style des mathématiciens, n'implique pas la représentation, même mentale, d'un objet graphique ; on utilise au lieu de cela la notion de profondeur d'une sous-formule dans une formule … ce qui, en fait, revient au même.

Seconde démonstration du lemme. On raisonne par *induction* sur la profondeur d de la sous-formule A dans C, qui correspond à la profondeur de la racine du sous-arbre syntaxique A dans l'arbre C.[16] Soit v une interprétation telle que $v(A) = v(B)$.

- $d = 0$: $A = C$ et $B = D$, donc $v(C) = v(D)$.
- $d > 0$: C est de la forme $\neg C'$ ou $(C'\ \mathit{op}\ C'')$. Dans le premier cas, A est de profondeur $d-1$ dans C' et, en nommant D' le résultat du remplacement de A par B dans C', on a (hypothèse inductive) $v(C') = v(D')$; comme $C = \neg C'$ et $D = \neg D'$, on a aussi $v(C) = v(D)$. Dans le second cas, si l'occurrence à remplacer se trouve dans C', on a aussi $v(C') = v(D')$, d'où $v(C) = v(C'\ \mathit{op}\ C'') = v(D'\ \mathit{op}\ C'') = v(D)$.[17]

Théorème de l'échange. Le théorème suivant est une conséquence immédiate du lemme de remplacement.

Théorème de l'échange. Soient A et B deux formules ; soient C une formule admettant A comme sous-formule, et D la formule obtenue en remplaçant une ou plusieurs occurrence(s) de A par B dans C. On a

$$(A \equiv B) \models (C \equiv D)\,.$$

Démonstration. Il suffit de montrer que, pour toute interprétation v, si $v(A) = v(B)$, alors $v(C) = v(D)$. Dans le cas particulier où D s'obtient par remplacement d'une seule occurrence de A, on applique le lemme de remplacement. Dans le cas général où n occurrences sont remplacées, il suffit d'appliquer n fois le lemme de remplacement.

Démonstration directe. Considérons une table de vérité pour la formule C. Elle comporte une colonne pour C elle-même, et une colonne pour chacune des sous-formules de C et en particulier une colonne relative à A. Pour obtenir une table de vérité pour D il suffit de

1. Juxtaposer à la colonne relative à A une table relative à B (l'ordre des lignes étant le même).
2. Remplacer les têtes de colonne comportant les occurrences remplacées par la formule résultant du remplacement ; le reste de la colonne n'est pas altéré.
3. Supprimer les colonnes inutiles.

Corollaire. Avec les notations du théorème de l'échange, si A et B sont logiquement équivalents, alors C et D sont logiquement équivalents.[18]

[16] La notion de profondeur d'une sous-formule peut se définir par induction, sans évoquer la notion d'arbre syntaxique : X est de profondeur 0 dans X ; si X est de profondeur d dans Y, alors X est de profondeur $d+1$ dans $\neg Y$, dans $Y \Rightarrow Z$, etc. On notera aussi qu'une formule peut contenir plusieurs occurrences d'une sous-formule (cf. § 2.2.4) ; ces occurrences peuvent avoir des profondeurs distinctes.

[17] Ce dernier point utilise explicitement le caractère vérifonctionnel de *op*.

[18] Cela s'écrit, si $\models (A \equiv B)$, alors $\models (C \equiv D)$.

Exemple. Soient les formules

$$A : \ p \Rightarrow q$$
$$B : \ \neg p \vee q$$
$$C : ((p \Rightarrow q) \wedge r) \vee ((p \Rightarrow q) \Rightarrow r)$$
$$D : ((\neg p \vee q) \wedge r) \vee ((p \Rightarrow q) \Rightarrow r)$$

Une table de vérité relative à C est

p	q	r	$p \Rightarrow q$	$(p \Rightarrow q) \wedge r$	$(p \Rightarrow q) \Rightarrow r$	C
V	**V**	**V**	**V**	**V**	**V**	**V**
V	**V**	**F**	**V**	**F**	**F**	**F**
V	**F**	**V**	**F**	**F**	**V**	**V**
V	**F**	**F**	**F**	**F**	**V**	**V**
F	**V**	**V**	**V**	**V**	**V**	**V**
F	**V**	**F**	**V**	**F**	**F**	**F**
F	**F**	**V**	**V**	**V**	**V**	**V**
F	**F**	**F**	**V**	**F**	**F**	**F**

On introduit les colonnes supplémentaires nécessaires (ici, une seule) :

p	q	r	$p \Rightarrow q$	$\neg p \vee q$	$(p \Rightarrow q) \wedge r$	$(p \Rightarrow q) \Rightarrow r$	C
V	**V**	**V**	**V**	**V**	**V**	**V**	**V**
V	**V**	**F**	**V**	**V**	**F**	**F**	**F**
V	**F**	**V**	**F**	**F**	**F**	**V**	**V**
V	**F**	**F**	**F**	**F**	**F**	**V**	**V**
F	**V**	**V**	**V**	**V**	**V**	**V**	**V**
F	**V**	**F**	**V**	**V**	**F**	**F**	**F**
F	**F**	**V**	**V**	**V**	**V**	**V**	**V**
F	**F**	**F**	**V**	**V**	**F**	**F**	**F**

On change les têtes de colonnes concernées par le remplacement (ici, deux), sans changer les colonnes elles-mêmes :

p	q	r	$p \Rightarrow q$	$\neg p \vee q$	$(\neg p \vee q) \wedge r$	$(p \Rightarrow q) \Rightarrow r$	D
V	**V**	**V**	**V**	**V**	**V**	**V**	**V**
V	**V**	**F**	**V**	**V**	**F**	**F**	**F**
V	**F**	**V**	**F**	**F**	**F**	**V**	**V**
V	**F**	**F**	**F**	**F**	**F**	**V**	**V**
F	**V**	**V**	**V**	**V**	**V**	**V**	**V**
F	**V**	**F**	**V**	**V**	**F**	**F**	**F**
F	**F**	**V**	**V**	**V**	**V**	**V**	**V**
F	**F**	**F**	**V**	**V**	**F**	**F**	**F**

Enfin, on supprime les colonnes devenues inutiles (ici, aucune) ; le résultat est une table de vérité pour D.

Corollaire. Avec les notations du théorème de l'échange, si A et B sont logiquement équivalents, alors C et D sont logiquement équivalents.[19]

Exemple. On donne $A =_{def} p$, $B =_{def} \neg\neg p$ et $C =_{def} (p \Rightarrow q) \equiv (\neg q \Rightarrow \neg p)$. Trois choix sont possibles pour D :

- $D =_{def} (\neg\neg p \Rightarrow q) \equiv (\neg q \Rightarrow \neg p)$;
- $D =_{def} (p \Rightarrow q) \equiv (\neg q \Rightarrow \neg\neg\neg p)$;
- $D =_{def} (\neg\neg p \Rightarrow q) \equiv (\neg q \Rightarrow \neg\neg\neg p)$.

Comme A et B sont logiquement équivalents, C et D le sont aussi.

Applications du théorème de l'échange. La figure 14 donne une série d'équivalences logiques qui, via le théorème de l'échange, permettent de simplifier des formules. Ces équivalences logiques sont des lois algébriques, analogues aux identités remarquables utilisées en arithmétique et en algèbre.

$$(X \wedge X) \leftrightarrow X \leftrightarrow (X \vee X)$$
$$(X \wedge Y) \leftrightarrow (Y \wedge X)$$
$$(X \vee Y) \leftrightarrow (Y \vee X)$$
$$((X \wedge Y) \wedge Z) \leftrightarrow (X \wedge (Y \wedge Z))$$
$$((X \vee Y) \vee Z) \leftrightarrow (X \vee (Y \vee Z))$$
$$(X \Rightarrow X) \leftrightarrow \mathit{true}$$
$$((X \Rightarrow Y) \wedge (Y \Rightarrow X)) \leftrightarrow (X \equiv Y)$$
$$(((X \Rightarrow Y) \wedge (Y \Rightarrow Z)) \Rightarrow (X \Rightarrow Z)) \leftrightarrow \mathit{true}$$
$$(X \Rightarrow Y) \leftrightarrow ((X \wedge Y) \equiv X)$$
$$(X \Rightarrow Y) \leftrightarrow ((X \vee Y) \equiv Y)$$
$$(X \wedge (Y \vee Z)) \leftrightarrow ((X \wedge Y) \vee (X \wedge Z))$$
$$(X \vee (Y \wedge Z)) \leftrightarrow ((X \vee Y) \wedge (X \vee Z))$$
$$(X \Rightarrow (Y \Rightarrow Z)) \leftrightarrow ((X \Rightarrow Y) \Rightarrow (X \Rightarrow Z))$$
$$(X \vee \neg X) \leftrightarrow \mathit{true} \ ; \ (X \wedge \neg X) \leftrightarrow \mathit{false}$$
$$(X \vee \mathit{true}) \leftrightarrow \mathit{true} \ ; \ (X \wedge \mathit{true}) \leftrightarrow X$$
$$(X \vee \mathit{false}) \leftrightarrow X \ ; \ (X \wedge \mathit{false}) \leftrightarrow \mathit{false}$$
$$\neg\neg X \leftrightarrow X$$
$$\neg(X \wedge Y) \leftrightarrow (\neg X \vee \neg Y)$$
$$\neg(X \vee Y) \leftrightarrow (\neg X \wedge \neg Y)$$

FIG. 14 – Quelques équivalences logiques.

- On utilise souvent ces lois algébriques, combinées au théorème de l'échange, pour simplifier les formules. Voici un exemple :
  $p \wedge (\neg p \vee q) \leftrightarrow (p \wedge \neg p) \vee (p \wedge q) \leftrightarrow \mathit{false} \vee (p \wedge q) \leftrightarrow p \wedge q$

[19] Cela s'écrit, si $\models (A \equiv B)$, alors $\models (C \equiv D)$.

– Ces équivalences décrivent des propriétés des connecteurs, telles l'associativité, la commutativité et l'idempotence de $\wedge$, $\vee$, la transitivité de l'implication et de l'équivalence, etc.
– D'un point de vue sémantique, des formules telles que $p \Rightarrow q$ et $\neg p \vee q$ ne doivent pas être distinguées. L'ensemble des formules construites sur un lexique donné et dans lequel des formules logiquement équivalentes ne "comptent" que pour une seule formule est intéressant à étudier. Cela suggère (aux mathématiciens ...) l'étude de l'ensemble-quotient $\Phi_\Pi/\leftrightarrow$, où Φ_Π désigne l'ensemble des formules basées sur le lexique Π. Cet ensemble-quotient est une *algèbre de Boole* particulière (appelée *algèbre de Lindenbaum*), dont les opérations sont naturellement les connecteurs. Si $\Pi = \{p\}$, l'algèbre correspondante comporte quatre éléments ; on a $A_1 = \{false, p, \neg p, true\}$. Si $\Pi = \{p, q\}$, l'algèbre correspondante comporte seize éléments ; on a A_2 = { $false$, $p \wedge q$, $p \wedge \neg q$, $\neg p \wedge q$, $\neg p \wedge \neg q$, p, $\neg p$, q, $\neg q$, $p \oplus q$, $p \equiv q$, $p \vee q$, $p \vee \neg q$, $\neg p \vee q$, $\neg p \vee \neg q$, $true$ }. L'algèbre A_n est isomorphe à $\mathcal{P}(E_n)$, où E_n est un ensemble à 2^n éléments. La négation, la conjonction, la disjonction, l'implication et l'équivalence correspondent respectivement à la complémentation, l'intersection, la réunion, l'inclusion et l'égalité.

Théorème de la substitution uniforme. Soient A_1 et A_2 des formules et B la formule $A_1 \Rightarrow (A_1 \vee A_2)$. La formule B peut être très longue, mais on "voit" immédiatement qu'elle est valide, comme "instance" de la formule valide $p \Rightarrow (p \vee q)$. Si C est une formule, on note $C[p, q \,/\, A_1, A_2]$ la formule obtenue en remplaçant *toutes* les occurrences des propositions p et q dans C par les formules A_1 et A_2, respectivement. On note aussi $[p, q \,/\, A_1, A_2]$ la fonction (dite "substitution uniforme double") qui à toute formule C associe la formule $C[p, q \,/\, A_1, A_2]$.

Remarque. Les formules $C[p, q \,/\, A_1, A_2]$, $C[p/A_1][q/A_2]$ et $C[q/A_2][p/A_1]$ peuvent être distinctes ; c'est le cas par exemple si C est $p \vee q$ et si A_1 et A_2 sont q et p, respectivement. Dans le cas particulier où aucune proposition p_i n'intervient dans les éléments de l'ensemble $\{A_1, \ldots, A_n\}$, la substitution uniforme n-uple est dite *indépendante* ; on note alors que les formules $C[p_1, \ldots, p_n \,/\, A_1, \ldots, A_n]$ et $C[p_1/A_1] \ldots [p_n/A_n]$ sont nécessairement identiques. Ceci montre que toute substitution uniforme n-uple indépendante est la composée (dans n'importe quel ordre) des n substitutions simples correspondantes. Ce résultat intéresse le programmeur, qui peut remplacer l'affectation n-uple

$$(x_1, \ldots, x_n) := (e_1, \ldots, e_n)$$

par la séquence

$$x_1 := e_1; \ldots; x_n := e_n$$

à condition que les expressions affectantes ne contiennent pas les variables affectées.

Remarque. Toute substitution uniforme n-uple est la composition de $2n$ substitutions uniformes simples indépendantes.[20]

Lemme de substitution uniforme. Soient $C, A_1, \ldots, A_n$ des formules et $p_1, \ldots, p_n$ des propositions deux à deux distinctes. Si v est une interprétation, définie sur un lexique comportant toute proposition intervenant dans C ou dans l'un des A_i, et telle que $v(p_i) = v(A_i)$ ($i = 1, \ldots, n$), alors on a $v(C[p_1, \ldots, p_n \,/\, A_1, \ldots, A_n]) = v(C)$.

Exemple de substitution uniforme. Soit

$$C =_{def} p_1 \vee (q \Rightarrow p_2)\,,\ A_1 =_{def} p_2 \wedge (p_1 \vee r)\,,\ A_2 =_{def} p_1 \vee q\,.$$

On a alors

$$C[p_1, p_2 \,/\, A_1, A_2] =_{def} (p_2 \wedge (p_1 \vee r)) \vee (q \Rightarrow (p_1 \vee q))\,.$$

Si on choisit $v =_{def} \{(p_1, \mathbf{F}), (p_2, \mathbf{V}), (q, \mathbf{V}), (r, \mathbf{F})\}$, on a $v(A_1) = v(p_1) = \mathbf{F}$ et $v(A_2) = v(p_2) = \mathbf{V}$; on a aussi $v([p_1, p_2 \,/\, A_1, A_2]) = v(C) = \mathbf{V}$.

Démonstration du lemme. Dans le cas où la substitution est indépendante, si r_i désigne le nombre d'occurrences de p_i dans C, il suffit d'appliquer $r_1 + \cdots + r_n$ fois le lemme de remplacement (ou n fois le théorème de l'échange). On laisse au lecteur l'extension au cas des substitutions non indépendantes.

Théorème de substitution uniforme. Soient $C, A_1, \ldots, A_n$ des formules et $p_1, \ldots, p_n$ des propositions deux à deux distinctes ; si C est une tautologie, alors $C[p_1, \ldots, p_n / A_1, \ldots, A_n]$ est une tautologie.
Démonstration. On suppose d'abord que la substitution est indépendante ; aucun p_i n'apparaît donc dans $\{A_1, \ldots, A_n\}$, pas plus que dans $C' =_{def} C[p_1, \ldots, p_n \,/\, A_1, \ldots, A_n]$. Soit v, une interprétation quelconque de C', et w l'*extension* de v obtenue en posant $w(p_i) =_{def} v(A_i)$. Le lemme de substitution uniforme implique $w(C') = w(C)$. Par hypothèse on a $w(C) = \mathbf{V}$ et par construction on a $w(C') = v(C')$. On a donc $v(C') = \mathbf{V}$.
Remarque. Si les p_i intervenaient dans les A_k, la technique pourrait ne pas fonctionner. De $\models p \equiv \neg\neg p$, on ne déduit pas immédiatement que $\models (p \vee r) \equiv \neg\neg(p \vee r)$ car l'interprétation $v : v(p) = \mathbf{F}, v(r) = \mathbf{V}$ telle que $v(A) = v(p \vee r) = \mathbf{V}$ n'admet pas d'extension w telle que $w(p) = \mathbf{V}$. Le remède est simple. De $\models p \equiv \neg\neg p$ on déduit $\models q \equiv \neg\neg q$, d'où on déduit $\models (p \vee r) \equiv \neg\neg(p \vee r)$.
Suite de la démonstration. Si en revanche les p_i interviennent dans $\{A_1, \ldots, A_n\}$, on se donne une famille de nouveaux atomes q_i. Si C est une tautologie, alors $C'' =_{def} C(p_1/q_1, \ldots, p_n/q_n)$ est une tautologie. D'autre part, C' peut s'écrire $C''[q_1, \ldots, q_n \,/\, A_1, \ldots, A_n]$, où les q_i n'interviennent pas dans les A_k ; C' est donc une tautologie.
Remarque. Où l'exigence d'uniformité de la substitution est-elle utilisée dans cette démonstration ?

Tables de vérité abrégées. On vérifie un énoncé tel que $(p \vee q) \leftrightarrow (q \vee p)$ au moyen d'une table de vérité (de quatre lignes). Il semble naturel de vérifier un énoncé tel que $(A \vee B) \leftrightarrow (B \vee A)$ de la même manière, sans recourir à un théorème de substitution uniforme. Cette méthode est acceptable et se justifie comme suit. On peut voir la "pseudo-table" relative à $(A \vee B) \leftrightarrow (B \vee A)$ comme une abréviation de la table complète, potentiellement très longue, relative à une instance du schéma, telle que

$$[(p \Rightarrow q) \vee r] \leftrightarrow [r \vee (p \Rightarrow q)]\,.$$

[20] La démonstration est laissée au lecteur. On pourra utiliser la décomposition d'une affectation n-uple $(x_1, \ldots, x_n) := (e_1, \ldots, e_n)$ en une séquence équivalente de $2n$ affectations simples $t_1 := e_1; \ldots; t_n := e_n; x_1 := t_1; \ldots; x_n := t_n$, où les t_i sont n "nouvelles" variables.

Il est clair que chaque ligne v de la table complète est "représentée" dans la "pseudo-table", par la ligne qui attribue à A la valeur $v(p \Rightarrow q)$ et à B la valeur r. En revanche, certaines lignes de la pseudo-table peuvent ne correspondre à aucune ligne de la table complète. C'est le cas par exemple si A est instancié par une formule valide : les lignes de la pseudo-table concernant les cas où A est faux n'ont pas de correspondant dans la table complète. Cela a la conséquence suivante.

- Si C est une formule valide, alors $C(p_1/A_1, \ldots, p_n/A_n)$ est une formule valide ;
- Si C est une formule inconsistante, alors $C(p_1/A_1, \ldots, p_n/A_n)$ est une formule inconsistante ;
- Si C est une formule simplement consistante, on ne peut rien dire.

Donnons un contre-exemple très simple pour le dernier cas : soit $C =_{def} p$. On voit que C est simplement consistante, tandis que $C[p/(q \vee \neg q)]$ est valide et que $C[p/(q \wedge \neg q)]$ est inconsistante.

2.5 Quelques théorèmes sémantiques

2.5.1 Interpolation et définissabilité

Introduction. Considérons des fonctions réelles f et g de domaine $\mathbf{R}^2$ et un ensemble $D \subset \mathbf{R}^3$ tels que

$$\forall (x, y, z) \in D : f(x, y) \leq g(x, z)\,.$$

Existe-t-il une fonction *interpolante* $h : \mathbf{R} \longrightarrow \mathbf{R}$ telle que

$$\forall (x, y, z) \in D : [f(x, y) \leq h(x) \leq g(x, z)]?$$

Cela dépend du domaine D. Si par exemple $D = D_1 \times D_2 \times D_3$, deux interpolantes possibles sont

$$x \mapsto \sup_{y \in D_2} f(x, y) \text{ et } x \mapsto \inf_{z \in D_3} g(x, z)$$

Si en revanche on a $D = \{(0, 0, 0), (0, 1, 1)\}$, l'hypothèse devient

$$f(0, 0) \leq g(0, 0) \ \wedge \ f(0, 1) \leq g(0, 1)$$

et la thèse devient

$$f(0, 0) \leq h(0) \leq g(0, 0) \ \wedge \ f(0, 1) \leq h(0) \leq g(0, 1)\,.$$

On voit que, dans ce cas, l'interpolante peut ne pas exister.

Dans $\mathbf{R}^+$, l'équation $x^2 = x + 1$ caractérise un nombre unique (le "nombre d'or"). Autrement dit, si $y^2 = y + 1$ et $z^2 = z + 1$, on a $y = z$, et l'équation définit implicitement le nombre d'or. L'explicitation de ce nombre n'est possible que si on introduit une fonction non rationnelle (la racine carrée) ; on a alors $x = (1 + \sqrt{5})/2$.

La définissabilité et l'interpolation correspondent à la résolution d'équations et d'inéquations. Ces concepts existent aussi en logique, où "$\leq$" et "$=$" deviennent respectivement "$\Rightarrow$" et "$\equiv$".

Théorème d'interpolation de Craig. En logique propositionnelle, l'interpolation doit permettre notamment l'optimisation des circuits digitaux ; une formule comportant n variables propositionnelles distinctes correspond à un circuit digital à n entrées et une sortie. Le plus souvent, un circuit digital n'est pas complètement spécifié et le concepteur peut mettre à profit les degrés de liberté tolérés par la spécification pour obtenir un circuit aussi simple que possible. Dans le cas où la spécification prend la forme d'un intervalle logique, le théorème d'interpolation donne lieu à une technique de simplification.

Théorème. Soient A et B deux formules propositionnelles. Si $\models A \Rightarrow B$, il existe une formule C, ne contenant que des propositions communes à A et B, telle que $\models A \Rightarrow C$ et $\models C \Rightarrow B$.
Démonstration. On raisonne par *induction* sur l'ensemble Π des propositions communes à A et B. Cela signifie que l'on démontre d'abord le théorème dans le cas particulier où l'ensemble Π est vide (cas de base). On suppose ensuite que le théorème est vrai dans le cas d'un ensemble, quelconque mais fixé, ne contenant pas une proposition, elle aussi quelconque mais fixée (cette supposition est l'hypothèse inductive), puis on démontre que le théorème reste vrai dans le cas de cet ensemble augmenté de cette proposition.
Cas de base. Si $\Pi = \emptyset$, $\models A \Rightarrow B$ implique que A est inconsistante (et on choisit $C =_{def} false$) ou que B est valide (et on choisit $C =_{def} true$). Cela se démontre par l'absurde. S'il existait des interprétations u et v (de domaines disjoints) telles que $u(A) = \mathbf{V}$ et $v(B) = \mathbf{F}$, l'interprétation $w =_{def} u \cup v$ serait telle que $w(A \Rightarrow B) = \mathbf{F}$.
Cas inductif. Si $p \in \Pi$, l'hypothèse inductive affirme l'existence d'interpolantes C_T et C_F relatives à $A(p/true), B(p/true)$ et à $A(p/false), B(p/false)$, respectivement.
On vérifie immédiatement que la formule $(p \wedge C_T) \vee (\neg p \wedge C_F)$ interpole A et B.

Théorème de définissabilité de Beth. *Théorème.* Soit A une formule ne contenant ni q ni r telle que $[A(p/q) \wedge A(p/r)] \Rightarrow (q \equiv r)$ est une tautologie. Il existe une formule B ne contenant pas p, q et r telle que $A \Rightarrow (p \equiv B)$ soit une tautologie.

Remarque. L'hypothèse affirme que A *caractérise* (la valeur de) la proposition p, donc que A définit implicitement p. La thèse affirme qu'une certaine formule B existe, qui définit *explicitement* p. Le théorème affirme donc qu'en logique propositionnelle, une définition implicite peut toujours être explicitée. De ce point de vue, le domaine des formules propositionnelles se distingue de la plupart des domaines mathématiques, dans lesquels l'explicitation des définitions implique souvent l'introduction d'outils nouveaux. Dans une logique plus puissante que la logique propositionnelle, adaptée à l'intelligence artificielle, on peut concevoir, sur base du théorème de Beth, des techniques permettant d'expliciter une information donnée implicitement (résolution d'énigmes par exemple).
Démonstration. On a successivement
$\models [A(p/q) \wedge A(p/r)] \Rightarrow (q \equiv r)]$,
$\models [A(p/q) \wedge A(p/r)] \Rightarrow (q \Rightarrow r)]$,
$\models [A(p/q) \wedge A(p/r) \wedge q] \Rightarrow r$,
$\models [A(p/q) \wedge q] \Rightarrow [A(p/r) \Rightarrow r]$.
Soit B une interpolante (théorème de Craig), ne contenant pas p, q, r. On a
$\models [A(p/q) \wedge q] \Rightarrow B$, et donc
$\models A(p/q) \Rightarrow (q \Rightarrow B)$, et par substitution
$\models A(p/r) \Rightarrow (r \Rightarrow B)$.

D'autre part, on a
$\models B \Rightarrow [A(p/r) \Rightarrow r]$, d'où
$\models [B \wedge A(p/r)] \Rightarrow r$, d'où
$\models A(p/r) \Rightarrow (B \Rightarrow r)$, et par substitution
$\models A(p/q) \Rightarrow (B \Rightarrow q)$.
On en déduit la thèse, sous la forme
$\models A(p/q) \Rightarrow (q \equiv B)$, ou sous la forme
$\models A(p/r) \Rightarrow (r \equiv B)$, ou encore
$\models A \Rightarrow (p \equiv B)$.

2.5.2 Théorème de compacité

Préliminaires. La majorité des théorèmes mathématiques évoquent, explicitement ou non, des objets infinis ou des ensembles infinis d'objets. La difficulté vient de ce que les propriétés des objets et ensembles finis ne s'étendent pas systématiquement aux objets et ensembles infinis. Par exemple, l'addition finie est toujours associative et commutative ; l'addition infinie (séries) ne l'est que dans des cas particuliers. Certains ensembles structurés, dits *compacts*, héritent de la plupart des propriétés intéressantes des ensembles finis.

En logique propositionnelle, l'ensemble des interprétations devient infini si l'ensemble des propositions est lui-même infini. La compacité est ici la faculté de ne considérer qu'un sous-ensemble fini (de propositions, d'interprétations, de formules, ...) pour tirer des conclusions portant sur un ensemble infini. Plus concrètement, si E est un ensemble de formules et A une formule, on souhaite que, si A est conséquence logique de E, alors il existe un sous-ensemble fini $E' \subset E$ tel que A soit conséquence logique de E'. D'une manière équivalente, il faudrait que tout ensemble inconsistant (par exemple $E \cup \{\neg A\}$) admette un sous-ensemble fini inconsistant (par exemple $E' \cup \{\neg A\}$). Ou encore (contraposition), que chaque ensemble *finiment consistant*, c'est-à-dire dont toutes les parties finies sont consistantes, soit lui-même consistant.

Ensembles finiment consistants maximaux. *Définitions.* Un ensemble est *finiment consistant* si tous ses sous-ensembles finis sont consistants.[21] Un ensemble finiment consistant est *maximal* s'il n'admet pas de sur-ensemble finiment consistant.[22]

Théorème. Soient Π un ensemble de propositions et Φ l'ensemble des formules construites sur Π. Un ensemble $E \subset \Phi$ est finiment consistant maximal si et seulement s'il existe une interprétation v sur Π telle que $E = \{\varphi \in \Phi : v(\varphi) = \mathbf{V}\}$.
Corollaire. Tout ensemble finiment consistant maximal est consistant et admet un modèle unique.

Démonstration. La preuve montre la correspondance biunivoque entre les interprétations et les ensembles finiment consistants maximaux (pour un lexique fixé).

[21] Tout ensemble consistant est donc finiment consistant ; le but de cette section est de montrer que l'inverse est vrai aussi.

[22] On écrira parfois "f.c." et "f.c.max" au lieu de "finiment consistant" et "finiment consistant maximal".

La condition est suffisante. L'ensemble $E = \{\varphi \in \Phi : v(\varphi) = \mathbf{V}\}$ est (finiment) consistant, puisqu'il admet le modèle (unique) v, et est maximal, parce que si $\psi \notin E$, l'ensemble $E \cup \{\psi\}$ contient le sous-ensemble fini inconsistant $\{\neg\psi, \psi\}$.
La condition est nécessaire. On se restreint au cas où le lexique Π est dénombrable, soit $\Pi = \{p_1, p_2, \ldots\}$. Soit E un sous-ensemble f.c. maximal de Φ.

- Pour tout i, E contient exactement un des éléments de la paire $\{p_i, \neg p_i\}$. D'une part, il ne peut contenir les deux éléments, puisque $\{p_i, \neg p_i\}$ est inconsistant. D'autre part, si $p_i \notin E$, l'ensemble $E \cup \{p_i\}$ n'est pas finiment consistant et E admet un sous-ensemble fini E' tel que $E' \cup \{p_i\}$ est inconsistant ; on a alors $E' \models \{\neg p_i\}$. On en déduit que $E \cup \{\neg p_i\}$ est finiment consistant [si $E'' \subset E$, tout modèle de $E'' \cup E'$ est un modèle de $E'' \cup \{\neg p_i\}$] d'où, vu la maximalité, $\neg p_i \in E$.
- Pour tout i, soit ℓ_i l'unique élément de $\{p_i, \neg p_i\}$ appartenant à E ; ces éléments déterminent une interprétation unique, rendant vrais tous les ℓ_i. On note v cette interprétation, dont on va montrer qu'elle est celle requise par l'énoncé.
- On commence par démontrer l'inclusion $E \subset \{\varphi \in \Phi : v(\varphi) = \mathbf{V}\}$. Soit $\varphi \in E$ et $\{p_{i_1}, \ldots, p_{i_n}\}$ les propositions intervenant dans φ. Comme E est finiment consistant, son sous-ensemble $\{\ell_{i_1}, \ldots, \ell_{i_n}, \varphi\}$ est consistant, d'où $v(\varphi) = \mathbf{V}$.
- On conclut en observant que, l'ensemble E étant finiment consistant maximal, l'inclusion $E \subset \{\varphi \in \Phi : v(\varphi) = \mathbf{V}\}$ entraîne l'égalité $E = \{\varphi \in \Phi : v(\varphi) = \mathbf{V}\}$.

Théorème. Tout ensemble finiment consistant est consistant.
Remarque. Il suffit de prouver que tout ensemble finiment consistant est inclus dans un ensemble finiment consistant maximal.
Démonstration. Soit D un ensemble finiment consistant. On pose $E_0 = D$ et, si $n > 0$, $E_n = E_{n-1} \cup \{p_n\}$ si cet ensemble est finiment consistant, $E_n = E_{n-1} \cup \{\neg p_n\}$ sinon.

On démontre par récurrence que tous les E_n sont finiment consistants. C'est trivial pour $n = 0$. Pour E_n, c'est trivial si $E_{n-1} \cup \{p_n\}$ est finiment consistant. Sinon, il existe un sous-ensemble fini $E' \subset E_{n-1}$ tel que $E' \cup \{p_n\}$ est inconsistant, et donc que $E' \models \neg p_n$. Dans ce cas, $E_n = E_{n-1} \cup \{\neg p_n\}$ est finiment consistant, car pour tout sous-ensemble fini $E'' \subset E_{n-1}$, tout modèle de $E'' \cup E'$ est aussi un modèle de $E'' \cup \{\neg p_n\}$.
On pose $E =_{def} \bigcup_n E_n$. L'intersection $\{p_i, \neg p_i\} \cap E$ contient un élément unique ℓ_i. Ces ℓ_i déterminent une interprétation unique v telle que $v(\varphi) = \mathbf{V}$ pour tout $\varphi \in E$. L'ensemble D, comme l'ensemble E, est donc inclus dans l'ensemble maximal $\{\varphi \in \Phi : v(\varphi) = \mathbf{V}\}$.

Remarque. Le théorème de compacité facilite l'emploi de l'outil logique, mais indique aussi une certaine faiblesse de cet outil. Par exemple, en arithmétique (théorie des nombres entiers), un ensemble infini de formules peut être inconsistant tout en étant finiment consistant. Si z_0 est une constante sur le domaine $\mathbf{Z}$, on pose $E_{z_0} = \{(z > z_0) : z \in \mathbb{Z}\}$. Cet ensemble est inconsistant, puisque pour toute interprétation v, l'entier $v(z_0)$ admet des minorants. Néanmoins, tout sous-ensemble fini de E_{z_0} est consistant. Un tel ensemble s'écrit $\{(z > z_0) : z \in A\}$, où A est un ensemble fini d'entiers. Un modèle v s'obtient en posant $v(z_0) = (\inf A) - 1$. Cela montre simplement que le calcul des propositions ne permet pas d'exprimer toute la théorie arithmétique.

Variante de la démonstration. On peut combiner la construction d'un sur-ensemble maximal et la démonstration du théorème de compacité. Il suffit d'observer que si Π est dénombrable, alors Φ l'est aussi ; en effet, on a $\Phi = \bigcup_n \Phi_n$, où Φ_n est l'ensemble (fini) des formules construites

avec le lexique $\{p_1, \ldots, p_n\}$ et comportant au plus n connecteurs. On peut alors considérer une énumération $(\varphi_1, \varphi_2, \ldots)$ de l'ensemble Φ et récrire la démonstration comme suit.

Soit D un ensemble finiment consistant. On pose $E_0 = D$ et, si $n > 0$, $E_n = E_{n-1} \cup \{\varphi_n\}$ si cet ensemble est finiment consistant, $E_n = E_{n-1} \cup \{\neg\varphi_n\}$ sinon.

On démontre par récurrence que tous les E_n sont finiment consistants. C'est trivial pour $n = 0$. Pour E_n, c'est trivial si $E_{n-1} \cup \{\varphi_n\}$ est finiment consistant. Sinon, il existe un sous-ensemble fini $E' \subset E_{n-1}$ tel que $E' \cup \{\varphi_n\}$ est inconsistant, et donc tel que $E' \models \neg\varphi_n$. Dans ce cas, $E_n = E_{n-1} \cup \{\neg\varphi_n\}$ est finiment consistant, car pour tout sous-ensemble fini $E'' \subset E_{n-1}$, tout modèle de $E'' \cup E'$ est aussi un modèle de $E'' \cup \{\neg\varphi_n\}$. On pose $E =_{def} \bigcup_n E_n$. Par construction, pour tout $i > 0$, l'intersection $\{p_i, \neg p_i\} \cap E$ contient un élément unique ℓ_i. Ces éléments déterminent une interprétation v, dont on montre qu'elle est un modèle de E. En effet, soit $\varphi \in E$ et $\{p_{i_1}, \ldots, p_{i_n}\}$ les propositions intervenant dans φ. Soit k le plus petit naturel tel que l'ensemble E_k comporte tous les éléments de $\{\ell_{i_1}, \ldots, \ell_{i_n}, \varphi\}$ (k existe toujours). Comme E_k est finiment consistant, son sous-ensemble $\{\ell_{i_1}, \ldots, \ell_{i_n}, \varphi\}$ est consistant et admet un modèle. Ce modèle ne peut être que v (ou plus exactement la restriction de v au lexique $\{p_{i_1}, \ldots, p_{i_n}\}$), d'où $v(\varphi) = \mathbf{V}$.

Remarquons qu'en arithmétique ce résultat est faux. L'ensemble

$$\{n > 0, n > 1, \ldots, n > 143, \ldots\}$$

est inconsistant, car aucun nombre n'est plus grand que tous les autres, mais tous ses sous-ensembles finis sont consistants.

Pourquoi ce théorème est-il important ? Pour plusieurs raisons, mais nous n'en citons que deux. La première raison est technique. Supposons qu'une formule A soit conséquence logique de l'ensemble infini E de formules. A priori, on pourrait craindre qu'une infinité de formules de E soient des hypothèses nécessaires pour obtenir la conclusion A. Le théorème de compacité montre que cette crainte n'est pas fondée. En effet, si A est conséquence logique de E, alors $E \cup \{\neg A\}$ est inconsistant et, par le théorème de compacité, il existe un sous-ensemble fini E' de E tel que $E' \cup \{\neg A\}$ soit inconsistant, et donc tel que A soit conséquence logique de E'.

La seconde raison est plus philosophique. L'une des motivations de Frege, dans sa tentative remarquablement réussie de formaliser la logique, était de "réduire" les mathématiques à la logique. Ce "logicisme", dans la lignée du projet leibnizien de "calculus ratiocinator", ne peut réussir que de manière très partielle. Le théorème de compacité (surtout dans le cadre prédicatif, que nous aborderons plus loin) montre que l'arithmétique ne peut se réduire à la logique. Les célèbres résultats d'incomplétude de Gödel montrent que cela a des conséquences importantes.

3 Procédures de décision analytiques

Le théorème de la déduction permet de ramener le problème fondamental de la logique à la détermination de la consistance d'un ensemble de formules, en réduisant la question "la formule A est-elle conséquence logique de l'ensemble de formules E ?" à la question "l'ensemble de formules $E \cup \{\neg A\}$ est-il inconsistant ?". En pratique, on développe surtout des algorithmes de détermination de la consistance d'une formule ou d'un ensemble de formules. Un tel algorithme permet aussi de résoudre le problème de la validité : un ensemble de formules est valide si et seulement si tous ses éléments sont valides[23] et une formule est valide si et seulement si sa négation est inconsistante. Une formule est simplement consistante, ou contingente, si elle n'est ni valide ni inconsistante.

3.1 La méthode des tables de vérité

Le principe de cette méthode est très simple. Toute formule contient un nombre fini d'atomes et admet donc un nombre fini d'interprétations. En conséquence, on peut déterminer la valeur de vérité de la formule pour toutes ses interprétations. On présente souvent le résultat sous forme d'une *table de vérité*, appelée aussi *tableau matriciel*.

On voit immédiatement que la méthode est très inefficace ; si la formule à analyser contient n atomes distincts, elle admet 2^n interprétations. L'algorithme est donc exponentiel (en temps et espace) en fonction du nombre de propositions intervenant dans la formule. Cela signifie que le temps nécessaire à la mise en œuvre de cet algorithme augmente très vite en fonction du nombre de propositions intervenant dans la formule. La figure 15 illustre la méthode des tables de vérité pour trois formules simples.

Le problème de la consistance en logique des propositions est NP-complet. On ne s'attend donc pas à trouver un algorithme polynomial mais, en pratique, on escompte une méthode qui soit exponentielle en pire cas, et non dans tous les cas. La suite de ce chapitre est consacrée à des méthodes qui, le plus souvent, sont nettement meilleures que la méthode des tables de vérité.

On peut améliorer la méthode des tables de vérité en utilisant diverses simplifications. La plus importante consiste à ne pas attendre la fin de la construction de la table pour tirer une conclusion.

Considérons l'exemple de la formule

$$(p \Rightarrow q) \vee (q \Rightarrow r)\,.$$

C'est une disjonction de deux implications. La première implication n'est fausse que si p est vrai et q faux, mais dans ce cas la deuxième implication est vraie ; la formule est donc valide.

Nous n'approfondissons pas ici les raffinements que l'on peut apporter à la méthode des tables de vérité, parce qu'il existe d'autres méthodes nettement plus efficaces.[24]

[23] Rappelons ici qu'un ensemble de formules consistantes peut être inconsistant.

[24] La méthode des tables de vérité (avec simplifications) reste intéressante pour résoudre certaines questions théoriques et surtout pour analyser "à la main" des formules très courtes.

p	q	$p \Rightarrow q$	$\neg q \Rightarrow \neg p$	$(p \Rightarrow q) \Rightarrow (\neg q \Rightarrow \neg p)$
V	**V**	**V**	**V**	**V**
V	**F**	**F**	**F**	**V**
F	**V**	**V**	**V**	**V**
F	**F**	**V**	**V**	**V**

Table de vérité de la formule valide $(p \Rightarrow q) \Rightarrow (\neg q \Rightarrow \neg p)$.

p	q	$p \wedge q$
V	**V**	**V**
V	**F**	**F**
F	**V**	**F**
F	**F**	**F**

Table de vérité de la formule simplement consistante $p \wedge q$.

p	q	$p \vee q$	$\neg p$	$\neg q$	$(p \vee q) \wedge \neg p \wedge \neg q$
V	**V**	**V**	**F**	**F**	**F**
V	**F**	**V**	**F**	**V**	**F**
F	**V**	**V**	**V**	**F**	**F**
F	**F**	**F**	**V**	**V**	**F**

Table de vérité de la formule inconsistante $(p \vee q) \wedge \neg p \wedge \neg q$.

FIG. 15 – Application de la méthode des tables de vérité.

3.2 Les tableaux sémantiques

3.2.1 Introduction

La méthode des tables de vérité consiste à passer en revue toutes les interprétations possibles d'une formule φ donnée. La valeur attribuée par une interprétation à la formule φ est calculée en parcourant l'arbre syntaxique de φ du bas vers le haut, des feuilles vers la racine. On utilise le fait que la valeur de vérité d'une formule est déterminée par les valeurs de ses composants.

La méthode des tableaux sémantiques consiste en la recherche systématique d'un modèle d'une formule φ donnée.[25] Le fait d'imposer une valeur de vérité à une formule peut déterminer univoquement la valeur des composants de la formule, ou au contraire laisser plusieurs choix possibles. La recherche systématique d'un modèle conduit à la construction progressive d'une

[25] On peut aussi rechercher un antimodèle, une interprétation qui falsifie la formule φ. Il est inutile de considérer explicitement cette variante, puisqu'un modèle de φ est un antimodèle de $\neg\varphi$.

structure arborescente particulière, appelée *tableau sémantique.*[26]

3.2.2 Technique de construction du tableau

Un exemple introductif. Considérons la formule $\varphi =_{def} (p \Rightarrow q) \wedge \neg(p \Rightarrow r)$, en vue de la recherche systématique d'un modèle. La formule étant une conjonction, tout modèle de φ sera un modèle de ses deux composants (et réciproquement), donc un modèle de l'ensemble $\{p \Rightarrow q\,,\, \neg(p \Rightarrow r)\}$. Le deuxième élément de cet ensemble est la négation d'une implication ; il sera vrai si et seulement si l'antécédent est vrai et le conséquent faux. L'analyse de φ est donc réduite à celle de l'ensemble $\{p \Rightarrow q\,,\, p\,,\, \neg r\}$. Enfin, le premier élément est une implication, que l'on rend vraie en falsifiant l'antécédent ou en vérifiant le conséquent ; il y a là deux possibilités (non exclusives). Les modèles de φ sont donc, d'une part, ceux de l'ensemble $\{\neg p\,,\, p\,,\, \neg r\}$ et, d'autre part, ceux de l'ensemble $\{q\,,\, p\,,\, \neg r\}$.

A ce stade, la décomposition de la formule est achevée parce que les ensembles ne comportent plus que des *littéraux*. Un littéral est un atome (littéral positif) ou la négation d'un atome (littéral négatif). En effet, un ensemble de littéraux est consistant si et seulement s'il ne contient pas simultanément un littéral et son opposé, c'est-à-dire une *paire complémentaire* du type $\{p, \neg p\}$; ceci se détermine par simple inspection. On voit notamment que l'ensemble $\{\neg p\,,\, p\,,\, \neg r\}$ est inconsistant et que l'ensemble $\{q\,,\, p\,,\, \neg r\}$ est consistant ; les modèles de la formule φ sont ceux de ce dernier ensemble.

La méthode des tableaux sémantiques consiste donc à réduire la question (complexe) "la formule A est-elle consistante ?" à la question (triviale) "la famille (finie) $\mathcal{A}$ d'ensembles de littéraux contient-elle un élément consistant ?".

Il est commode d'organiser la recherche sous forme d'un arbre, appelé *tableau sémantique*. Celui correspondant à la formule φ est représenté à la figure 16.

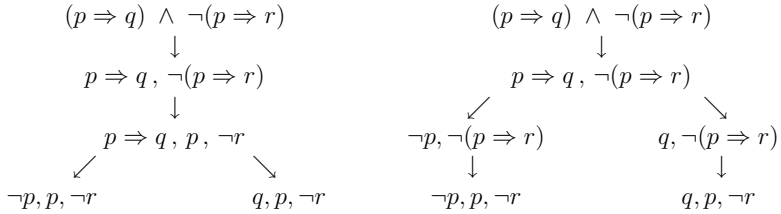


FIG. 16 – Deux tableaux sémantiques.

Une formule peut donner lieu à plusieurs tableaux sémantiques différents suivant l'ordre d'application des règles de construction, mais tous conduisent à la même conclusion (la bonne !) concernant la consistance de la formule. La signification commune des deux tableaux de la figure 16 est

[26] Cette structure est bien un arbre, mais ne doit pas être confondue, d'une part, avec la notion d'arbre syntaxique déjà introduite ni, d'autre part, avec la notion d'arbre sémantique qui sera introduite plus loin.

Une interprétation est un modèle de la formule $(p \Rightarrow q) \ \wedge \ \neg(p \Rightarrow r)$
si et seulement si c'est un modèle de l'un des ensembles $\{\neg p, p, \neg r\}$, $\{q, p, \neg r\}$.

Construction des tableaux sémantiques. La construction des tableaux sémantiques est basée sur la partition des formules en trois catégories :

- les littéraux ;
- les formules conjonctives ;
- les formules disjonctives.

La formule $\neg(X \Rightarrow Y)$ est conjonctive car elle est équivalente à la conjonction des deux formules (plus simples) X et $\neg Y$. La formule $X \Rightarrow Y$ est disjonctive car elle est équivalente à la disjonction de $\neg X$ et Y. Le connecteur $\equiv$ est exclu ici,[27] et on convient d'assimiler $\neg\neg X$ a X.

La construction est basée sur deux types de règles de décomposition de formules : les règles de *prolongation* (type α) et les règles de *ramification* (type β). Dans le contexte des tableaux sémantiques on utilise les règles α pour les formules conjonctives ; les règles β pour les formules disjonctives.[28] La figure 17 répertorie les formules conjonctives et disjonctives, ainsi que les résultats de l'application à ces formules des règles de prolongation et de ramification, respectivement.[29]

α	α_1	α_2
$A_1 \wedge A_2$	A_1	A_2
$\neg(A_1 \vee A_2)$	$\neg A_1$	$\neg A_2$
$\neg(A_1 \Rightarrow A_2)$	A_1	$\neg A_2$
$\neg(A_1 \Leftarrow A_2)$	$\neg A_1$	A_2

β	β_1	β_2
$B_1 \vee B_2$	B_1	B_2
$\neg(B_1 \wedge B_2)$	$\neg B_1$	$\neg B_2$
$B_1 \Rightarrow B_2$	$\neg B_1$	B_2
$B_1 \Leftarrow B_2$	B_1	$\neg B_2$

FIG. 17 – Les règles de décomposition.

Le processus général de construction d'un tableau sémantique pour une formule donnée φ est décrit à la figure 18. Ce tableau est un arbre dont chaque nœud est étiqueté par un ensemble de formules. Un nœud est *terminal* quand son étiquette ne comporte que des littéraux. Quand la construction du tableau est achevée, toutes les feuilles sont des nœuds terminaux. On convient de marquer un nœud terminal par $\bigcirc$ si l'étiquette est consistante (feuille *ouverte*), et par $\times$ si l'étiquette est inconsistante (feuille *fermée*).

On utilise parfois des *assertions* plutôt que des formules, une assertion étant l'attribution d'une valeur de vérité à une formule. La figure 19 comporte un tableau classique, en notation "formule", et sa variante *signée*, en notation "assertion".

[27] On remplacera donc une équivalence $X \equiv Y$ par une formule conjonctive $(X \Rightarrow Y) \wedge (Y \Rightarrow X)$, ou par une formule disjonctive $(X \wedge Y) \vee (\neg X \wedge \neg Y)$, au choix. On élimine de même les formules du type $X \oplus Y$.

[28] Ce point sera nuancé plus loin.

[29] Dans la suite, on notera souvent α une formule conjonctive et β une formule disjonctive ; les composants respectifs seront notés respectivement α_1 et α_2, et β_1 et β_2. Soulignons qu'en général il s'agit de composants sémantiques et non de composants syntaxiques ; par exemple, $\neg p$ n'est pas un composant syntaxique de la formule disjonctive $p \Rightarrow q$.

- *Initialisation.* On crée une racine étiquetée $\{\varphi\}$.
- *Itération.* On sélectionne une feuille non marquée ℓ, d'étiquette $U(\ell)$.
  - Si $U(\ell)$ est un ensemble de littéraux :
    - si $U(\ell)$ contient une paire complémentaire, alors marquer ℓ comme étant fermée ;
    - sinon, marquer ℓ comme étant ouverte.
  - Si $U(\ell)$ n'est pas un ensemble de littéraux, sélectionner une formule dans $U(\ell)$:
    - si c'est une α-formule A, créer un nouveau nœud ℓ', descendant de ℓ, et étiqueter ℓ' avec
      $$U(\ell') = (U(\ell) - \{A\}) \cup \{\alpha_1, \alpha_2\};$$
    - si c'est une β-formule B, créer deux nouveaux nœuds ℓ' et ℓ'', descendants de ℓ, et étiqueter ℓ' avec
      $$U(\ell') = (U(\ell) - \{B\}) \cup \{\beta_1\}$$
      et étiqueter ℓ'' avec
      $$U(\ell'') = (U(\ell) - \{B\}) \cup \{\beta_2\}.$$
- *Terminaison.* La construction est achevée quand toutes les feuilles sont marquées '$\times$' ou '$\bigcirc$'.

FIG. 18 – Algorithme de construction d'un tableau sémantique.

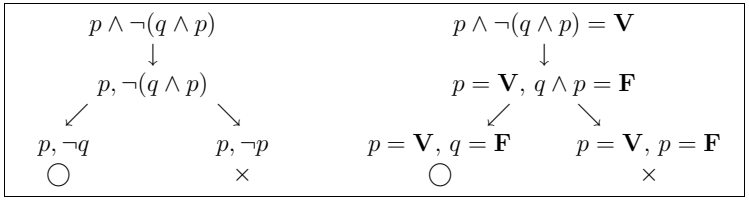
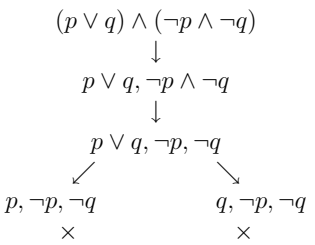


FIG. 19 – Tableau classique et tableau signé.

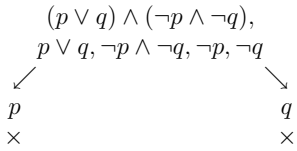
Remarque. Si on adopte cette variante, les liens α-conjonction et β-disjonction ne sont plus valable tels quels ; par exemple, l'assertion $p \wedge q = \mathbf{V}$ est de type α, l'assertion $p \wedge q = \mathbf{F}$ est de type β. En revanche, les liens α-prolongation et β-ramification subsistent. C'est pour éviter ces complications (à tout le moins, ces apparences de complication) que nous n'avons pas directement introduit les tableaux signés, pourtant fréquemment utilisés.

Programmation de la méthode. La méthode des tableaux sémantiques est facilement programmable. Pour économiser l'espace mémoire, on convient de ne pas créer un nouveau nœud lors de l'application d'une règle α, et de ne pas étiqueter un nœud n avec une formule

qui étiquète déjà un *ancêtre* de n. Avec ces conventions, le tableau



prend la forme plus compacte suivante :



Une autre économie possible est de ne pas étiqueter les nœuds directement avec des sous-formules, mais plutôt avec des pointeurs vers les sous-arbres correspondants dans l'arbre syntaxique. Enfin, certaines branches d'un tableau sémantique sont très semblables. Il est possible d'éviter aussi les redondances de ce type en utilisant non plus une structure d'arbre, mais une structure de graphe sans cycle. Ces raffinements sortent du cadre de ces notes (voir [L2] pour plus de détails).

Terminaison de l'algorithme de construction. *Théorème.* La construction d'un tableau sémantique se termine.
Principe de la démonstration. Chaque étape remplace une formule par une ou deux formules strictement plus simples. Comme la complexité des formules est limitée, on ne peut appliquer qu'un nombre fini d'étapes.
Remarque. On peut, sans restriction essentielle, supposer que le connecteur d'équivalence n'est pas employé. Cette restriction simplifie la forme de la mesure W définie dans la démonstration.
Démonstration. Soit $\mathbf{V}$ le tableau d'une formule A, à une étape quelconque de sa construction, soit ℓ une feuille quelconque de $\mathbf{V}$; soient $b(\ell)$ le nombre de connecteurs binaires dans $U(\ell)$ et $n(\ell)$ le nombre de négations dans $U(\ell)$.
On définit $W(\ell) = 2b(\ell) + n(\ell)$. Quelle que soit la règle utilisée, toute étape de la construction crée un nouveau nœud ℓ' ou deux nouveaux nœuds ℓ', ℓ'' tels que $W(\ell') < W(\ell)$ et $W(\ell'') < W(\ell)$. Or, $W(\ell)$ prend ses valeurs dans $\mathbb{N}$; il ne peut donc y avoir de branche infinie dans $\mathbf{V}$.
Remarque. La démonstration peut être étendue au cas où "$\equiv$" et "$\oplus$" apparaissent dans la formule (exercice).

Un tableau *complet* (dont la construction est achevée) est *fermé* si toutes ses feuilles sont fermées. Sinon, il est *ouvert*.

3.2.3 Propriétés de la méthode des tableaux sémantiques

La méthode des tableaux sémantiques est le plus souvent utilisée pour montrer la validité d'une formule (ou l'inconsistance de sa négation), ou encore pour montrer l'inconsistance d'un ensemble fini de formules. On souhaite naturellement que la méthode donne uniquement des résultats corrects ; elle ne peut conclure, par exemple, à l'inconsistance d'une formule, que si la formule est effectivement inconsistante. Cette propriété est *l'adéquation* de la méthode. D'autre part, on souhaite que, si une formule est inconsistante, la méthode mette ce fait en évidence ; c'est la propriété de *complétude*. En résumé, une méthode est adéquate si elle est correcte ; elle est complète si elle est assez puissante.

Dans le cas présent, prouver l'adéquation revient à prouver l'un des énoncés suivants :
- si $T(A)$ est fermé, alors A est inconsistante ;
- si $T(\neg B)$ est fermé, alors B est valide ;
- si A est consistante, alors $T(A)$ est ouvert ;
- si B n'est pas valide, alors $T(\neg B)$ est ouvert.

Prouver la complétude revient à prouver la réciproque, c'est-à-dire l'un des énoncés suivants :
- si A est inconsistante, alors $T(A)$ est fermé ;
- si B est valide, alors $T(\neg B)$ est fermé ;
- si $T(A)$ est ouvert, alors A est consistante ;
- si $T(\neg B)$ est ouvert, alors B n'est pas valide.

Théorème d'adéquation. Si $T(A)$ est fermé, A est inconsistante.
Démonstration. On suppose que l'arbre $T(A)$ est fermé (tous ses sous-arbres le sont aussi). On démontre par induction sur la hauteur h du nœud n dans $T(A)$ que pour tout sous-arbre de racine n de $T(A)$ l'ensemble $U(n)$ est inconsistant. La hauteur d'une feuille est 0 ; la hauteur d'un nœud α est celle de son fils, plus 1 ; la hauteur d'un nœud β est celle du plus haut de ses fils, plus 1.
- $h = 0$: n est une feuille fermée donc $U(n)$ contient une paire complémentaire de littéraux et $U(n)$ est inconsistant.
- $h > 0$: une règle α ou β a été utilisée pour créer le(s) descendant(s) de n.

  Règle α : n : $\{\alpha\} \cup U_0$ ↓ n' : $\{\alpha_1, \alpha_2\} \cup U_0$

  On a $h(n') < h(n)$ et l'hypothèse inductive s'applique au sous-arbre (fermé) de racine n' ; l'ensemble $U(n')$ est inconsistant. Pour toute interprétation v, il y a une formule $A' \in U(n')$ telle que $v(A') = \mathbf{F}$; trois cas sont possibles :
  1. soit $A' \in U_0 \subseteq U(n)$;
  2. soit $A' = \alpha_1 : v(\alpha_1) = \mathbf{F}$ d'où $v(\alpha) = \mathbf{F}$ (cf. règles α) ;
  3. soit $A' = \alpha_2 : v(\alpha_2) = \mathbf{F}$ d'où $v(\alpha) = \mathbf{F}$.

  Dans les trois cas, il y a une formule dans $U(n)$ que v rend fausse ; v étant quelconque, $U(n)$ est donc inconsistant.

  Règle β : n : $\{\beta\} \cup U_0$; n' : $\{\beta_1\} \cup U_0$; n'' : $\{\beta_2\} \cup U_0$

$h(n') < h(n)$ et $h(n'') < h(n)$; les ensembles $U(n')$ et $U(n'')$ sont tous deux inconsistants. Pour toute interprétation v,

1. soit il y a une formule $A' \in U_0 \subseteq U(n) : v(A') = \mathbf{F}$
2. soit $v(\beta_1) = v(\beta_2) = \mathbf{F}$, d'où $v(\beta) = \mathbf{F}$ (cf. règles β).

Dans les deux cas, il y a une formule dans $U(n)$ que v rend fausse ; v étant quelconque, on en déduit que $U(n)$ est inconsistant.

Ensembles de Hintikka. Les méthodes de la logique sont en général constructive. En particulier, si une formule A est consistante, non seulement tout tableau sémantique pour cette formule doit être ouvert, mais en outre il doit être possible de construire un modèle de A à partir de ce tableau.

Cette construction est possible et même très simple. On va démontrer dans ce paragraphe que toute interprétation vérifiant l'étiquette d'une feuille ouverte vérifie aussi l'étiquette de tout ancêtre de cette feuille, et en particulier de la racine du tableau.

Exemple. Les tableaux de la figure 20 sont ouverts. A chaque feuille ouverte correspond un modèle de la formule-racine. Il peut se faire que cette formule comporte une proposition n'apparaissant pas dans l'étiquette de la feuille ; cela signifie que tout prolongement du modèle spécifié par cette étiquette rend vraie la formule-racine. Le tableau de droite indique que toute interprétation v vérifiant p est un modèle de la formule $p \vee (q \wedge \neg q)$ (et ceci, quelle que soit la valeur attribuée à $v(q)$).

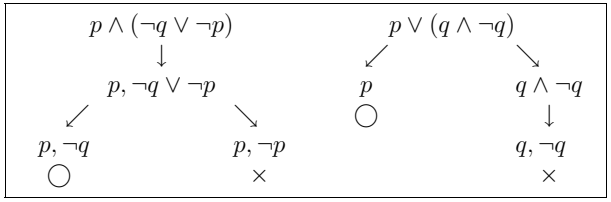


FIG. 20 – Deux tableaux sémantiques ouverts.

Définition : Soit U un ensemble de formules. U est un *ensemble de Hintikka* si les trois conditions suivantes sont satisfaites :

1. Pour tout atome p, $p \notin U$ ou $\neg p \notin U$
2. Si $\alpha \in U$ est une α-formule, alors $\alpha_1 \in U$ et $\alpha_2 \in U$.
3. Si $\beta \in U$ est une β-formule, alors $\beta_1 \in U$ ou $\beta_2 \in U$.

Lemme de la branche ouverte. Soit *b* une branche ouverte d'un tableau complet. L'ensemble $U =_{def} \bigcup_{n \in b} U(n)$ est un ensemble de Hintikka.

Démonstration. On montre que U respecte les trois conditions caractérisant les ensembles de Hintikka.

1. On note ℓ la feuille (ouverte) de *b*. Pour tout littéral m ($m \in \{p, \neg p\}$), $m \in U$ implique $m \in U(\ell)$ car aucune règle ne décompose les littéraux. Or, ℓ est un nœud ouvert, sans paire complémentaire ; on a donc $p \notin U$ ou $\neg p \notin U$.

2. Pour toute α-formule $\alpha \in U$, l'arbre étant complet, la règle α correspondante a dû être utilisée à un certain nœud n. Par construction, $\alpha_1, \alpha_2 \in U(n') \subseteq U$.
3. Pour toute β-formule $\beta \in U$, l'arbre étant complet, la règle β correspondante a dû être utilisée à un certain nœud n. Par construction, $\beta_1 \in U(n')$ et $\beta_2 \in U(n'')$, et $U(n') \subseteq U$ ou $U(n'') \subseteq U$, d'où $\beta_1 \in U$ ou $\beta_2 \in U$.

Lemme de Hintikka. Tout ensemble de Hintikka est consistant.

Démonstration. Soit U un ensemble de Hintikka et soit $\Pi = \{p_1, \ldots, p_m\}$ l'ensemble des atomes apparaissant dans les formules de U. Définissons une interprétation v de U :

$$v(p) = \mathbf{V} \quad \text{si} \quad \neg p \notin U$$
$$v(p) = \mathbf{F} \quad \text{si} \quad \neg p \in U$$

L'interprétation v assigne une et une seule valeur de vérité à chaque atome de Π (car U est un ensemble de Hintikka). Il faut démontrer que pour tout $A \in U$, on a $v(A) = \mathbf{V}$. Cela se fait par induction sur la structure de A :

- A *est un littéral.* Par définition de v on a :
  - Si $A = p$, alors $v(A) = v(p) = \mathbf{V}$.
  - Si $A = \neg p$, alors $v(p) = \mathbf{F}$, d'où $v(A) = \mathbf{V}$.
- A *est une* α*-formule* α. On a $\alpha_1, \alpha_2 \in U$, donc par hypothèse inductive, $v(\alpha_1) = \mathbf{V}$ et $v(\alpha_2) = \mathbf{V}$, d'où $v(\alpha) = \mathbf{V}$ par définition des règles α.
- A *est une* β*-formule* β. On a $\beta_1 \in U$ ou $\beta_2 \in U$, donc par hypothèse inductive, $v(\beta_1) = \mathbf{V}$ ou $v(\beta_2) = \mathbf{V}$, d'où $v(\beta) = \mathbf{V}$ par définition des règles β.

Théorème de complétude. Si $T(A)$ ouvert, alors A est consistante.
Démonstration. Si $T(A)$ est ouvert, il existe une branche ouverte dans $T(A)$. La réunion des ensembles de formules étiquetant les nœuds de cette branche forme un ensemble de Hintikka (donc consistant) ; cet ensemble contient la formule A, qui est donc consistante.

Résumé. La méthode des tableaux sémantiques est un algorithme de décision pour la validité (la consistance, l'inconsistance) dans le calcul des propositions. La formule A est inconsistante si et seulement si $T(A)$ est un tableau fermé ; la formule B est valide si et seulement si $T(\neg B)$ est un tableau fermé ; la formule C est simplement consistante si et seulement si $T(C)$ et $T(\neg C)$ sont des tableaux ouverts.

3.2.4 Exercice sur la méthode des invariants

La preuve d'adéquation et de complétude que nous avons donnée est classique et se trouve dans beaucoup de livres de logique mathématique. Il est quand même intéressant, à titre d'exercice, de développer une vue plus "informatique" de cette double propriété et de sa preuve.

On note d'abord que la méthode des tableaux sémantiques est essentiellement un algorithme itératif, constitué d'une simple boucle (Fig. 18). Si on néglige le marquage par des ronds et des croix, les éléments constitutifs de cette boucle sont l'instruction d'initialisation, la garde de la boucle et le corps de la boucle. L'instruction d'initialisation consiste en la création de la racine du tableau et de son étiquette, composée uniquement de la formule de départ.

La garde exprime l'existence d'un nœud, "feuille provisoire", dont l'étiquette comporte une formule α ou β. Le corps de la boucle consiste en la génération du ou des successeur(s) direct(s) d'une "feuille provisoire" du tableau.

Rien n'empêche donc l'application de la méthode des invariants. En fait, cette application est simple et éclairante ; l'invariant de boucle est la propriété suivante :

Les modèles de la formule de départ sont exactement les interprétations qui sont modèles d'au moins une étiquette de feuille provisoire.

Lorsque le tableau est achevé, le mot "provisoire" disparaît et, clairement, la formule de départ est consistante si et seulement si le tableau comporte au moins une feuille ouverte.

En ce qui concerne la terminaison, le raisonnement fait plus haut reste valable dans le cadre de la méthode des invariants.

3.2.5 La méthode en pratique

Si on présuppose qu'une formule X est inconsistante, on construira d'abord $T(X)$; si on présuppose qu'elle est valide, on construira d'abord $T(\neg X)$. Si le tableau $T(Y)$ est fermé, l'analyse est terminée : on sait que Y est inconsistant et que $\neg Y$ est valide. Si le tableau $T(Z)$ est ouvert, on sait que Z est consistant et que $\neg Z$ n'est pas valide ; il faut construire $T(\neg Z)$ pour en savoir plus.

On tient compte en pratique de trois règles simplificatrices élémentaires :

- Une branche peut être fermée lorsqu'une paire complémentaire de formules (pas seulement de littéraux) apparaît.
- Les formules inchangées ne doivent pas être recopiées d'un nœud à son descendant (économie d'espace).
- Des heuristiques peuvent éventuellement écourter le tableau (par exemple, utiliser les règles α avant les règles β).

3.3 La méthode analytique des séquents

3.3.1 Introduction

La méthode des tableaux sémantiques admet une méthode duale, dite "des séquents". Nous donnons d'abord l'algorithme qui transforme un tableau sémantique en une dérivation de séquent. La figure 21 présente un tableau sémantique et la dérivation de séquent correspondante.

- On opère une symétrie d'axe horizontal, amenant les feuilles en haut et la racine en bas. (La construction directe d'une dérivation de séquent procède donc de bas en haut.)
- Chaque étiquette du nouvel arbre se compose des négations des éléments de l'étiquette correspondante du tableau sémantique ; de plus, chaque étiquette est précédée du symbole $\rightarrow$.
- Les arcs du tableau deviennent des lignes horizontales dans la dérivation de séquent.
- Les symboles $\bigcirc$ et $\times$ sont remplacés respectivement par les symboles **H** et **A**.

Un tableau sémantique et la dérivation de séquent correspondante ne se rapportent donc pas à une même formule. On peut éliminer cette divergence (au moins en apparence) en utilisant la notation signée plutôt que la notation classique pour représenter le tableau (voir figure 22).

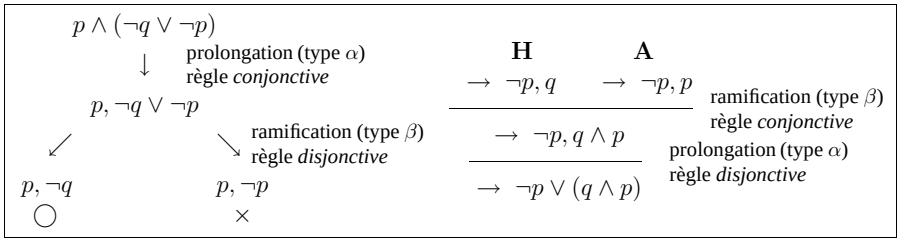


FIG. 21 – Un tableau sémantique et une dérivation de séquent.

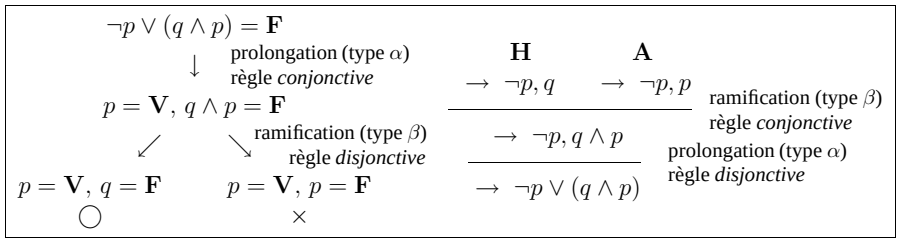


FIG. 22 – Un tableau sémantique signé et une dérivation de séquent.

3.3.2 Interprétation

Fondamentalement, le contenu sémantique d'une dérivation de séquent est le même que celui du tableau correspondant, mais la dualité permet de présenter ce contenu différemment.

- Chaque étiquette d'une dérivation de séquent s'interprète comme un ensemble *disjonctif* de formules.
- Les feuilles correspondent à des *clauses* c'est-à-dire à des disjonctions de littéraux.
- Les feuilles valides sont étiquetées **A** ; ce symbole signifie "Axiome" : vérité universelle. Les feuilles non valides sont étiquetées **H** ; ce symbole signifie "Hypothèse" : énoncé contingent.
- La ligne horizontale s'interprète comme la relation d'équivalence logique : une interprétation rend vraie(s) la (les) *prémisse(s)*, au numérateur, si et seulement si elle rend vraie la *conclusion*, au dénominateur.

On a aussi les définitions et règles suivantes.

- Une clause est un *axiome* si elle comporte une paire complémentaire de littéraux, et une *hypothèse* sinon.
- Les *règles d'inférence* sont de deux types
  - *règles* α (prolongation) :
    $$\frac{\rightarrow U \cup \{\alpha_1, \alpha_2\}}{\rightarrow U \cup \{\alpha\}}$$
  - *règles* β (ramification) :

$$\frac{\rightarrow\ U \cup \{\beta_1\} \qquad \rightarrow\ U \cup \{\beta_2\}}{\rightarrow\ U \cup \{\beta\}}$$

α	α_1	α_2
$A_1 \vee A_2$	A_1	A_2
$\neg(A_1 \wedge A_2)$	$\neg A_1$	$\neg A_2$
$A_1 \Rightarrow A_2$	$\neg A_1$	A_2
$A_1 \Leftarrow A_2$	A_1	$\neg A_2$

β	β_1	β_2
$B_1 \wedge B_2$	B_1	B_2
$\neg(B_1 \vee B_2)$	$\neg B_1$	$\neg B_2$
$\neg(B_1 \Rightarrow B_2)$	B_1	$\neg B_2$
$\neg(B_1 \Leftarrow B_2)$	$\neg B_1$	B_2

FIG. 23 – Les règles de (dé)composition.

Rappelons que les doubles négations sont systématiquement simplifiées et que les connecteurs d'équivalence et de disjonction exclusive sont interdits. On observe (figure 23) que les formules *conjonctives* donnent lieu à une *ramification* (type β) et les formules *disjonctives* à une *prolongation* (type α). C'était le contraire pour les tableaux sémantiques.

Si on lit la dérivation de haut en bas, les règles de décomposition deviennent des règles de composition, ou *règles d'inférence*.

3.3.3 Propriétés de la méthode des séquents

Elles se déduisent immédiatement de celles des tableaux. La méthode de dérivation de séquent est adéquate : toute formule racine d'une dérivation de séquent dont toutes les feuilles sont des axiomes est valide. La méthode de dérivation de séquent est complète : toute formule valide est racine d'une dérivation de séquent dont toutes les feuilles sont des axiomes.

3.3.4 Extension d'écriture

On convient que le séquent
$\rightarrow \neg A, B, \neg C, \neg D, E, F$
peut aussi s'écrire
$A, C, D\ \rightarrow\ B, E, F$.
Ce séquent est vrai pour v si v rend vraie au moins une des formules B, E et F, ou si v rend fausse au moins une des formules A, C et D. La partie de gauche (ici A, C, D) est l'*antécédent*, la partie de droite (ici B, E, F) est le *succédent*. Le séquent est vrai pour v si et seulement si l'implication $(A \wedge C \wedge D) \Rightarrow (B \vee E \vee F)$ est vraie pour v.

La virgule a valeur conjonctive dans l'antécédent et valeur disjonctive dans le succédent. Un antécédent vide correspond à $true$, un succédent vide correspond à $false$, le séquent vide correspond à $false$. Tout séquent dont l'antécédent et le succédent comportent une formule commune est valide.

On peut transférer une formule de l'antécédent au succédent, ou inversement, en changeant sa *polarité*, c'est-à-dire en transformant A en $\neg A$ ou inversement. Le séquent $A, C, D\ \rightarrow\ B, E, F$ est donc logiquement équivalent au séquent $A, \neg B, C\ \rightarrow\ \neg D, E, F$.

Il est commode de récrire les règles en tenant compte des nouvelles notations. A titre d'exemple, si A et B désignent des formules et si U et V désignent des ensembles de formules,

les anciennes règles

$$\frac{\rightarrow\ V, \neg A, B}{\rightarrow\ V, (A \Rightarrow B)} \qquad \frac{\rightarrow\ V, A \qquad \rightarrow\ V, \neg B}{\rightarrow\ V, \neg(A \Rightarrow B)}$$

peuvent être étendues en

$$\frac{U\ \rightarrow\ V, \neg A, B}{U\ \rightarrow\ V, (A \Rightarrow B)} \qquad \frac{U\ \rightarrow\ V, A \qquad U\ \rightarrow\ V, \neg B}{U\ \rightarrow\ V, \neg(A \Rightarrow B)}$$

puis récrites en les nouvelles règles suivantes :

$$\frac{U, A\ \rightarrow\ V, B}{U\ \rightarrow\ V, (A \Rightarrow B)} \qquad \frac{U\ \rightarrow\ V, A \qquad U, B\ \rightarrow\ V}{U, (A \Rightarrow B)\ \rightarrow\ V}$$

3.3.5 Règles réversibles, règles analytiques et synthétiques

La règle d'inférence

$$\frac{U\ \rightarrow\ V, A \qquad U, B\ \rightarrow\ V}{U, (A \Rightarrow B)\ \rightarrow\ V}$$

est *réversible* : la barre horizontale peut s'interpréter comme l'*équivalence* logique. Dans une règle non réversible, la conclusion est conséquence logique des prémisses, mais non l'inverse. Si on pose $U_c = \bigwedge U$ et $V_d = \bigvee V$, la règle ci-dessus exprime que les modèles communs des formules $U_c \Rightarrow (V_d \vee A)$ et $(U_c \wedge B) \Rightarrow V_d$ *sont exactement* les modèles de la formule $(U_c \wedge (A \Rightarrow B)) \Rightarrow V_d$.

Cette règle est aussi *analytique* : toute formule apparaissant en haut apparaît aussi en bas (comme formule ou sous-formule). Les dérivations de séquents peuvent se lire de bas en haut (analyse d'une formule) ou de haut en bas (déduction d'une formule au départ d'axiomes et/ou d'hypothèses).

Il existe aussi des méthodes *synthétiques* de déduction, ne se prêtant pas directement à l'analyse des formules. La méthode synthétique est le plus souvent la seule utilisable en mathématique. On utilise des règles où la barre s'interprète comme la relation (non symétrique) de *conséquence* logique. L'exemple le plus connu de règle synthétique (donc non analytique) et non réversible est sans doute le *Modus ponens* :

$$\frac{U\ \rightarrow\ A \qquad U\ \rightarrow\ (A \Rightarrow B)}{U\ \rightarrow\ B}$$

Un exemple de règle synthétique mais réversible est la règle de *coupure* :

$$\frac{U, A\ \rightarrow\ V \quad U\ \rightarrow\ V, A}{U\ \rightarrow\ V}$$

La formule A et ses sous-formules peuvent ne pas apparaître dans les conclusions. Il est donc difficile de “deviner” des prémisses adéquates au départ des conclusions.[30]

3.3.6 Différences entre implication et séquent

Dans la présentation qui vient d'être faite, la seule différence est que les deux termes d'un séquent sont des ensembles (éventuellement infinis) de formules, tandis que les termes d'une implication sont des formules, finies par nature.

Une autre différence plus subtile apparaît souvent. Les séquents peuvent être utilisés dans des contextes variés, mais le sont habituellement dans un contexte de preuve de validité. Autrement dit, seules des dérivations sans hypothèses sont considérées. Tout séquent apparaissant dans une telle dérivation est valide, et il est alors naturel et fréquent d'introduire cette information dans la définition même de la notion de séquent. Cela signifie qu'un séquent n'est plus un objet du langage (assimilable à une implication) mais devient un objet du métalangage. Dans ce cadre, la signification du séquent

$$A, B, C \ \rightarrow \ D, E, F$$

noté parfois

$$A, B, C \vdash D, E, F$$

est “L'implication $(A \wedge B \wedge C) \Rightarrow (D \vee E \vee F)$ est valide”.

3.3.7 Tableaux signés vs. séquents

Il y a correspondance naturelle entre les tableaux signés et les séquents : les formules apparaissent dans l'antécédent ou dans le succédent d'un séquent selon qu'elles sont assertées positivement ou négativement dans le nœud homologue du tableau (figure 24).

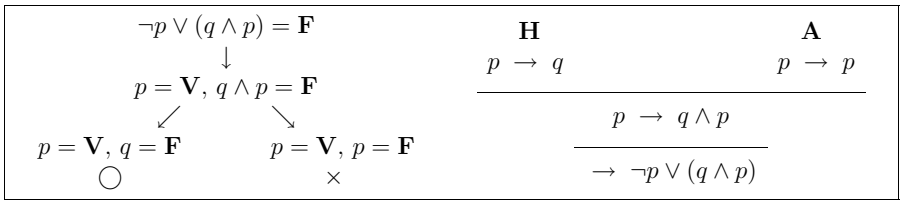


FIG. 24 – Un tableau sémantique signé et une dérivation de séquent.

[30] Les deux règles synthétiques de Modus ponens et de coupure formalisent deux modes de raisonnement omniprésents en mathématique et dans la vie quotidienne. Le Modus ponens traduit la notion même de théorème ou de résultat général : “appliquer” $(A \Rightarrow B)$, c'est déduire B dans le cas où A est connu. La règle de coupure formalise le raisonnement par cas : si on infère V de U quand A est vrai, et aussi quand A est faux, on infère V de U en toute généralité.

3.4 Le raisonnement automatique

3.4.1 Introduction

La logique formelle permet, comme on vient de le voir tout au long de ce chapitre, de transformer le raisonnement en un objet mathématique, susceptible d'être traité, et même construit, par un ordinateur. Nous avons vu, au paragraphe 2.3.3, qu'un texte court mais relativement dense, présentant un raisonnement, pouvait être transformé en formules et ainsi devenir accessible à une analyse fiable et automatique.[31] Un célèbre ouvrage de science-fiction[32] évoque ainsi l'analyse formelle d'un très compliqué et volumineux document diplomatique … aboutissant à la conclusion que ce document était sémantiquement vide et que ses auteurs méritaient, étymologiquement du moins, leur statut de diplomate. Plus concrètement, un livre d'analyse mathématique a été entièrement vérifié par ordinateur dans le cadre d'un projet d'intelligence artificielle. Peut-on réellement espérer ramener ainsi le raisonnement au calcul, dans le but de l'automatiser ?

3.4.2 Digression : Leibniz et le raisonnement automatisable

Leibniz avait imaginé un “ratiocinator universalis”, sorte de machine abstraite capable de raisonner et, dans une certaine mesure, de trancher des débats épineux. Il croyait possible de représenter les idées et le raisonnement dans un langage symbolique pourvu d'une syntaxe et d'une sémantique précises. Dans une discussion, il devenait théoriquement possible de remplacer un débat d'idées animé par une séance de froid calcul dont l'issue serait inconditionnellement admise par les protagonistes du débat. Boole et surtout Frege ont créé le langage symbolique rêvé par Leibniz, et la logique prédicative est parfaitement apte à la représentation de tout type de raisonnement.[33] Il y a cependant loin entre la capacité de représenter un problème et la capacité de le résoudre ; dans cette section, nous évoquons quelques aspects fragmentaires mais importants de cette question.

3.4.3 Automatiser la logique

Cette question est comme beaucoup d'autres : il est plus facile de l'examiner dans le cadre propositionnel que dans le cadre prédicatif, et certaines conclusions établies dans le cadre propositionnel s'étendront au cadre prédicatif. Cela étant admis, on pourrait croire qu'il n'y a pas de question. La logique propositionnelle est en effet automatisable, puisque nous l'avons effectivement automatisée. La méthode des tables de vérité et celle des tableaux sémantiques par exemple, permettent d'analyser tout raisonnement propositionnel, si complexe soit-il. Nous allons voir maintenant que la portée *pratique* de ces méthodes est sévèrement limitée par un problème de dimension. En dépit de la puissance qu'ils ont atteinte actuellement (et qui continuera à croître de longues années encore), les ordinateurs ne sont pas capables d'analyser, en toute généralité, un problème logique d'une certaine taille. Considérons par exemple le problème classique consistant à déterminer si un ensemble de formules est consistant ou pas.

[31] L'informaticien dirait : fiable parce que complètement automatique…

[32] *Foundation*, d'Isaac Asimov.

[33] Des logiques spéciales ont été et continuent à être introduites pour modéliser plus commodément certains aspects de la connaissance mais, fondamentalement, la logique prédicative peut prétendre à l'universalité.

Si nous utilisons les tables de vérité, et si l'ensemble en question comporte des occurrences de n propositions élémentaires distinctes, il suffit de construire une table de vérité ... qui comportera 2^n lignes. Si n vaut 30, la table comportera plus d'un milliard de lignes ; si n vaut 100, ce qui n'a rien d'irréaliste, le problème est, sauf cas particulier, définitivement hors d'atteinte de tout ordinateur présent ou à venir. Observons au passage que *multiplier* par 1 000 les performances d'un ordinateur ne permet que d'*ajouter* 10 nouvelles variables propositionnelles à notre lexique.

Il existe a priori deux moyens de contourner cet écueil. D'une part, il est possible de développer des procédures de décision plus rapides que la méthode des tables de vérité et, d'autre part, on peut essayer d'isoler certains types de formules et d'ensembles de formules pour lesquels le problème de la consistance pourrait se résoudre plus rapidement. Nous avons déjà adopté la première approche : la méthode des tableaux sémantiques est souvent — mais pas toujours — nettement plus efficace que celle des tables de vérité. Une analyse plus fine montrerait quand même que cette méthode et, à des degrés divers, toutes les méthodes connues actuellement, restent fondamentalement trop lentes. Plus précisément, dans beaucoup de cas, les performances se dégradent très vite dès que la dimension du problème augmente. Une théorie récemment développée[34] laisse peu de chances de progrès significatifs dans cette voie.

La seconde approche est plus prometteuse ; nous la développons dans la suite de ce chapitre.

3.4.4 Cubes, clauses et formes normales

Considérons la table de vérité de la figure 25, se rapportant à une formule inconnue X, dépendant des trois variables propositionnelles p, q et r. Peut-on reconstituer la formule sur base de la table ? Oui, à une équivalence logique près. La table indique que la formule est vraie dans quatre cas, correspondant aux lignes 1, 3, 5 et 6. A chaque cas correspond un *cube*, c'est-à-dire une conjonction de littéraux. Par exemple, le cube correspondant à la ligne 6 est $(\neg p \wedge q \wedge \neg r)$, ce qui s'interprète en $v(p) = \mathbf{F}$, $v(q) = \mathbf{V}$ et $v(r) = \mathbf{F}$.

p	q	r	$X(p,q,r)$
V	**V**	**V**	**V**
V	**V**	**F**	**F**
V	**F**	**V**	**V**
V	**F**	**F**	**F**
F	**V**	**V**	**V**
F	**V**	**F**	**V**
F	**F**	**V**	**F**
F	**F**	**F**	**F**

FIG. 25 – Table de vérité d'une formule inconnue

La formule X est donc (logiquement équivalente à) la formule

$$(p \wedge q \wedge r) \vee (p \wedge \neg q \wedge r) \vee (\neg p \wedge q \wedge r) \vee (\neg p \wedge q \wedge \neg r)\,.$$

[34] Et notamment le théorème de la NP-complétude du problème de la consistance, démontré par Cook en 1970.

Cette formule peut s'écrire différemment, et notamment

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

ou encore

$$(p \Rightarrow r) \wedge (\neg p \Rightarrow q)\,.$$

Le point important est que toute formule, puisqu'elle admet une table de vérité, est équivalente à une disjonction de cubes, ce que l'on appelle aussi une *forme disjonctive normale*.

Observons aussi que la négation d'une disjonction de cubes est une conjonction de clauses, ce que l'on appelle aussi une *forme conjonctive normale*. On obtient aisément la forme conjonctive normale d'une formule à partir de la forme disjonctive normale de la négation de cette formule ; on peut aussi l'obtenir directement à partir de la table de vérité, en considérant les lignes pour lesquelles la formule est fausse. Par exemple, la formule $X(p,q,r)$ est fausse dans quatre cas ; elle peut donc s'écrire

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

ou encore

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

ce qui est une forme conjonctive normale, c'est-à-dire une conjonction de clauses ; la formule peut se simplifier en

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

qui est encore une forme conjonctive normale, appelée aussi *forme clausale*.

Le principe de la déduction affirme que la formule A est conséquence logique de l'ensemble E si et seulement si l'ensemble $E \cup \{\neg A\}$ est inconsistant. Chaque formule de ce dernier ensemble est logiquement équivalente à une conjonction de clauses ; si $\mathcal{C}$ est l'ensemble de toutes ces clauses, on peut dire que A est conséquence logique de E si et seulement si $\mathcal{C}$ est inconsistant. Le problème fondamental de la logique se résume donc à celui de déterminer si un ensemble de clauses est inconsistant ou non.

Remarque. Construire la table de vérité d'une formule donnée n'est généralement pas le moyen le plus rapide d'obtenir une forme normale disjonctive ou conjonctive logiquement équivalente à cette formule.

3.4.5 Clauses de Horn et ensembles de Horn

Nous venons de rappeler que, pour les problèmes d'une certaine taille, l'approche automatique était très aléatoire. C'est étonnant, dans la mesure où l'être humain moyen est capable d'effectuer des raisonnements logiques de grande taille avec une certaine efficacité. Une raison à cela est l'aptitude, typique de l'être humain, à s'adapter aux circonstances et à remplacer une méthode générale par une approche plus spécifique et plus rapide, du moins dans le cas particulier considéré. Une autre raison, plus pertinente ici, est que bien souvent les formules de l'ensemble E, qui constituent la "base de connaissance" à partir de laquelle on va déduire la formule A, ont une forme très particulière, qui se retrouve aussi dans les énoncés mathématiques, à savoir

$$(p_1 \wedge \cdots \wedge p_n) \Rightarrow q\,.$$

Les p_i et q sont des propositions élémentaires.[35] La formule exprime simplement que toute interprétation rendant vraies les propositions $p_1, \ldots, p_n$ rend vraie aussi la conclusion q.[36]

On voit immédiatement que cette formule est une clause, que l'on peut récrire en

$$\neg p_1 \vee \cdots \vee \neg p_n \vee q\,.$$

De même, la conclusion A est souvent une simple proposition, ou une conjonction de propositions, donc de litéraux positifs. Sa négation est une disjonction de littéraux négatifs.

On appelle *clause de Horn* toute clause contenant au plus un littéral positif. Une clause de Horn est *définie* si elle comporte exactement un littéral positif ; elle est *négative* si elle n'en comporte pas. Un *ensemble de Horn* est un ensemble de clauses de Horn. Cette notion est extrêmement importante parce que, dans le cas particulier des ensembles de Horn, le problème de la consistance peut se résoudre de manière efficace, en toute généralité.

3.4.6 L'algorithme de résolution unitaire

Une *clause unitaire* est une clause comportant un seul littéral. Une clause unitaire peut être *positive* ou *négative*. Dans la suite, sauf mention explicite du contraire, une clause unitaire est une clause unitaire positive, réduite à une proposition élémentaire. L'algorithme de *résolution unitaire* (figure 26) est un moyen simple de déterminer si un ensemble de Horn est consistant ou inconsistant.

$\{S := S_0\}$
Tant que $\Box \notin S$ faire
choisir p et c tels que
p est une clause unitaire positive de S,
c est une clause de S contenant $\neg p$;
$r := c \setminus \{\neg p\}$;
$S := (S \setminus \{c\}) \cup \{r\}$.

FIG. 26 – Résolution unitaire

Le principe de cet algorithme est très simple. Il consiste à supprimer, dans les clauses d'un ensemble S de clauses de Horn, tous les littéraux négatifs dont la proposition sous-jacente apparaît comme clause unitaire, jusqu'à ce qu'une clause soit devenue vide, ou que plus aucune suppression ne soit possible. Dans le premier cas, on conclut à l'inconsistance, dans le second, à la consistance. La notation ":=" signifie "devient" ; exécuter l'instruction $x := x + y$ signifie que la nouvelle valeur de x est l'ancienne valeur de $x + y$. Si c est une clause contenant $\neg p$, $c \setminus \{\neg p\}$ est la clause obtenue en supprimant $\neg p$. La *clause vide*, qui ne contient aucun littéral, est représentée par le symbole $\Box$. Une clause est vraie si au moins un de ses littéraux est vrai ; la clause vide est donc logiquement équivalente à $false$. Si c est une clause de l'ensemble S, $S \setminus \{c\}$ est l'ensemble obtenu en enlevant c de S. Si r est une clause, $S \cup \{r\}$ est l'ensemble obtenu en ajoutant r à S.

La figure 27 donne un exemple d'exécution de l'algorithme, permettant de montrer que l'ensemble

$$S = \{p \vee \neg r \vee \neg t\,,\ q\,,\ r\,,\ t \vee \neg p \vee \neg r\,,\ t \vee \neg q\,,\ \neg p \vee \neg q \vee \neg r\}$$

est inconsistant.

1.	$p \vee \neg r \vee \neg t$	$\underline{q}$	r	$t \vee \neg p \vee \neg r$	$t \vee \underline{\neg q}$	$\neg p \vee \neg q \vee \neg r$
2.	$p \vee \underline{\neg r} \vee \neg t$	q	$\underline{r}$	$t \vee \neg p \vee \neg r$	t	$\neg p \vee \neg q \vee \neg r$
3.	$p \vee \neg t$	$\underline{q}$	r	$t \vee \neg p \vee \neg r$	t	$\neg p \vee \underline{\neg q} \vee \neg r$
4.	$p \vee \neg t$	q	$\underline{r}$	$t \vee \neg p \vee \neg r$	t	$\neg p \vee \underline{\neg r}$
5.	$p \vee \underline{\neg t}$	q	r	$t \vee \neg p \vee \neg r$	$\underline{t}$	$\neg p$
6.	$\underline{p}$	q	r	$t \vee \neg p \vee \neg r$	t	$\underline{\neg p}$
7.	p	q	r	$t \vee \neg p \vee \neg r$	t	$\Box$

FIG. 27 – Résolution unitaire : un exemple positif

La figure 28 donne un second exemple d'exécution de l'algorithme, permettant de montrer que l'ensemble

$$S = \{p \vee \neg r \vee \neg t\,,\ q\,,\ s\,,\ t \vee \neg p \vee \neg r\,,\ t \vee \neg q \vee \neg s\,,\ \neg p \vee \neg q \vee \neg r\}$$

est consistant.

1.	$p \vee \neg r \vee \neg t$	$\underline{q}$	s	$t \vee \neg p \vee \neg r$	$t \vee \underline{\neg q} \vee \neg s$	$\neg p \vee \neg q \vee \neg r$
2.	$p \vee \neg r \vee \neg t$	q	$\underline{s}$	$t \vee \neg p \vee \neg r$	$t \vee \underline{\neg s}$	$\neg p \vee \neg q \vee \neg r$
3.	$p \vee \neg r \vee \neg t$	$\underline{q}$	s	$t \vee \neg p \vee \neg r$	t	$\neg p \vee \underline{\neg q} \vee \neg r$
4.	$p \vee \neg r \vee \underline{\neg t}$	q	s	$t \vee \neg p \vee \neg r$	$\underline{t}$	$\neg p \vee \neg r$
5.	$p \vee \neg r$	q	s	$t \vee \neg p \vee \neg r$	t	$\neg p \vee \neg r$

FIG. 28 – Résolution unitaire : un exemple négatif

La première propriété de l'algorithme de résolution unitaire est d'être efficace. A chaque étape, un littéral est enlevé donc le nombre d'étapes ne peut dépasser le nombre total de littéraux. A chaque étape, l'ensemble S change (un littéral est supprimé). Comme d'habitude, le point crucial de l'analyse de l'algorithme consiste à déterminer ce qui ne change pas. Comme précédemment, c'est l'ensemble des modèles de S qui ne change pas. En effet, si la clause unitaire p se trouve dans S, seules les interprétations rendant p vrai peuvent être des modèles. Soit I une telle interprétation ; quelle que soit la clause c contenant $\neg p$, on a $I(c) = I(c \setminus \{\neg p\})$. On a donc, après chaque étape, $\mathcal{M}(S) = \mathcal{M}(S_0)$. En particulier, après la dernière étape, on a $\mathcal{M}(S_f) = \mathcal{M}(S_0)$, si S_f désigne l'état final de l'ensemble S. Cela prouve en particulier

[35] On peut considérer que cette formule représente un théorème mathématique, dont les p_i sont les hypothèses et q la thèse.

[36] On observera que cette formule, même si elle représente un théorème de mathématique, n'est pas valide. La raison en est que dans le cadre de la logique propositionnelle, les propositions élémentaires ne sont pas analysées.

que S_0 (l'ensemble de départ, celui qui nous intéresse) est consistant si et seulement si S_f est consistant. Il se fait que déterminer si S_f est consistant est immédiat. En effet, il n'y a que deux possibilités :

- L'ensemble S_f contient la clause vide, qui est inconsistante, donc S_f est inconsistant.
- L'ensemble S_f ne contient pas la clause vide. Soit I l'interprétation qui rend vraies toutes les clauses unitaires (positives) de S_f et fausses toutes les autres propositions. Cette interprétation rend vraies toutes les clauses unitaires, et aussi toutes les clauses non unitaires, puisque ces dernières contiennent au moins un littéral négatif dont la proposition sous-jacente est fausse par définition de I (car cette proposition n'est pas une clause unitaire, sinon elle donnerait lieu à une étape supplémentaire).

On appelle *base de connaissance* un ensemble de clauses de Horn positives ; on appelle *question* une conjonction de propositions. L'algorithme de résolution unitaire permet de déterminer si une question A est conséquence logique d'une base de connaissance H ; ce sera le cas si l'ensemble $H \cup \{\neg A\}$ est reconnu inconsistant.

Remarque. On peut aussi tester l'ensemble H seul ; il est nécessairement consistant puisqu'il ne comporte que des clauses de Horn positives.[37] La détermination de H_f permet d'obtenir l'ensemble de toutes les propositions qui sont conséquences logiques de H ; ce sont les propositions qui apparaissent comme clauses unitaires dans H_f. L'interprétation qui rend vraies ces propositions et seulement celles-là est le *modèle canonique*, ou *modèle minimal* de H. Le modèle minimal de l'ensemble traité à la figure 28 est donc l'interprétation qui rend vraies les propositions q, s et t et seulement celles-là.

Remarque. On représente souvent les exécutions de l'algorithme de résolution unitaire sous forme arborescente ; la représentation correspondant à l'exécution de la figure 27 se trouve à la figure 29. L'arborescence s'appelle *arbre de dérivation*, ou *arbre de réfutation* dans le cas particulier où on dérive la clause vide.

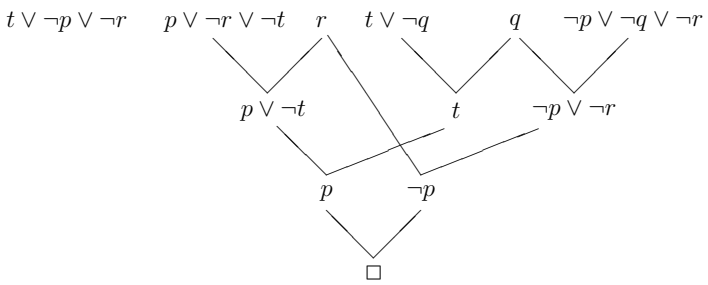


FIG. 29 – Arbre de réfutation unitaire pour l'ensemble S

[37] L'interprétation qui rend vraies toutes les propositions est donc un modèle de H.

3.4.7 La programmation logique propositionnelle

Le problème de la programmation logique propositionnelle consiste à déterminer si une proposition est ou n'est pas conséquence logique d'un ensemble de clauses de Horn définies, appelé *base de connaissance* ou *programme logique*. Nous venons de voir que l'algorithme de résolution unitaire est une solution générale et raisonnablement efficace pour ce problème. Elle n'est cependant pas optimale en pratique. La raison en est que, dans la plupart des cas, la base de connaissance est énorme, voire infinie, et que la plupart des clauses qu'elle contient n'ont rien à voir avec la question particulière à traiter. L'algorithme de résolution unitaire n'accorde aucun rôle particulier à la question traitée, dont la négation est simplement ajoutée à la base de connaissance. L'*algorithme de résolution d'entrée* est une variante de l'algorithme de résolution unitaire, dans laquelle la négation de la question joue un rôle privilégié. Cette variante est représentée à la figure 30, où L désigne un programme logique.

$\{G = G_0\}$
Tant que $G \neq \square$ faire
choisir p et c tels que
$\neg p \in G$,
$c \in L$ et
$p \in c$;
$G := (G \setminus \{\neg p\}) \vee (c \setminus \{p\})$.

FIG. 30 – Résolution d'entrée

Cet algorithme utilise une variable G, appelée le *but*, dont la valeur est toujours une clause de Horn négative ; initialement, le but est la négation de la question. A chaque étape, le but est transformé selon la règle suivante : un littéral $\neg p$ du but est remplacé par $(c \setminus \{p\})$, où c est une clause de la base de connaissance, dont le littéral positif est p. On pourrait démontrer que l'algorithme de résolution d'entrée est équivalent à l'algorithme de résolution unitaire. A titre d'exemple, nous montrons à la figure 31 que la proposition p est bien conséquence logique du programme logique

$$L = \{\, t \vee \neg p \vee \neg r\,,\ p \vee \neg r \vee \neg t\,,\ r\,,\ t \vee \neg q\,,\ q\,\}\,.$$

On voit que, dans un arbre de réfutation d'entrée, il existe une branche principale unissant le but initial (ici, $\neg p$) à la clause vide. Les branches auxiliaires sont de longueur 1 et unissent une clause d'entrée (d'où le nom de l'algorithme) à un but intermédiaire.

3.4.8 Prolog propositionnel

L'algorithme de Prolog est une version concrète de l'algorithme de résolution d'entrée. Les clauses sont représentées par des listes de littéraux et le programme logique L est une liste de clauses. Les choix de p et c sont imposés par une stratégie très simple ; p est nécessairement la proposition correspondant au premier littéral du but et c est la première clause de L convenable. En effet, le système Prolog essaiera, dans l'ordre où elles se présentent dans la liste L, toutes les clauses dont la tête est p. Cela peut poser un problème si l'ordre des clauses ou des littéraux

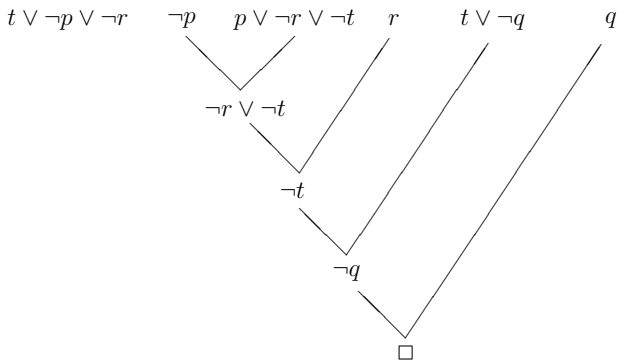


FIG. 31 – Arbre de réfutation d'entrée pour L et p

dans une clause est mal choisi. A titre d'exemple, voici la version Prolog du programme logique L donné plus haut :

```
t :- p,r.
p :- r,t.
r.
t :- q.
q.
```

On voit que la virgule a valeur conjonctive et que le symbole "`:-`" représente le connecteur $\Leftarrow$ (inverse de l'implication). L'ordre n'est pas adéquat ici car la clause inutile "`t :- p,r.`" court-circuite la clause utile "`t :- q.`". Si on omet la clause inutile, le système Prolog détecte que la proposition p est conséquence logique du programme logique L. Nous reviendrons sur le système Prolog dans le cadre prédicatif, qui permet des applications plus intéressantes.

3.5 Quelques exercices

3.5.1 Argumentation

Le récit de la création du monde. Au paragraphe 2.3.3, nous avons formalisé un argument ; les techniques présentées dans ce chapitre permettent de déterminer si cet argument est correct ou non ou, plus précisément, si l'argument formel correspondant est correct. Un argument formel se compose d'un ensemble (fini) de prémisses $E = \{P_1, \ldots, P_n\}$ et d'une conclusion C ; il est dit *correct* si la conclusion est conséquence logique des prémisses, ce qui s'écrit $E \models C$; cela a lieu si et seulement si l'implication $(P_1 \wedge \cdots \wedge P_n) \Rightarrow C$ est valide. L'argument formel introduit au paragraphe 2.3.3 conduit donc à la question

$$\{E \Rightarrow \neg Q\,,\ \neg Q \Rightarrow \neg A\,,\ D \vee A\} \overset{?}{\models} \neg E \vee D\,.$$

La méthode des tables de vérité est peu intéressante ici car le lexique utilisé comporte quatre propositions ; la table de vérité aurait donc seize lignes. La méthode des tableaux sémantiques peut s'appliquer ; si nous croyons que l'argument est correct, la racine de notre tableau sera la négation de l'implication associée à cet argument. La figure 32 représente ce tableau.

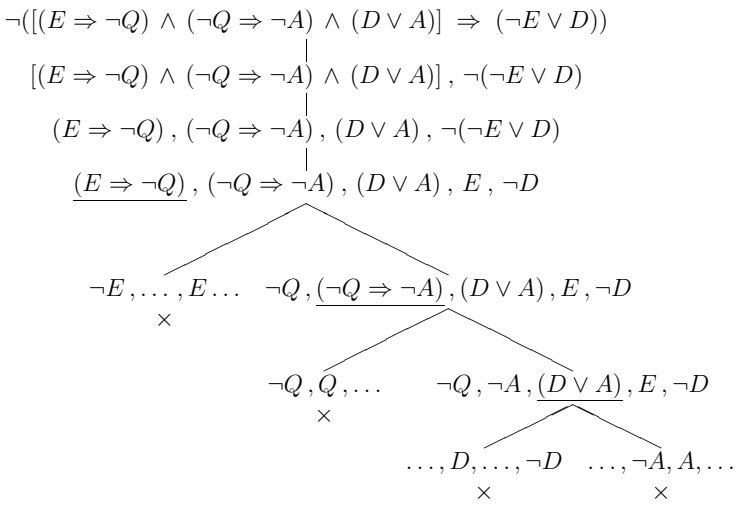


FIG. 32 – Tableau sémantique, argument "biblique"

Ce tableau est fermé, donc l'argument est correct. On notera cependant que cette construction n'est que marginalement moins fastidieuse que celle d'une table de vérité à seize lignes. Il serait souhaitable de disposer de techniques plus expéditives, non seulement pour gagner du temps, mais aussi pour limiter le risque d'erreur.[38] En relisant attentivement la justification de la méthode des tableaux sémantiques, on peut observer que l'étiquette d'un nœud peut être remplacée par une autre, pour peu que les deux étiquettes admettent exactement les mêmes modèles.[39]

La figure 33 donne un tableau sémantique exploitant ce principe. Les simplifications successives se basent sur les faits suivants :

- Les ensembles $\{E \Rightarrow \neg Q\,,\ E\}$ et $\{\neg Q\,,\ E\}$ sont logiquement équivalents ;
- Les ensembles $\{D \vee A\,,\ \neg D\}$ et $\{A\,,\ \neg D\}$ sont logiquement équivalents ;
- Les ensembles $\{\neg Q \Rightarrow \neg A\,,\ A\}$ et $\{Q\,,\ A\}$ sont logiquement équivalents.

Chacune de ces simplifications a permis l'économie d'un branchement.[40]

[38] On dit parfois que le taux d'erreur d'un développement formel (non vérifié par ordinateur) est proportionnel au *carré* de la taille de ce développement…

[39] De plus, pour éviter le risque de non-terminaison, la nouvelle étiquette ne pourra être plus complexe que l'ancienne.

[40] Une autre manière de justifier ces simplifications est d'observer que chaque couple simplifiable comporte une formule disjonctive et un littéral, et que la décomposition de la formule disjonctive donne lieu à une branche comportant le littéral opposé et à une branche comportant le couple simplifié. Le "raccourci" proposé consiste à faire l'économie de la première branche, inutile puisque fermable immédiatement.

$$\neg([(E \Rightarrow \neg Q) \wedge (\neg Q \Rightarrow \neg A) \wedge (D \vee A)] \Rightarrow (\neg E \vee D))$$
$$\downarrow$$
$$[(E \Rightarrow \neg Q) \wedge (\neg Q \Rightarrow \neg A) \wedge (D \vee A)]\,,\ \neg(\neg E \vee D)$$
$$\downarrow$$
$$(E \Rightarrow \neg Q)\,,\ (\neg Q \Rightarrow \neg A)\,,\ (D \vee A)\,,\ \neg(\neg E \vee D)$$
$$\downarrow$$
$$\underline{(E \Rightarrow \neg Q)}\,,\ (\neg Q \Rightarrow \neg A)\,,\ (D \vee A)\,,\ \underline{E}\,,\ \neg D$$
$$\downarrow$$
$$\neg Q\,,\ (\neg Q \Rightarrow \neg A)\,,\ \underline{(D \vee A)}\,,\ E\,,\ \underline{\neg D}$$
$$\downarrow$$
$$\neg Q\,,\ \underline{(\neg Q \Rightarrow \neg A)}\,,\ \underline{A}\,,\ E\,,\ \neg D$$
$$\downarrow$$
$$\neg Q\,,\ Q\,,\ A\,,\ E\,,\ \neg D$$
$$\times$$

FIG. 33 – Tableau sémantique simplifié

On peut aussi envisager l'utilisation de la théorie de Horn, pour tester l'inconsistance de l'ensemble composé des prémisses et de la négation de la conclusion :

$$\{E \Rightarrow \neg Q\,,\ \neg Q \Rightarrow \neg A\,,\ D \vee A\,,\ \neg(\neg E \vee D)\}\,,$$

qui se récrit en

$$\{\,\neg E \vee \neg Q\,,\ Q \vee \neg A\,,\ D \vee A\,,\ E\,,\ \neg D\,\}\,.$$

L'une des clauses de cet ensemble comporte deux littéraux positifs, ce qui est incompatible avec l'utilisation de la théorie de Horn. Dans le cas présent, on remédie à ce problème par obversion.[41] On introduit donc la proposition $\tilde{D}$, définie comme logiquement équivalente à la formule $\neg D$. L'ensemble devient

$$\{\,\neg E \vee \neg Q\,,\ Q \vee \neg A\,,\ \neg\tilde{D} \vee A\,,\ E\,,\ \tilde{D}\,\}\,.$$

Cet ensemble de Horn est bien inconsistant, comme le montre le développement de la figure 34.

Croire aux fantômes ? C'est ce que nous suggère le raisonnement ci-dessous, qu'il est prudent d'analyser ...

> *Si on considère que les gens qui étudient les perceptions extra-sensorielles sont honnêtes, alors il faut admettre l'existence de telles perceptions. De plus, si l'on met à l'épreuve l'existence des perceptions extra-sensorielles, on se doit de considérer sérieusement la doctrine de la clairvoyance. Admettre l'existence des perceptions extra-sensorielles doit nous pousser à mettre celles-ci à l'épreuve et à les expliquer.*

[41] L'*obversion* consiste à introduire ou à supprimer une négation dans une phrase sans en changer le sens. Par exemple, la phrase "Tout ensemble contenant une paire complémentaire de littéraux est inconsistant" devient par obversion "Aucun ensemble contenant une paire complémentaire de littéraux n'est consistant".

1.	$\neg E \vee \neg Q$	$Q \vee \neg A$	$\neg\tilde{D} \vee A$	$\underline{E}$	$\tilde{D}$
2.	$\neg Q$	$Q \vee \neg A$	$\neg\tilde{D} \vee A$	E	$\underline{\tilde{D}}$
3.	$\neg Q$	$Q \vee \neg A$	$\underline{A}$	E	$\tilde{D}$
3.	$\neg Q$	$\underline{Q}$	A	E	$\tilde{D}$
4.	$\Box$	$\overline{Q}$	A	E	$\tilde{D}$

FIG. 34 – Résolution unitaire : argument "biblique"

> *La doctrine de la clairvoyance doit être considérée sérieusement si on est prêt à considérer sérieusement les phénomènes occultes. Et si on est prêt à considérer sérieusement ces phénomènes, nous devrons respecter les médiums. Aussi, si nous respectons ces gens, nous devrons aussi prendre au sérieux leur prétendue aptitude à communiquer avec les morts. Enfin, si nous devons prendre au sérieux cette aptitude à communiquer avec les morts, on ne peut que croire aux fantômes.*
>
> *Considérer que les gens qui étudient les perceptions extra-sensorielles sont honnêtes nous oblige donc à croire aux fantômes.*

On utilise le lexique suivant :

hon	on considère que les gens qui étudient les perceptions extra-sensorielles sont honnêtes ;
adm	on admet l'existence des perceptions extra-sensorielles ;
epr	on met à l'*épr*euve l'existence des perceptions extra-sensorielles ;
cla	on considère sérieusement la doctrine de la clairvoyance ;
exp	on cherche à expliquer les perceptions extra-sensorielles ;
occ	on considère sérieusement les phénomènes occultes ;
med	on respecte les *méd*iums ;
com	on prend au sérieux l'aptitude des médiums à communiquer avec les morts ;
fan	on croit aux fantômes.

Les prémisses sont, pour le premier paragraphe,

$$hon \Rightarrow adm\,,\ epr \Rightarrow cla\,,\ adm \Rightarrow (epr \wedge exp)\,;$$

celles du second paragraphe sont

$$cla \Leftarrow occ\,,\ occ \Rightarrow med\,,\ med \Rightarrow com\,,\ com \Rightarrow fan\,.$$

La conclusion (dernier paragraphe) est

$$hon \Rightarrow fan\,.$$

Les procédés utilisés pour résoudre le problème du récit biblique s'appliquent également à ce problème ; nous considérons ici seulement la résolution unitaire. La prémisse $adm \Rightarrow (epr \wedge exp)$ n'est pas une clause, mais est logiquement équivalente à la conjonction des deux

clauses $adm \Rightarrow epr$ et $adm \Rightarrow exp$; de même, la négation de la conclusion $hon \Rightarrow fan$ n'est pas une clause, mais est logiquement équivalente à la conjonction des deux clauses hon et $\neg fan$. On obtient ainsi le développement de la figure 35, dans laquelle les clauses (de Horn) ont gardé leur forme implicative.

1.	$hon \Rightarrow adm\,,\ epr \Rightarrow cla\,,\ adm \Rightarrow epr\,,\ adm \Rightarrow exp$ $cla \Leftarrow occ\,,\ occ \Rightarrow med\,,\ med \Rightarrow com\,,\ com \Rightarrow fan\,,\ \underline{hon}\,,\ \neg fan$
2.	$\underline{adm}\,,\ epr \Rightarrow cla\,,\ adm \Rightarrow epr\,,\ adm \Rightarrow exp$ $cla \Leftarrow occ\,,\ occ \Rightarrow med\,,\ med \Rightarrow com\,,\ com \Rightarrow fan\,,\ hon\,,\ \neg fan$
3.	$adm\,,\ epr \Rightarrow cla\,,\ \underline{epr}\,,\ exp$ $cla \Leftarrow occ\,,\ occ \Rightarrow med\,,\ med \Rightarrow com\,,\ com \Rightarrow fan\,,\ hon\,,\ \neg fan$
4.	$adm\,,\ cla\,,\ epr\,,\ exp$ $cla \Leftarrow occ\,,\ occ \Rightarrow med\,,\ med \Rightarrow com\,,\ com \Rightarrow fan\,,\ hon\,,\ \neg fan$

FIG. 35 – Résolution unitaire : croire au fantômes ?

On voit immédiatement que l'obtention de la nouvelle clause unitaire cla ne permet pas de progresser, car l'unique autre occurrence de cla est positive, dans la prémisse $cla \Leftarrow occ$. Cependant, si la prémisse

> *La doctrine de la clairvoyance doit être considérée sérieusement si on est prêt à considérer sérieusement les phénomènes occultes.*

était remplacée par la prémisse

> *La doctrine de la clairvoyance doit être considérée sérieusement* **seulement** *si on est prêt à considérer sérieusement les phénomènes occultes.*

ou encore, ce qui revient au même, par la prémisse

> *Si la doctrine de la clairvoyance est considérée sérieusement, alors on doit aussi considérer sérieusement les phénomènes occultes.*

la clause $cla \Leftarrow occ$ serait remplacée par la clause $cla \Rightarrow occ$; cela rendrait l'argument correct, comme le montre le développement de la figure 36. On voit toute l'importance qu'un seul mot peut avoir dans un texte …

3.5.2 Analyse de formules

Soient A, B, X, Y des formules. On suppose $A \models B$. Dans les quatre cas suivants, peut-on affirmer $C_i \models D_i$ et/ou $D_i \models C_i$?

1. $C_1 =_{def} X \Rightarrow (A \Rightarrow Y)$ et $D_1 =_{def} X \Rightarrow (B \Rightarrow Y)$;
2. $C_2 =_{def} (X \Rightarrow A) \vee Y$ et $D_2 =_{def} (X \Rightarrow B) \vee Y$;
3. $C_3 =_{def} (X \equiv A) \Rightarrow Y$ et $D_3 =_{def} (X \equiv B) \Rightarrow Y$;
4. $C_4 =_{def} X \equiv ((A \Rightarrow (B \vee A)) \Rightarrow Y)$ et $D_4 =_{def} X \equiv ((B \Rightarrow (A \vee B)) \Rightarrow Y)$.

Comme toujours, la méthode des tables de vérité peut être utilisée. En principe, puisque les formules C et D dépendent des quatre formules A, B, X, Y, seize lignes sont nécessaires.

1.	$hon \Rightarrow adm\,,\ epr \Rightarrow cla\,,\ adm \Rightarrow epr\,,\ adm \Rightarrow exp$ $cla \Rightarrow occ\,,\ occ \Rightarrow med\,,\ med \Rightarrow com\,,\ com \Rightarrow fan\,,\ \underline{hon}\,,\ \neg fan$
2.	$\underline{adm}\,,\ epr \Rightarrow cla\,,\ adm \Rightarrow epr\,,\ adm \Rightarrow exp$ $cla \Rightarrow occ\,,\ occ \Rightarrow med\,,\ med \Rightarrow com\,,\ com \Rightarrow fan\,,\ hon\,,\ \neg fan$
3.	$adm\,,\ epr \Rightarrow cla\,,\ \underline{epr}\,,\ exp$ $cla \Rightarrow occ\,,\ occ \Rightarrow med\,,\ med \Rightarrow com\,,\ com \Rightarrow fan\,,\ hon\,,\ \neg fan$
4.	$adm\,,\ \underline{cla}\,,\ epr\,,\ exp$ $cla \Rightarrow occ\,,\ occ \Rightarrow med\,,\ med \Rightarrow com\,,\ com \Rightarrow fan\,,\ hon\,,\ \neg fan$
5.	$adm\,,\ cla\,,\ epr\,,\ exp$ $\underline{occ}\,,\ occ \Rightarrow med\,,\ med \Rightarrow com\,,\ com \Rightarrow fan\,,\ hon\,,\ \neg fan$
6.	$adm\,,\ cla\,,\ epr\,,\ exp$ $occ\,,\ \underline{med}\,,\ med \Rightarrow com\,,\ com \Rightarrow fan\,,\ hon\,,\ \neg fan$
7.	$adm\,,\ cla\,,\ epr\,,\ exp$ $occ\,,\ med\,,\ \underline{com}\,,\ com \Rightarrow fan\,,\ hon\,,\ \neg fan$
8.	$adm\,,\ cla\,,\ epr\,,\ exp\,,\ occ\,,\ med\,,\ com\,,\ \underline{fan}\,,\ hon\,,\ \neg fan$
9.	$adm\,,\ cla\,,\ epr\,,\ exp\,,\ occ\,,\ med\,,\ com\,,\ fan\,,\ hon\,,\ \square$

FIG. 36 – Argument "Croire au fantômes ?" corrigé

Cependant, l'hypothèse $A \models B$ élimine les cas $A = \mathbf{V}$, $B = \mathbf{F}$, ce qui ne laisse subsister que douze lignes. La table complète est représentée à la figure 37 ; on en déduit immédiatement les solutions :

1. $C_1 \not\models D_1$ et $D_1 \models C_1$;
2. $C_2 \models D_2$ et $D_2 \not\models C_2$;
3. $C_3 \not\models D_3$ et $D_3 \not\models C_3$;
4. $C_4 \models D_4$ et $D_4 \models C_4$.

Remarque. Il se peut que, pour des choix particuliers de A, B, X, Y, on ait $C_i \models D_i$ et/ou $D_i \models C_i$ même si c'est faux dans le cas général. Plus concrètement, le résultat négatif $C_1 \not\models D_1$ que nous venons d'obtenir signifie que pour certains choix des formules A, B, C, D, la condition $A \models B$ ne garantit pas $C_1 \models D_1$; c'est le cas notamment si A et Y sont identiquement vraies et si B et X sont identiquement fausses. Cela n'exclut pas que, pour d'autres choix (par exemple celui où les quatre formules sont identiquement vraies), $C_1 \models D_1$ puisse avoir lieu. En revanche, tout résultat positif, tel $D_1 \models C_1$ ou $C_4 \models D_4$, s'entend pour tous les choix de formules compatibles avec l'hypothèse. Dans le même ordre d'idée, rappelons que les énoncés $\not\models (A \Rightarrow B)$ et $\models \neg(A \Rightarrow B)$ ne sont pas équivalents : le second (qui garantit la validité de A et l'inconsistance de B) est nettement plus fort que le premier.

La méthode des tableaux sémantiques est également utilisable ici. A titre d'exemple, le tableau de la figure 38 montre la consistance de la formule $(A \Rightarrow B) \wedge \neg(D_2 \Rightarrow C_2)$, ce qui établit le résultat négatif $D_2 \not\models C_2$.

Il convient de souligner que les méthodes des tables de vérité et des tableaux sémantiques sont toujours utilisables, mais rarement optimales. Dans le cas présent, on peut arriver aux conclusions plus rapidement, en notant qu'a priori il est évident que certaines interprétations

A	B	X	Y	C_1	D_1	C_2	D_2	C_3	D_3	C_4	D_4
V	**V**	**V**	**V**	**V**	**V**	**V**	**V**	**V**	**V**	**V**	**V**
V	**V**	**V**	**F**	**F**	**F**	**V**	**V**	**F**	**F**	**F**	**F**
V	**V**	**F**	**V**	**V**	**V**	**V**	**V**	**V**	**V**	**F**	**F**
V	**V**	**F**	**F**	**V**	**V**	**V**	**V**	**V**	**V**	**V**	**V**
F	**V**	**V**	**V**	**V**	**V**	**V**	**V**	**V**	**V**	**V**	**V**
F	**V**	**V**	**F**	**V**	**F**	**F**	**V**	**V**	**F**	**F**	**F**
F	**V**	**F**	**V**	**V**	**V**	**V**	**V**	**V**	**V**	**F**	**F**
F	**V**	**F**	**F**	**V**	**V**	**V**	**V**	**F**	**V**	**V**	**V**
F	**F**	**V**	**V**	**V**	**V**	**V**	**V**	**V**	**V**	**V**	**V**
F	**F**	**V**	**F**	**V**	**V**	**F**	**F**	**V**	**V**	**F**	**F**
F	**F**	**F**	**V**	**V**	**V**	**V**	**V**	**V**	**V**	**F**	**F**
F	**F**	**F**	**F**	**V**	**V**	**V**	**V**	**F**	**F**	**V**	**V**

FIG. 37 – Analyse de formules par table de vérité

rendent C_i et D_i logiquement équivalentes. De telles interprétations ne doivent naturellement pas être étudiées, puisque nous cherchons à mettre en évidence les differences sémantiques entre les deux formules. Par exemple, C_1 et D_1 sont toujours vraies (et donc logiquement équivalentes) dès que X est fausse ou que Y est vraie ; le problème relatif à C_1 et D_1 peut donc se réduire au problème relatif à $C'_1 =_{def} \neg A$ et $D'_1 =_{def} \neg B$, puisque C_1 se réduit à C'_1, et D_1 se réduit à D'_1 dès que X est vraie et que Y est fausse. Dans le même ordre d'idée, on note que les formules $A \Rightarrow (B \vee A)$ et $B \Rightarrow (A \vee B)$ sont identiquement vraies (valides). En fait, le problème initial se réduit de la sorte au problème concernant les paires suivantes :

1. $C'_1 =_{def} \neg A$ et $D'_1 =_{def} \neg B$;
2. $C'_2 =_{def} A$ et $D'_2 =_{def} B$;
3. $C'_3 =_{def} \neg(X \equiv A)$ et $D'_3 =_{def} \neg(X \equiv B)$;
4. $C'_4 =_{def} X \equiv Y$ et $D'_4 =_{def} X \equiv Y$.

Cette simplification du problème rend les résultats évidents.

Enfin, notons que le tableau sémantique de la figure 38 pouvait être simplifié, comme dans le cas de l'argument "biblique" ; le résultat est donné à la figure 39.

On notera l'emploi d'une nouvelle règle de simplification : la paire $\{(A \Rightarrow B)\,,\, \neg A\}$ est récrite en le singleton $\{\neg A\}$, ces deux ensembles étant logiquement équivalents.

3.5.3 Problèmes

Le coffre partagé. Cinq personnes (A, B, C, D, E) ont des économies en commun dans un coffre. N'ayant pas confiance l'une en l'autre, elles décident que le coffre ne pourra s'ouvrir qu'en présence de A et B, ou de A et C, ou de B, D et E. Combien de serrures le coffre doit-il avoir ? Combien faut-il de clés et à qui les donne-t-on ?

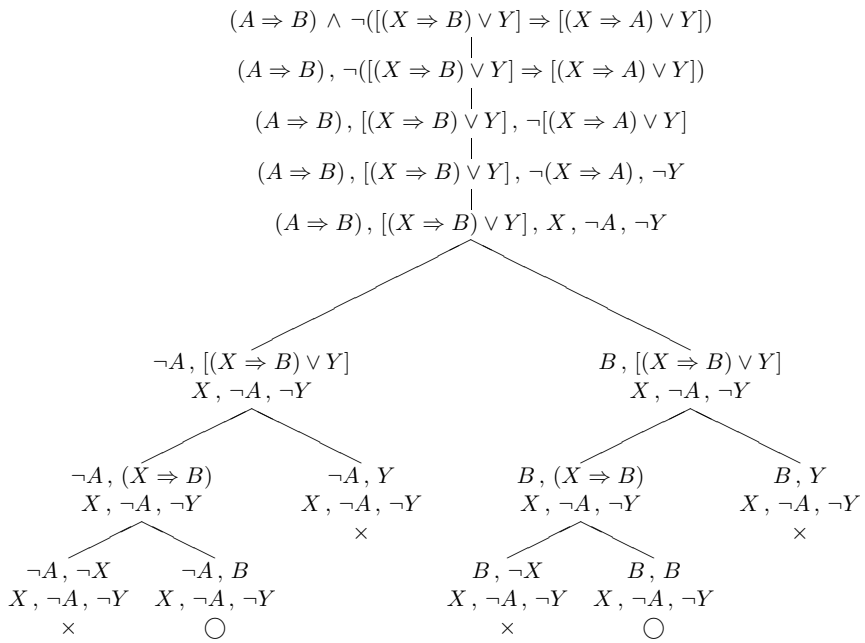


FIG. 38 – Tableau sémantique : $D_2 \not\models C_2$

On introduit les propositions a, b, c, d, e pour modéliser la présence éventuelle de A, B, C, D, E, respectivement. Le coffre peut être ouvert dans toute situation vérifiant la formule

$$\Phi =_{def} (a \wedge b) \vee (a \wedge c) \vee (b \wedge d \wedge e)\,.$$

En utilisant la règle de distributivité de la disjonction sur la conjonction, on voit que Φ est logiquement équivalente à la conjonction des 12 clauses suivantes :

$$\begin{array}{lll} (a \vee a \vee b)\,, & (a \vee a \vee d)\,, & (a \vee a \vee e)\,, \\ (b \vee a \vee b)\,, & (b \vee a \vee d)\,, & (b \vee a \vee e)\,, \\ (a \vee c \vee b)\,, & (a \vee c \vee d)\,, & (a \vee c \vee e)\,, \\ (b \vee c \vee b)\,, & (b \vee c \vee d)\,, & (b \vee c \vee e)\,. \end{array}$$

On peut omettre les répétitions de littéraux au sein d'une clause, ce qui simplifie les clauses en

$$\begin{array}{lll} (a \vee b)\,, & (a \vee d)\,, & (a \vee e)\,, \\ (a \vee b)\,, & (b \vee a \vee d)\,, & (b \vee a \vee e)\,, \\ (a \vee c \vee b)\,, & (a \vee c \vee d)\,, & (a \vee c \vee e)\,, \\ (b \vee c)\,, & (b \vee c \vee d)\,, & (b \vee c \vee e)\,. \end{array}$$

De plus, si une clause (vue comme un ensemble de littéraux) en contient une autre, la clause *contenante* peut être omise ; seules quatre clauses subsistent :

$$1 : (a \vee b)\,,\ 2 : (a \vee d)\,,\ 3 : (a \vee e)\,,\ 4 : (b \vee c)\,.$$

$$(A \Rightarrow B) \wedge \neg([(X \Rightarrow B) \vee Y] \Rightarrow [(X \Rightarrow A) \vee Y])$$
$$|$$
$$(A \Rightarrow B)\,,\ \neg([(X \Rightarrow B) \vee Y] \Rightarrow [(X \Rightarrow A) \vee Y])$$
$$|$$
$$(A \Rightarrow B)\,,\ [(X \Rightarrow B) \vee Y]\,,\ \neg[(X \Rightarrow A) \vee Y]$$
$$|$$
$$(A \Rightarrow B)\,,\ [(X \Rightarrow B) \vee Y]\,,\ \neg(X \Rightarrow A)\,,\ \neg Y$$
$$|$$
$$(A \Rightarrow B)\,,\ (X \Rightarrow B)\,,\ \neg(X \Rightarrow A)\,,\ \neg Y$$
$$|$$
$$(A \Rightarrow B)\,,\ (X \Rightarrow B)\,,\ X\,,\ \neg A\,,\ \neg Y$$
$$|$$
$$(X \Rightarrow B)\,,\ X\,,\ \neg A\,,\ \neg Y$$
$$|$$
$$B\,,\ X\,,\ \neg A\,,\ \neg Y$$
$$\bigcirc$$

FIG. 39 – Tableau sémantique simplifié : $D_2 \not\models C_2$

Ceci montre qu'une solution à quatre serrures $(1, 2, 3, 4)$ convient, avec la distribution de clés suivante :

$$A : 1, 2, 3\,;\ \ B : 1, 4\,;\ \ C : 4\,;\ \ D : 2\,;\ \ E : 3\,.$$

Penser ou payer ! Vous entrez dans un pub écossais et le barman vous dit : "Vous voyez ces trois hommes ? L'un d'eux est Monsieur X, qui dit toujours la vérité, un autre est Monsieur Y, qui ment toujours, et le troisième est Monsieur Z, qui répond au hasard sans écouter les questions. Vous pouvez poser trois questions (appelant une réponse par oui ou non), en indiquant chaque fois lequel des trois doit répondre. Si après cela vous pouvez identifier correctement ces messieurs, ils vous offrent un whisky !". Comment vous y prenez-vous ?

Il est clair que les réponses de Monsieur Z sont sans intérêt, aussi une bonne tactique consistera à repérer d'abord quelqu'un qui n'est pas Monsieur Z ; cela peut se faire au moyen d'une question bien choisie. C'est à ce quelqu'un que l'on posera les deux dernières questions.
On pose à l'un des hommes (I) la question

Votre voisin de gauche (G) est-il plus menteur que votre voisin de droite (D) ?

Examinons, pour les six dispositions possibles, la réponse fournie, étant entendu que Y est plus menteur que Z, lui-même plus menteur que X :

I	G	D	Réponse exacte	Réponse fournie
X	Y	Z	oui	oui
X	Z	Y	non	non
Y	X	Z	non	oui
Y	Z	X	oui	non
Z	X	Y	non	oui/non
Z	Y	X	oui	oui/non

On observe que si la réponse fournie est *oui*, le voisin de gauche n'est jamais Z ; c'est donc à lui que l'on s'adressera pour les deux questions suivantes. de même, si la réponse fournie est *non*, c'est au voisin de droite, qui n'est jamais Z, que l'on s'adressera.

Dans les deux cas, on posera ensuite une question dont on connaît la réponse, par exemple "Etes-vous Monsieur Z ?", qui identifiera ce nouvel interlocuteur (X si "non", Y si "oui"). La troisième question, "Votre voisin de gauche est-il Monsieur Z ?", permettra de compléter les identifications.

L'enquête policière. Cinq suspects (A, B, C, D et E) sont interrogés à propos d'un crime. Voici leurs déclarations :

A : C et D mentent.
B : A et E mentent.
C : B et D mentent.
D : C et E mentent.
E : A et B mentent.

Que peut-on en déduire ?

Si x signifie "X dit la vérité", on peut modéliser les déclarations en les formules

A : $a \equiv (\neg c \wedge \neg d)$
B : $b \equiv (\neg a \wedge \neg e)$
C : $c \equiv (\neg b \wedge \neg d)$
D : $d \equiv (\neg c \wedge \neg e)$
E : $e \equiv (\neg a \wedge \neg b)$

Remarque. On pourrait imaginer d'autres interprétations des déclarations, et donc d'autres modélisations.

Supposons que A dise la vérité ; ceux qui disent le contraire ont donc menti et on en déduit

A : $\neg c, \neg d$
B : $\neg b$
C : $c \equiv \neg d$
D : $d \equiv \neg c$
E : $\neg e$

On obtient une contradiction, ce qui montre que A a menti. La situation est donc

A : $\neg a, c \vee d$
B : $b \equiv \neg e$
C : $c \equiv (\neg b \wedge \neg d)$
D : $d \equiv (\neg c \wedge \neg e)$
E : $e \equiv \neg b$

Si B dit la vérité, on obtient $\neg a$, $c \vee d$, b, $\neg e$, $\neg c$, d.

Si B a menti, on obtient $\neg a$, $c \vee d$, $\neg b$, e, c, $\neg d$

En conclusion, A est certainement menteur mais, pour les quatre autres suspects, il y a deux possibilités : B et D disent la vérité et C et E mentent, ou B et D mentent et C et E disent la vérité.

3.6 La méthode de résolution

La méthode de résolution est la technique de preuve la plus souvent utilisée dans les programmes de démonstration automatique ; elle intervient souvent, sous une forme ou sous une autre, dans les programmes d'intelligence artificielle. Cette méthode opère par réfutation : elle permet de démontrer la validité de A en démontrant l'inconsistance de $\neg A$. Plus généralement, pour démontrer $E \models A$, on prouve l'inconsistance de $E \cup \{\neg A\}$.

On peut appliquer la méthode de résolution à n'importe quel ensemble de formules mais, comme pour les méthodes vues antérieurement, il est plus simple de se limiter à un certain type de formules, appelées formes clausales.

3.6.1 Formes normales

Formes normales en algèbre. L'expression $(x^2 - 4x)(x + 3) + (2x - 1)^2 + 4x - 19$ est un polynôme, mais ses propriétés ne sont pas directement apparentes. On souhaitera donc *normaliser* le polynôme, c'est-à-dire trouver un polynôme équivalent (en fait, égal) mais de forme plus "agréable". Le type de *forme normale* ou de *forme canonique* choisi dépendra des propriétés que l'on souhaite mettre en évidence et/ou utiliser, et aussi de l'existence et de l'efficacité de l'algorithme de recherche de la forme choisie. On a notamment les formes suivantes :

$x^3 + 3x^2 - 12x - 18$ (somme de monômes de degrés décroissants) ;

$(x - 3)(x + 3 - \sqrt{3})(x + 3 + \sqrt{3})$ (produit de facteurs du premier degré) ;

$[(x + 3)x - 12]x - 18$ (forme de Horner).

Les formules propositionnelles, comme les polynômes, peuvent prendre diverses formes normales ; nous en introduisons ici deux.

Forme normale disjonctive. La figure 40 montre qu'une formule est toujours équivalente à une disjonction de conjonctions de littéraux.

On appelle *forme normale disjonctive* (FND) toute disjonction de conjonctions de littéraux. Toute formule est donc équivalente à une FND.
Remarque. Il s'agit de disjonctions et de conjonctions *généralisées*, c'est-à-dire à nombre quelconque (mais fini) de termes.[42]

Un *cube* est une conjonction de littéraux, c'est-à-dire une formule $(\ell_1 \wedge \ell_2 \wedge \cdots \wedge \ell_n)$, $(n \in \mathbb{N})$, où les ℓ_i sont des littéraux. On écrit parfois $\bigwedge\{\ell_1, \ldots, \ell_n\}$, ou $\bigwedge_i \ell_i$, ou simplement $\{\ell_1, \ldots, \ell_n\}$.

[42] NB : $\bigvee \emptyset \leftrightarrow \mathit{false}$, $\bigwedge \emptyset \leftrightarrow \mathit{true}$, $\bigvee\{A\} \leftrightarrow A \leftrightarrow \bigwedge\{A\}$, $\bigvee\{A, B\} \leftrightarrow A \vee B$, $\bigwedge\{A, B\} \leftrightarrow A \wedge B$.

p	q	r	$p \Rightarrow q$	$(p \Rightarrow q) \Rightarrow r$
V	**V**	**V**	**V**	**V**
V	**V**	**F**	**V**	**F**
V	**F**	**V**	**F**	**V**
V	**F**	**F**	**F**	**V**
F	**V**	**V**	**V**	**V**
F	**V**	**F**	**V**	**F**
F	**F**	**V**	**V**	**V**
F	**F**	**F**	**V**	**F**

$$
\begin{array}{ll}
 & (\ p \wedge \ \ q \wedge \ \ r) \\
\vee & (\ p \wedge \neg q \wedge \ \ r) \\
\vee & (\ p \wedge \neg q \wedge \neg r) \\
\vee & (\neg p \wedge \ \ q \wedge \ \ r) \\
\vee & (\neg p \wedge \neg q \wedge \ \ r)
\end{array}
$$

FIG. 40 – Table de vérité et forme normale disjonctive.

Remarque. Dans ce contexte, les connecteurs "0-aires" true et false ne sont pas utilisés.

On observe immédiatement qu'un cube est inconsistant si et seulement s'il contient une paire complémentaire de littéraux ; de plus, le cube vide est le seul cube valide.

Une forme normale disjonctive est inconsistante si et seulement si tous ses cubes sont inconsistants. En particulier, la forme normale disjonctive vide est inconsistante.

Forme normale conjonctive. Une *clause* est une disjonction de littéraux. Une telle formule est parfois représentée par la notation ensembliste $\bigvee\{\ell_i : i = 1, \ldots, n\}$, voire $\{\ell_i : i = 1, \ldots, n\}$.[43]
Remarque. Selon le contexte, la notation $p\overline{q}r$ peut représenter le cube $p \wedge \neg q \wedge r$ ou la clause $p \vee \neg q \vee r$. Cette notation compacte mais ambiguë est à éviter.

La seule clause inconsistante est la *clause vide*, représentée par **F** ou par $\Box$. (On n'utilise pas les connecteurs *true* et *false*.) Une clause est valide si et seulement si elle comporte une paire complémentaire de littéraux. Une *clause unitaire* est une clause composée d'un seul littéral.

Une *forme normale conjonctive* (FNC, ou CNF pour *Conjunctive Normal Form*) est une conjonction de clauses, c'est-à-dire une conjonction de disjonctions de littéraux. Voici un exemple et un contre-exemple :

– $(\neg p \vee q \vee r) \wedge (\neg q \vee r) \wedge (\neg r)$ – CNF
– $(\neg p \vee q \vee r) \wedge \neg(\neg q \vee r) \wedge (\neg r)$ – non CNF

Une forme normale conjonctive est valide si et seulement toutes ses clauses sont valides. En particulier, la forme normale conjonctive vide est valide. Toute formule du calcul des propositions peut être transformée en une forme normale conjonctive équivalente.
Rappel. Les clauses, cubes et formes normales sont des formules ; on les traite souvent comme des ensembles (conjonctifs ou disjonctifs) mais ces ensembles sont toujours *finis*.

Intérêt des formes normales. Une forme normale doit être, idéalement
– assez générale pour que chaque formule soit réductible à une forme normale équivalente,

[43] Il faut se méfier de cette dernière notation : une clause est un ensemble *disjonctif* de littéraux.

– aussi restrictive que possible, pour que les algorithmes qui traitent les formes normales soient plus simples que les algorithmes généraux.

Des formes normales conjonctives ou disjonctives distinctes peuvent être équivalentes.

Exemple. La forme normale disjonctive

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

se simplifie en

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

Algorithme de normalisation. On ne considère que la forme normale *conjonctive*.

1. Eliminer les connecteurs autres que $\neg$, $\vee$, $\wedge$.
2. Utiliser les lois de De Morgan pour propager les occurrences de $\neg$ vers l'intérieur.

$$\neg(A \wedge B) \leftrightarrow (\neg A \vee \neg B)$$
$$\neg(A \vee B) \leftrightarrow (\neg A \wedge \neg B)$$

3. Eliminer les doubles négations.

$$\neg\neg A \leftrightarrow A$$

4. Utiliser les lois de distributivité de $\vee$ par rapport à $\wedge$ pour éliminer $\wedge$ des disjonctions.

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

Une formule en forme conjonctive normale équivaut à un *ensemble* (conjonctif) de clauses ; on dit aussi *forme clausale*.

Exercice. Comment obtenir une forme disjonctive normale équivalente à A au départ d'une forme conjonctive normale équivalente à $\neg A$?

Exemple de normalisation. On récrit la formule $(\neg p \Rightarrow \neg q) \Rightarrow (p \Rightarrow q)$ en forme conjonctive normale.

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

$\neg(\neg\neg p \vee \neg q) \vee (\neg p \vee q)$ (élimination $\Rightarrow$)

$(\neg\neg\neg p \wedge \neg\neg q) \vee (\neg p \vee q)$ (propagation $\neg$)

$(\neg p \wedge q) \vee (\neg p \vee q)$ (double négation)

$(\neg p \vee \neg p \vee q) \wedge (q \vee \neg p \vee q)$ (distributivité)

Une forme normale conjonctive de $(\neg p \Rightarrow \neg q) \Rightarrow (p \Rightarrow q)$ est $(\neg p \vee \neg p \vee q) \wedge (q \vee \neg p \vee q)$. Une forme plus simple est $\neg p \vee q$.

Algorithme de normalisation (variante). Une variante intéressante de l'algorithme de normalisation est représentée à la figure 41. La donnée manipulée est un ensemble conjonctif L de disjonctions généralisées. Initialement, l'unique élément de L est la formule donnée A (vue comme une disjonction généralisée à un terme). La valeur finale de L est une FNC équivalente à A. On appelle *non-clause* toute disjonction (généralisée) dont au moins un terme n'est pas un littéral. La preuve de terminaison est analogue à celle pour la construction des tableaux sémantiques. La valeur finale de L est l'ensemble de clauses cherché.
Remarque. Cet algorithme n'est qu'une reformulation de l'algorithme précédent. L'intérêt est lié à la méthode de résolution vue plus loin.

$L := \{A\}$;
Tant que L comporte une non-clause faire
  $\{ \bigwedge L \leftrightarrow A$ est invariant $\}$
  choisir une non-clause $D \in L$;
  choisir un non-littéral $t \in D$;
  $*$ si $t = \neg\neg t'$ faire
    $D' := (D - t) + t'$;
    $\{ D \leftrightarrow D' \}$
    $L := (L \backslash \{D\}) \cup \{D'\}$
  $*$ si $t = \alpha$ faire
    $t_1 := \alpha_1$; $t_2 := \alpha_2$;
    $D_1 := (D - t) + t_1$; $D_2 := (D - t) + t_2$;
    $\{ D \leftrightarrow D_1 \wedge D_2 \}$
    $L := (L \backslash \{D\}) \cup \{D_1, D_2\}$
  $*$ si $t = \beta$ faire
    $t_1 := \beta_1$; $t_2 := \beta_2$;
    $D' := ((D - t) + t_1) + t_2$;
    $\{ D \leftrightarrow D' \}$
    $L := (L \backslash \{D\}) \cup \{D'\}$

FIG. 41 – Algorithme de normalisation.

Exemple de normalisation (variante)

1.	$\{(\neg p \Rightarrow \neg q) \Rightarrow (p \Rightarrow q)\}$	$Init$
2.	$\{\neg(\neg p \Rightarrow \neg q) \vee (p \Rightarrow q)\}$	$\beta, 1$
3.	$\{\neg(\neg p \Rightarrow \neg q) \vee \neg p \vee q\}$	$\beta, 2$
4.	$\{\neg p \vee \neg p \vee q\,,\ \neg\neg q \vee \neg p \vee q\}$	$\alpha, 3$
5.	$\{\neg p \vee \neg p \vee q\,,\ q \vee \neg p \vee q\}$	$\alpha, 4$

La forme normale requise est donc $(\neg p \vee \neg p \vee q) \wedge (q \vee \neg p \vee q)$.
Elle se simplifie en $(\neg p \vee q) \wedge (\neg p \vee q)$, et puis en $\neg p \vee q$.

Simplifications des formes clausales. Les formes clausales ou ensembles conjonctifs de clauses fournis par l'algorithme de normalisation peuvent souvent être simplifiés.

1. On peut supprimer les répétitions de littéraux au sein d'une même clause.
   Exemple : $(\neg p \vee q \vee \neg p) \wedge (r \vee \neg p) \leftrightarrow (\neg p \vee q) \wedge (r \vee \neg p)$.
2. Les clauses valides (elles contiennent une paire complémentaire de littéraux) peuvent être supprimées.
   Exemple : $(\neg p \vee q \vee p) \wedge (r \vee \neg p) \leftrightarrow (r \vee \neg p)$.
3. Une clause *contenant* une autre clause peut être supprimée.
   Exercice : justifier la règle.
   Exemple : $(r \vee q \vee \neg p) \wedge (\neg p \vee r) \leftrightarrow (\neg p \vee r)$.

Ces simplifications élémentaires sont faciles à mettre en œuvre mais ne conduisent pas à une forme normale unique. Par exemple, elles ne permettent pas de réduire $(p \vee \neg q) \wedge q$ en $p \wedge q$.

3.6.2 La règle de résolution

Définition. On sait qu'un ensemble de clauses S est inconsistant si et seulement si $S \models \Box$. ($\Box$ est la clause vide qui dénote *false*.) Cela suggère de montrer l'inconsistance de S en essayant de dériver $\Box$ (*false*) à partir de S, au moyen d'un mécanisme adéquat.

Soient A, B, X des formules et soit v une interprétation. Supposons $v(A \vee X) = \mathbf{V}$ et $v(B \vee \neg X) = \mathbf{V}$. Si $v(X) = \mathbf{V}$, alors $v(B) = \mathbf{V}$ et donc $v(A \vee B) = \mathbf{V}$. Si $v(X) = \mathbf{F}$, alors $v(A) = \mathbf{V}$ et donc $v(A \vee B) = \mathbf{V}$. En conclusion, $\{(A \vee X), (B \vee \neg X)\} \models (A \vee B)$.

Cette règle très simple est appelée *règle de résolution* dans le cas où X est une proposition et où A, B sont des clauses. Avec la notation habituelle, on l'écrit

$$\frac{A \vee X\,, \quad B \vee \neg X}{A \vee B}$$

Fermeture par résolution. On définit par induction la relation $\vdash_{\mathcal{R}}$ (que nous noterons simplement $\vdash$) entre un ensemble de clauses et une clause ; c'est la plus petite relation vérifiant les deux conditions suivantes :

1. Si $C \in S$, alors $S \vdash C$.
2. Soient $C_1 = (C_1' \vee p)$ et $C_2 = (C_2' \vee \neg p)$;
   si $S \vdash C_1$ et $S \vdash C_2$, alors $S \vdash C_1' \vee C_2'$.

Les deux clauses C_1 et C_2 sont dites *résolvables* (par rapport à p) ; leur *résolvante* est la clause $res(C_1, C_2) =_{def} C_1' \vee C_2'$.
Si S est un ensemble de clauses, S^R dénote la *fermeture de S par résolution*, c'est-à-dire le plus petit sur-ensemble de S contenant les résolvantes de ses éléments. On a $S^R = \{C : S \vdash C\} = \{C : S^R \vdash C\}$.

Adéquation de la règle de résolution. Soient S un ensemble de clauses et C une clause. On doit montrer que si $S \vdash C$, alors $S \models C$. Il suffit de montrer que la relation $\models$ (restreinte aux ensembles de clauses et aux clauses) vérifie les deux conditions définissant la relation $\vdash_{\mathcal{R}}$:

1. Si $C \in S$, alors $S \models C$.

2. Soient $C_1 = (C_1' \vee p)$ et $C_2 = (C_2' \vee \neg p)$;
   si $S \models C_1$ et $S \models C_2$, alors $S \models C_1' \vee C_2'$.

La première condition est évidente, la seconde est une conséquence de l'énoncé $\{(A \vee X), (B \vee \neg X)\} \models (A \vee B)$, valable quelles que soient les formules A, B et X.

Remarque. On déduit de ceci que les ensembles S et S^R sont logiquement équivalents, pour tout ensemble S de clauses.

3.6.3 Complétude de la méthode de résolution

Introduction. Si S est un ensemble de clauses, si A est une clause et si $S \models A$, a-t-on nécessairement $S \vdash_{\mathcal{R}} A$? La réponse est clairement négative : on a

$$\{p, \neg p\} \models q\,,$$

mais on n'a pas

$$\{p, \neg p\} \vdash_{\mathcal{R}} q\,.$$

Cela n'est pas gênant, dans la mesure où on ne cherchera pas à utiliser la résolution pour établir directement $S \models A$, mais plutôt $S, \neg A \models \Box$. On démontrera et utilisera le théorème suivant.

Théorème. Si $S \models \Box$, alors $S \vdash_{\mathcal{R}} \Box$.

Cette "complétude affaiblie" (cas particulier où la clause à dériver est toujours la clause vide) est aussi puissante que la complétude usuelle (non satisfaite ici) puisqu'on peut toujours se ramener au cas particulier. C'est pourquoi la "complétude affaiblie" est nommée simplement "complétude".

Arbre sémantique. Soient S une formule ou un ensemble (conjonctif) de formules et $(p_1, p_2, \ldots)$ une énumération de Π_S, l'ensemble (fini ou dénombrable) des propositions présentes dans S. L'*arbre sémantique* de S est un arbre binaire complet dont toutes les branches de gauche de niveau i sont étiquetées par p_i et toutes les branches de droite de niveau i sont étiquetées par $\neg p_i$.

L'arbre sémantique de S décrit toutes les interprétations possibles de S. Chaque chemin $\mathcal{C}$ dans l'arbre allant de la racine à un nœud n de niveau i définit
- un ensemble de propositions, soit $\Pi(n) = \{p_1, \ldots, p_i\}$;
- une interprétation pour cet ensemble de propositions, soit v_n ; on a $v_n(p_k) = \mathbf{V}$ si $p_k \in \mathcal{C}$ et $v_n(p_k) = \mathbf{F}$ si $\neg p_k \in \mathcal{C}$.

Exemple. Soit $S = \{p \vee q, p \vee r, \neg q \vee \neg r, \neg p\}$, un ensemble de clauses. Le lexique Π_S est $\{p, q, r\}$. Un arbre sémantique relatif à ce lexique est donné à la figure 42. L'arbre est fini puisque Π_S est fini. Comme S est inconsistant, chaque feuille peut être étiquetée par une clause fausse pour l'interprétation correspondante.

Preuve de complétude dans le cas fini. Soit S inconsistant et fini ; on doit prouver $S \vdash \Box$. Comme d'habitude, on souhaite une preuve constructive, c'est-à-dire un moyen effectif d'obtenir $\Box$ au départ de S, par applications répétées de la règle de résolution.

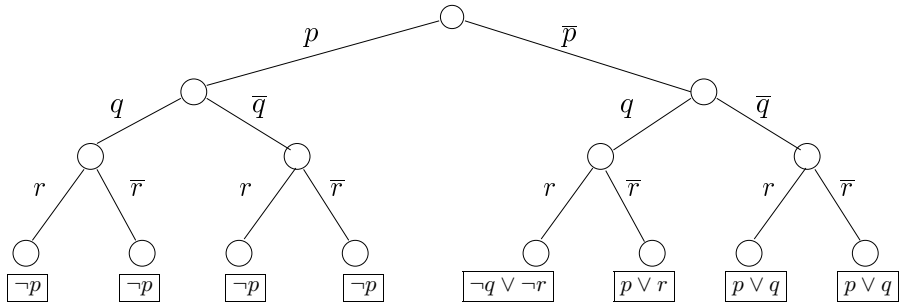


FIG. 42 – Arbre sémantique montrant l'inconsistance d'un ensemble de clauses.

Soit $\mathcal{A}$ un arbre sémantique pour S. Chaque chemin dans cet arbre allant de la racine à un nœud n définit un ensemble de propositions $\Pi(n)$ et une interprétation v_n pour cet ensemble ; v_n rend vrais les littéraux étiquetant le chemin.

S est inconsistant, donc S est falsifié par toutes les interprétations définies par les feuilles de $\mathcal{A}$. Par conséquent, à chaque feuille f de l'arbre correspond au moins une clause $C_f \in S$ telle que

$$\Pi_{C_f} \subseteq \Pi(f) = \Pi_S \quad \text{et} \quad v_f(C_f) = \mathbf{F}\,.$$

(Π_{C_f} est l'ensemble des propositions intervenant dans C_f.) La feuille f est étiquetée C_f.

Soit $S^R = S \cup \{C : S \vdash C\}$. On va montrer qu'il est possible d'étiqueter chaque nœud intérieur n de l'arbre au moyen d'une clause $C_n \in S^R$ telle que

$$\Pi_{C_n} \subseteq \Pi(n) \subseteq \Pi_S \quad \text{et} \quad v_n(C_n) = \mathbf{F}.$$

De cette manière, on aura au noeud racine r une clause $C_r \in S^R$ telle que

$$\Pi_{C_r} \subseteq \Pi(r) \quad \text{et} \quad v_r(C_r) = \mathbf{F}.$$

Or comme $\Pi(r) = \emptyset$ et $v_r(C_r) = \mathbf{F}$, on aura nécessairement $C_r = \Box$.

L'étiquetage se fait en remontant des feuilles vers la racine.
Soit une paire de nœuds n_1, n_2 de l'arbre ayant un nœud père n commun tel que

$$\Pi(n_1) = \Pi(n_2) = \Pi(n) \cup \{p\}.$$

On suppose
$C_{n_1} \in S^R$ et $\Pi_{C_{n_1}} \subseteq \Pi(n_1)$ et $v_{n_1}(C_{n_1}) = \mathbf{F}$
$C_{n_2} \in S^R$ et $\Pi_{C_{n_2}} \subseteq \Pi(n_2)$ et $v_{n_2}(C_{n_2}) = \mathbf{F}$
L'étiquette C_n de n sera C_{n_1} ou C_{n_2} ou $res(C_{n_1}, C_{n_2})$ et ne contiendra ni p ni $\neg p$; cela suggère la politique de choix suivante :

– Si $p \notin \Pi_{C_{n_i}}$ pour $i = 1$ ou 2, alors $C_n = C_{n_i}$.

– Si $p \in \Pi_{C_{n_1}}$ et $p \in \Pi_{C_{n_2}}$:
$v_{n_1}(C_{n_1}) = \mathbf{F}$ implique $C_{n_1} = C'_{n_1} \vee \neg p$ et $v_{n_2}(C_{n_2}) = \mathbf{F}$ implique $C_{n_2} = C'_{n_2} \vee p$.
On pose $C_n = C'_{n_1} \vee C'_{n_2} = res_p(C_{n_1}, C_{n_2})$.

Dans les deux cas, on a $C_n \in S^R$ et $\Pi_{C_n} \subseteq \Pi(n)$ et $v_n(C_n) = \mathbf{F}$.
Ceci achève la démonstration du cas fini.
Un exemple d'étiquetage complet de l'arbre sémantique est donné à la figure 43.

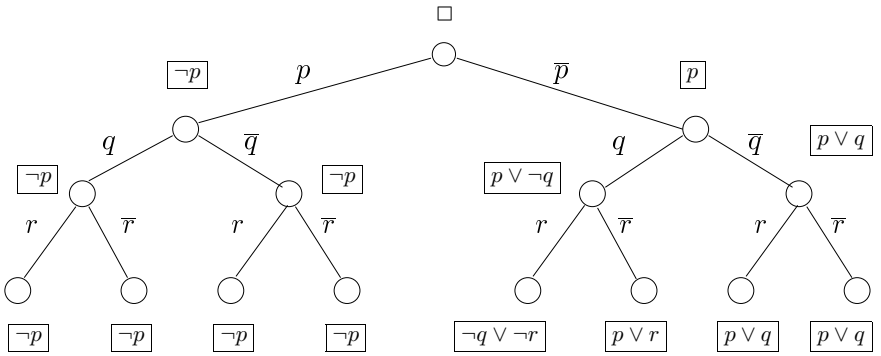


FIG. 43 – Arbre sémantique montrant l'inconsistance d'un ensemble de clauses.

Complétude dans le cas infini, preuve indirecte. On a jusqu'ici supposé que S était fini mais, vu le théorème de compacité, le résultat

$$\Box \in S^R \text{ ssi } S \text{ est inconsistant}$$

reste valable si S est infini. En effet, si S est inconsistant, il admet un sous-ensemble fini S_f inconsistant ; on en déduit $\Box \in S_f^R$, d'où a fortiori $\Box \in S^R$.

Complétude dans le cas infini, preuve directe. Prouver directement ce résultat revient à donner une autre preuve, moins abstraite, du théorème de compacité. On se limite au cas habituel où le lexique Π est un ensemble dénombrable. L'arbre sémantique correspondant $\mathcal{A}$ comporte une infinité de branches, elles-mêmes infinies. A chaque nœud n on associe v_n comme précédemment ; le nœud n est un *nœud-échec* s'il existe $C_n \in S$ telle que $v_n(C_n) = \mathbf{F}$.

On obtient l'arbre $\mathcal{B}$ en élaguant $\mathcal{A}$, de telle sorte que les feuilles de $\mathcal{B}$ soient des nœuds-échecs et que les nœuds intérieurs ne le soient pas. (Les feuilles de $\mathcal{B}$ ne sont pas nécessairement toutes au même niveau.)

L'ensemble S étant inconsistant, toutes les branches de $\mathcal{B}$ sont finies. Un arbre binaire dont toutes les branches sont finies est nécessairement fini (c'est un cas particulier du classique lemme de König, dont nous (re)verrons la démonstration au paragraphe suivant). On applique à $\mathcal{B}$ la technique d'étiquetage introduite pour le cas fini. L'ensemble $S_0 \subset S$ des clauses associées aux feuilles de $\mathcal{B}$ est donc tel que $\Box \in S_0^R$, et S_0 est un sous-ensemble fini

inconsistant de S. On a donc prouvé que tout ensemble inconsistant de clauses construit au moyen d'un lexique dénombrable admet un sous-ensemble fini inconsistant. Toute formule étant logiquement équivalente à un ensemble (conjonctif) de clauses, on a en fait démontré que tout ensemble inconsistant de formules construit au moyen d'un lexique dénombrable admet un sous-ensemble fini inconsistant.

Lemme de König. *Définition.* Un arbre *fini* est un arbre comportant un nombre fini de nœuds. Un arbre est *finitaire* si chaque nœud a un nombre fini de fils.

Lemme. Tout arbre infini finitaire a au moins une branche infinie.
Démonstration. Considérons un arbre infini finitaire. Soit n_0 sa racine. L'arbre est infini, donc n_0 a un nombre infini de descendants. L'arbre est finitaire, donc n_0 a un descendant direct, soit n_1, qui a un nombre infini de descendants. De même, n_1 doit avoir un descendant direct, soit n_2, qui a un nombre infini de descendants. On peut itérer indéfiniment ; on obtient ainsi la branche infinie $n_0, n_1, n_2, \ldots$

3.6.4 Procédure de résolution

Si S est un ensemble de clauses, on note $\mathcal{M}_S$ l'ensemble des modèles de S. L'ensemble S est inconsistant si et seulement si $\mathcal{M}_S = \emptyset$. L'algorithme représenté à la figure 44 met en œuvre la vérification d'inconsistance par résolution.

$S := S_0$; (S_0 est un ensemble de clauses)
$\{\mathcal{M}_S = \mathcal{M}_{S_0}\}$
Tant que $\Box \notin S$, répéter :
  choisir $p \in \Pi_S$,
    $C_1 = (C_1' \vee p) \in S$,
    $C_2 = (C_2' \vee \neg p) \in S$;
  $S := S \cup \{res(C_1, C_2)\}$
  $\{\mathcal{M}_S = \mathcal{M}_{S_0}\}$

FIG. 44 – Procédure de résolution.

Remarque sur l'invariant de boucle. Ajouter à S des conséquences logiques de ses éléments ne change pas l'ensemble $\mathcal{M}_S$ des modèles de S.

Remarque sur la procédure de choix. On admet qu'aucune paire de clauses résolvables ne peut être choisie plus d'une fois ; cela garantit la *terminaison* puisqu'un lexique de n propositions donne lieu à 3^n clauses distinctes non valides. Le programme peut se terminer normalement (garde falsifiée) ou anormalement (plus de choix possible).
Terminaison normale. Si la garde devient fausse, la valeur finale S_f vérifie $\mathcal{M}_{S_f} = \mathcal{M}_{S_0}$ et $\Box \in S_f$, ce qui implique l'*inconsistance* de S_f et de S_0.
Terminaison anormale. Si toutes les résolvantes ont été calculées sans produire $\Box$, on a $\mathcal{M}_{S_f} = \mathcal{M}_{S_0}$ et $\Box \notin S_f$. Cela implique la *consistance* de S_f et de S_0.

Remarque. Une dérivation de $\Box$ (*false*) à partir de S est appelée une *réfutation* de S.

Exemples de réfutations. Soit S l'ensemble des quatre clauses suivantes :

1. $p \vee q$
2. $p \vee r$
3. $\neg q \vee \neg r$
4. $\neg p$

Cet ensemble est inconsistant ; il admet au moins une réfutation. Comme souvent, il en existe plusieurs, telles que

5.	q	$(1,4)$
6.	r	$(2,4)$
7.	$\neg q$	$(3,6)$
8.	$\Box$	$(5,7)$

5.	$p \vee \neg r$	$(1,3)$
6.	q	$(1,4)$
7.	$p \vee \neg q$	$(2,3)$
8.	r	$(2,4)$
9.	p	$(2,5)$
10.	$\neg r$	$(3,6)$
11.	$\neg q$	$(3,8)$
12.	$\neg r$	$(4,5)$
13.	$\neg q$	$(4,7)$
14.	$\Box$	$(4,9)$

Soit S' l'ensemble des deux clauses suivantes :

1. p
2. $\neg p \vee q$

La seule dérivation possible est

3. q $(1,2)$

qui ne produit pas la clause vide. L'ensemble est donc consistant.

Soit S'' l'ensemble des trois clauses suivantes :

1. p
2. $\neg p \vee q$
3. $\neg q$

On obtient immédiatement la réfutation

4. q $(1,2)$
5. $\Box$ $(3,4)$

qui prouve l'inconsistance de l'ensemble.

Efficacité de la résolution. On sait que l'algorithme de résolution est correct et se termine toujours, mais est-il efficace ? Même si on évite de produire plusieurs fois la même résolvante, il est clair que le nombre de clauses produites peut être exponentiel en la taille de S (ou en la taille du lexique Π_S).

La plupart du temps, l'emploi d'une stratégie adaptée permet d'obtenir une efficacité acceptable (dans le cas où l'ensemble de départ est inconsistant). On peut cependant construire des "cas pathologiques" pour lesquels aucune stratégie efficace n'existe.

3.7 Exercice de récapitulation

Soit A la formule $[(p \wedge q) \vee (r \Rightarrow s)] \;\Rightarrow\; [(p \vee (r \Rightarrow s)) \wedge (q \vee (r \Rightarrow s))]$.
On utilise diverses méthodes pour prouver la validité de A.

3.7.1 Méthode directe

Elle consiste à considérer toutes les interprétations. La formule A comporte quatre propositions distinctes, il y a donc $2^4 = 16$ interprétations. Il est plus commode de structurer l'analyse que de procéder en 16 étapes ; on utilise aussi les règles de simplifications élémentaires concernant $a \circ b$, où $\circ$ est un connecteur binaire. Ces règles s'appliquent dès que a et b sont égaux ou opposés, et aussi si l'un des opérandes est $\mathbf{V}$ ou $\mathbf{F}$. On simplifie aussi les doubles négations.

$p = \mathbf{V}$:
$[(T \wedge q) \vee (r \Rightarrow s)] \Rightarrow [(T \vee (r \Rightarrow s)) \wedge (q \vee (r \Rightarrow s))]$,
$[q \vee (r \Rightarrow s)] \;\Rightarrow\; [T \wedge (q \vee (r \Rightarrow s))]$,
$[q \vee (r \Rightarrow s)] \;\Rightarrow\; [q \vee (r \Rightarrow s)]$,
$\mathbf{V}$;
$p = \mathbf{F}$:
$[(F \wedge q) \vee (r \Rightarrow s)] \Rightarrow [(F \vee (r \Rightarrow s)) \wedge (q \vee (r \Rightarrow s))]$,
$[F \vee (r \Rightarrow s)] \;\Rightarrow\; [(r \Rightarrow s) \wedge (q \vee (r \Rightarrow s))]$,
$(r \Rightarrow s) \;\Rightarrow\; [(r \Rightarrow s) \wedge (q \vee (r \Rightarrow s))]$;
$q = \mathbf{V}$: $(r \Rightarrow s) \;\Rightarrow\; [(r \Rightarrow s) \wedge (T \vee (r \Rightarrow s))]$,
$(r \Rightarrow s) \;\Rightarrow\; [(r \Rightarrow s) \wedge T]$,
$(r \Rightarrow s) \;\Rightarrow\; (r \Rightarrow s)$,
$\mathbf{V}$;
$q = \mathbf{F}$: $(r \Rightarrow s) \;\Rightarrow\; [(r \Rightarrow s) \wedge (F \vee (r \Rightarrow s))]$,
$(r \Rightarrow s) \;\Rightarrow\; [(r \Rightarrow s) \wedge (r \Rightarrow s)]$,
$(r \Rightarrow s) \;\Rightarrow\; (r \Rightarrow s)$,
$\mathbf{V}$.

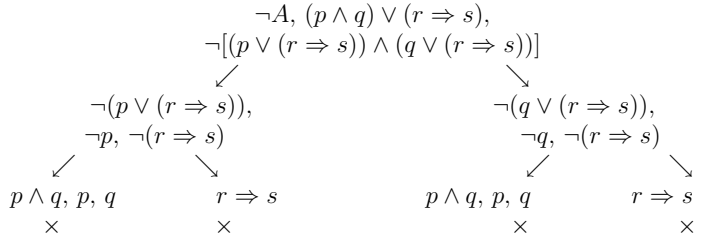
Diverses variantes existent selon le nombre de règles simplificatrices admises (assez réduit ici) et le niveau auquel on les applique (ici, tous).

3.7.2 Méthode algébrique

Elle consiste à utiliser les lois algébriques pour simplifier les formules. Cette méthode est très efficace si on fait les meilleurs choix … mais elle est très inefficace sinon !

$$[(p \wedge q) \vee (r \Rightarrow s)] \;\Rightarrow\; [(p \vee (r \Rightarrow s)) \wedge (q \vee (r \Rightarrow s))]$$
$$\{\text{ Distributivité de } \wedge \text{ sur } \vee \text{ (antécédent) }\}$$
$$[(p \vee (r \Rightarrow s)) \wedge (q \vee (r \Rightarrow s))] \Rightarrow [(p \vee (r \Rightarrow s)) \wedge (q \vee (r \Rightarrow s))]$$
$$\{\, X \;\Rightarrow\; X \;\leftrightarrow\; \mathit{true} \text{ pour tout } X \,\}$$
$$\mathit{true}$$

3.7.3 Tableau sémantique (notation réduite)



On observe une certaine redondance dans les calculs ; c'est le prix à payer pour une méthode facilement mécanisable.

3.7.4 Réduction à la forme conjonctive

On applique les règles habituelles :

$$[(p \wedge q) \vee (r \Rightarrow s)] \Rightarrow [(p \vee (r \Rightarrow s)) \wedge (q \vee (r \Rightarrow s))]$$
$$\neg[(p \wedge q) \vee \neg r \vee s] \vee [(p \vee \neg r \vee s) \wedge (q \vee \neg r \vee s)]$$
$$[(\neg p \vee \neg q) \wedge r \wedge \neg s] \vee [(p \vee \neg r \vee s) \wedge (q \vee \neg r \vee s)]$$
$$(\neg p \wedge r \wedge \neg s) \vee (\neg q \wedge r \wedge \neg s) \vee [(p \vee \neg r \vee s) \wedge (q \vee \neg r \vee s)]$$

Une conjonction est valide si et seulement si tous ses termes sont valides. On prouve donc séparément la validité des deux formules

$A_1 \;=_{def}\; (\neg p \wedge r \wedge \neg s) \vee (\neg q \wedge r \wedge \neg s) \vee p \vee \neg r \vee s$

et

$A_2 \;=_{def}\; (\neg p \wedge r \wedge \neg s) \vee (\neg q \wedge r \wedge \neg s) \vee q \vee \neg r \vee s$.

Chacune de ces formules se réduit à une conjonction de 9 clauses ; on considère seulement la formule A_1. Les clauses sont

$\neg p \vee \neg q \vee p \vee \neg r \vee s$
$\neg p \vee r \vee p \vee \neg r \vee s$
$\neg p \vee \neg s \vee p \vee \neg r \vee s$
$r \vee \neg q \vee p \vee \neg r \vee s$
$r \vee r \vee p \vee \neg r \vee s$
$r \vee \neg s \vee p \vee \neg r \vee s$
$\neg s \vee \neg q \vee p \vee \neg r \vee s$
$\neg s \vee r \vee p \vee \neg r \vee s$
$\neg s \vee \neg s \vee p \vee \neg r \vee s$

Chaque clause comporte une paire complémentaire de littéraux et est donc valide.

3.7.5 Résolution

On commence par réduire $\neg A$ en forme clausale.

$$\neg([(p \wedge q) \vee (r \Rightarrow s)] \Rightarrow [(p \vee (r \Rightarrow s)) \wedge (q \vee (r \Rightarrow s))])$$

$$[(p \wedge q) \vee \neg r \vee s] \wedge \neg[(p \vee \neg r \vee s) \wedge (q \vee \neg r \vee s)]$$

$$(p \vee \neg r \vee s) \wedge (q \vee \neg r \vee s) \wedge [(\neg p \wedge r \wedge \neg s) \vee (\neg q \wedge r \wedge \neg s)]$$

$$\ldots$$

Cela donne 11 clauses :

1. $p \vee \neg r \vee s$
2. $q \vee \neg r \vee s$
3. $\neg p \vee \neg q$
4. $\neg p \vee r$
5. $\neg p \vee \neg s$
6. $r \vee \neg q$
7. r
8. $r \vee \neg s$
9. $\neg s \vee \neg q$
10. $\neg s \vee r$
11. $\neg s$

On déduit la clause vide $\Box$ par résolution :

12. $p \vee s$ $1, 7$
13. $q \vee s$ $2, 7$
14. p $11, 12$
15. q $11, 13$
16. $\neg q$ $3, 14$
17. $\Box$ $15, 16$

Remarque. Les clauses valides ou sur-ensembles d'autres clauses sont inutiles et peuvent être supprimées d'emblée. Dans le cas présent, toutes les clauses minimales (1, 2, 3, 7 et 11) ont été utilisées.

3.7.6 Résolution généralisée

La déduction

$X \vee Y\,,\ \neg X \vee Z \ \models\ Y \vee Z$

reste correcte si X n'est pas un littéral.
On utilise aussi les trois règles suivantes :

$\alpha \vee X \ \models\ \alpha_1 \vee X\,,$
$\alpha \vee X \ \models\ \alpha_2 \vee X\,,$
$\beta \vee X \ \models\ \beta_1 \vee \beta_2 \vee X\,.$

On obtient alors une réfutation plus courte, sans réduction préalable à la forme clausale :

1. $\neg A$
2. $(p \wedge q) \vee \neg r \vee s$ $1, \alpha_1$
3. $\neg[(p \vee \neg r \vee s) \wedge (q \vee \neg r \vee s)]$ $1, \alpha_2$
4. $\neg(p \vee \neg r \vee s) \vee \neg(q \vee \neg r \vee s)$ $3, \beta$
5. $p \vee \neg r \vee s$ $2, \alpha_1$
6. $q \vee \neg r \vee s$ $2, \alpha_2$
7. $\neg(q \vee \neg r \vee s)$ $4, 5, R$
8. $\Box$ $6, 7, R$

3.7.7 Méthode *ad-hoc*

Elle consiste à tirer parti des particularités de la formule étudiée … c'est-à-dire à n'avoir pas de méthode !

On peut observer, par exemple, que

$$[(p \wedge q)\ \vee\ (r \Rightarrow s)] \Rightarrow [(p \vee (r \Rightarrow s))\ \wedge\ (q \vee (r \Rightarrow s))]$$

est de la forme

$$(A \vee B)\ \Rightarrow\ (C \wedge D)\,,$$

et qu'une telle formule est valide si et seulement si les formules $A \Rightarrow C$, $A \Rightarrow D$, $B \Rightarrow C$, $B \Rightarrow D$ sont valides. On doit donc prouver

$$\models\ (p \wedge q)\ \Rightarrow\ (p \vee (r \Rightarrow s))\,,$$
$$\models\ (p \wedge q)\ \Rightarrow\ (q \vee (r \Rightarrow s))\,,$$
$$\models\ (r \Rightarrow s)\ \Rightarrow\ (p \vee (r \Rightarrow s))\,,$$
$$\models\ (r \Rightarrow s)\ \Rightarrow\ (q \vee (r \Rightarrow s))\,,$$

ce qui est évident dans chaque cas.

4 Méthodes déductives : le système de Hilbert

4.1 Introduction

Nous avons vu qu'une théorie est l'ensemble des conséquences logiques d'un ensemble donné de formules, appelées *axiomes* ou *postulats*. Ces conséquences logiques sont appelées *théorèmes*. Développer une théorie consiste donc à repérer les théorèmes parmi les formules construites au moyen du lexique (du langage) utilisé pour introduire les postulats. Deux grandes techniques existent pour cela, la méthode analytique et la méthode synthétique.

Jusqu'ici, nous avons utilisé la méthode analytique, sous la forme d'une procédure de décision. Pour analyser une formule propositionnelle, il suffit de construire un ou deux tableau(x) sémantique(s). Cette approche est excellente ... quand elle est possible. En logique prédicative, qui est la logique des mathématiciens, on ne dispose pas en général d'une procédure de décision ; même quand elle existe, elle peut être difficile à mettre en œuvre. De plus, une procédure de décision pour la validité ne donne guère d'information sur le lien sémantique entre axiomes et théorèmes. Enfin, les procédures de décision sont souvent inefficaces parce qu'elles ne réutilisent pas les résultats. On ne peut pas, en général, accélérer la validation d'un théorème sur base d'autres théorèmes antérieurement démontrés.

Les mathématiciens utilisent le plus souvent la méthode synthétique. Des théorèmes simples sont obtenus à partir des postulats au moyen de quelques mécanismes de raisonnement. Ces mêmes mécanismes, appliqués aux théorèmes simples, permettent de démontrer des théorèmes plus difficiles, et ainsi de suite. L'approche synthétique est également utilisée dans les autres sciences exactes, et notamment en physique ; dans une certaine mesure, on utilise également l'approche synthétique en médecine, en psychologie, en sociologie, etc. Un aspect typique de cette approche est l'exploitation de résultats antérieurs pour produire des résultats nouveaux. Le principal avantage de cette approche est sa généralité. La méthode synthétique s'accommode d'un ensemble infini de postulats (chaque preuve n'en utilise qu'un nombre fini) ; elle permet d'isoler les postulats nécessaires à la production d'un théorème donné, ce qui permet notamment de déterminer si un théorème subsiste ou non quand l'ensemble des postulats est modifié. La théorie est développée de manière modulaire, chaque théorème pouvant être assimilé à un postulat supplémentaire, disponible pour l'obtention de nouveaux théorèmes.

Ces avantages ont un prix. L'obtention de théorèmes par l'approche synthétique est un processus foncièrement non déterministe, pouvant requérir créativité, inventivité ... et tâtonnement, au contraire de l'approche analytique dans laquelle le non-déterminisme est inexistant (tables de vérité) ou peu important (tableaux sémantiques). Des choix inadéquats conduisent à des théorèmes corrects mais inintéressants ; on voit qu'une certaine forme de créativité est nécessaire ici. On peut même dire que le talent du mathématicien consiste essentiellement à opérer les bons choix, ceux qui conduisent à valider (ou à infirmer) les conjectures les plus remarquables.[44]

La logique propositionnelle est suffisamment élémentaire pour être correctement

[44]Une autre facette du talent du mathématicien est l'aptitude à créer de nouveaux ensembles de postulats conduisant à des théories intéressantes.

appréhendée au moyen des seules méthodes analytiques. Nous introduisons néanmoins l'approche synthétique pour préparer le lecteur à son utilisation dans le cadre plus complexe de la logique prédicative.

4.2 Axiomes et règle d'inférence

Le système formel $\mathcal{H}$ est constitué
- de *schémas d'axiomes* ; on a
  1. $\vdash A \Rightarrow (B \Rightarrow A)$
  2. $\vdash (A \Rightarrow (B \Rightarrow C)) \;\Rightarrow\; ((A \Rightarrow B) \Rightarrow (A \Rightarrow C))$
  3. $\vdash (\neg B \Rightarrow \neg A) \Rightarrow (A \Rightarrow B)$
- de la *règle du Modus ponens (MP)* :

$$\frac{\vdash A \qquad \vdash A \Rightarrow B}{\vdash B}$$

A, B et C sont des formules quelconques, n'utilisant que les connecteurs "$\neg$" et "$\Rightarrow$".[45] Un axiome proprement dit s'obtient en *instanciant* l'un des trois schémas, c'est-à-dire en remplaçant A, B, C par des formules. Par exemple, la formule $(p \Rightarrow q) \Rightarrow (p \Rightarrow (p \Rightarrow q))$ est un axiome, obtenu en appliquant la substitution $[A/(p \Rightarrow q),\ B/p]$ au premier schéma. La notation "$\vdash \varphi$" se lit "φ est un théorème". Rappelons ici que tout axiome est un théorème. Le système de Hilbert comporte la seule règle d'inférence "Modus ponens". Cette règle permet d'obtenir le théorème B au départ des théorèmes A et $A \Rightarrow B$.

4.3 Preuves

Une *preuve* dans $\mathcal{H}$ est une séquence de formules, chaque formule étant
- l'instance d'un axiome, ou
- inférée de deux formules la précédant dans la séquence, au moyen de la règle d'inférence Modus ponens.

Par définition, tout élément d'une preuve, et en particulier le dernier, est un théorème ; si A est le dernier élément de la séquence, celle-ci est une *preuve de A.* A titre d'exemple, une preuve de l'implication $p \Rightarrow p$ est donnée à la figure 45.

Remarques. Par facilité, chaque élément d'une preuve est précédé d'un numéro d'ordre et suivi d'une brève justification ; "Axiome 1" veut dire "instance du schéma d'axiome 1" et "4, 3, MP" veut dire "obtenu à partir des formules de numéros 4 et 3 (prémisses) par la règle du Modus ponens". La preuve donnée ici établit que la formule $(p \Rightarrow p)$ est un théorème, ou encore que l'assertion $\vdash (p \Rightarrow p)$ (lire : "$(p \Rightarrow p)$ est un théorème") est un *métathéorème* (c'est-à-dire un théorème au sens mathématique courant ; le préfixe "méta" est souvent omis).[46] L'expression

[45]Il existe des variantes permettant l'emploi de tous les connecteurs habituels, mais la version présentée ici est plus simple, sans être réellement restrictive ; on considère $p \vee q$ comme une abréviation de $\neg p \Rightarrow q$, et $p \wedge q$ comme une abréviation de $\neg(p \Rightarrow \neg q)$.

[46]Signalons que $(p \Rightarrow p)$ est une formule, donc un objet du langage, tandis que $\vdash (p \Rightarrow p)$ est une assertion, donc un objet du métalangage.

1. $\vdash p \Rightarrow ((p \Rightarrow p) \Rightarrow p)$	(Axiome 1)
2. $\vdash (p \Rightarrow ((p \Rightarrow p) \Rightarrow p)) \Rightarrow ((p \Rightarrow (p \Rightarrow p)) \Rightarrow (p \Rightarrow p))$	(Axiome 2)
3. $\vdash (p \Rightarrow (p \Rightarrow p)) \Rightarrow (p \Rightarrow p)$	(1, 2, MP)
4. $\vdash p \Rightarrow (p \Rightarrow p)$	(Axiome 1)
5. $\vdash p \Rightarrow p$	(4, 3, MP)

FIG. 45 – Exemple de preuve dans le système de Hilbert.

$(A \Rightarrow A)$ est un *schéma de théorème* ; toute instance d'un schéma de théorème est un théorème. On transforme facilement la preuve de $(p \Rightarrow p)$ en une preuve de $(p \Rightarrow q) \Rightarrow (p \Rightarrow q)$, par exemple.

La preuve donnée plus haut peut se représenter de manière arborescente (figure 46). Cette représentation est plus naturelle que la représentation séquentielle introduite plus haut, mais n'est guère utilisée à cause de son encombrement.

$$\dfrac{\dfrac{\vdash p \Rightarrow ((p \Rightarrow p) \Rightarrow p) \qquad \vdash (p \Rightarrow ((p \Rightarrow p) \Rightarrow p)) \Rightarrow ((p \Rightarrow (p \Rightarrow p)) \Rightarrow (p \Rightarrow p))}{\vdash (p \Rightarrow (p \Rightarrow p)) \Rightarrow (p \Rightarrow p)} \qquad \vdash p \Rightarrow (p \Rightarrow p)}{\vdash p \Rightarrow p}$$

FIG. 46 – Représentation arborescente d'une preuve.

On peut aussi (figure 47) ne mentionner dans l'arborescence que les numéros des formules impliquées, ce qui réduit l'encombrement.

$$\dfrac{\dfrac{1 \qquad\qquad 2}{\qquad 3 \qquad\qquad 4}}{5}$$

FIG. 47 – Représentation arborescente abrégée d'une preuve.

Il existe une nette analogie entre une preuve de Hilbert (représentée de manière arborescente) et une dérivation de séquent, mais il y a aussi quelques différences :

- Le symbole "$\vdash$" remplace le symbole "$\rightarrow$".
- Les antécédents sont vides (pour l'instant).
- Les succédents comportent un seul élément.
- Le sens de "axiome" a changé.
- L'unique règle est le Modus ponens, qui n'est pas analytique, ni réversible.

En dépit de ces différences, on peut dire que le système de Hilbert est un calcul de séquent (synthétique).

Remarque. Le mot "axiome" a en fait plusieurs sens relativement voisins, mais qu'il convient de distinguer. Dans le cadre de la méthode (analytique) des séquents, un axiome est un séquent d'un type particulier, dont la validité est immédiate. Dans le cadre du système de Hilbert, un axiome est une tautologie d'un certain type, obtenue par instantiation d'un schéma particulier. Dans les deux cas, l'idée de validité est importante. En revanche, dans le langage courant, dans le langage mathématique général et dans le cadre plus technique des théories logiques (surtout prédicatives), les axiomes ne sont pas des tautologies mais des énoncés consistants dont on souhaite étudier l'ensemble des conséquences logiques. Dans le cadre de cette étude seulement, les axiomes sont considérés comme toujours vrais ; il s'agit donc d'une validité "locale", limitée à un certain contexte. En pratique, ce contexte peut paraître universel. C'est le cas de l'énoncé "l'addition est commutative", traduit par la formule $\forall x \, \forall y \, (x+y = y+x)$. Cette formule n'est pourtant valide que si on interprète de manière adéquate le symbole fonctionnel "$+$" et le symbole prédicatif "$=$".

Notons aussi que le mot "*postulat*" est synonyme du mot "*axiome*" mais que ce dernier insiste plus sur l'aspect "vérité universelle" tandis que "*postulat*" met plus l'accent sur l'aspect relatif de la validité. Le célèbre énoncé d'Euclide "Par tout point extérieur à une droite passe une et une seule parallèle à cette droite" est un axiome de la géométrie classique[47] et un postulat — auquel il est parfois profitable de renoncer — de la géométrie moderne. De même, la commutativité de l'addition est un axiome (ou un théorème) en arithmétique et un postulat en théorie des groupes.

4.4 Dérivations

Un ensemble U de formules quelconques étant donné, une *dérivation* ou *preuve avec hypothèses* dans $\mathcal{H}$ est une séquence de formules, chaque formule étant

- une *hypothèse* (élément de U), ou
- une instance d'un axiome, ou
- inférée de deux formules précédentes, au moyen du Modus ponens.

Le métathéorème relatif à une preuve avec hypothèses s'écrit $U \vdash_{\mathcal{H}} A$ ou $U \vdash A$. Un exemple de dérivation est donné à la figure 48. Comme pour les preuves, les lignes sont numérotées et accompagnées d'une courte justification. En outre, l'ensemble des hypothèses est rappelé à chaque ligne. On verra plus loin pourquoi.

Remarque. En général, le dernier élément A d'une dérivation n'est pas un théorème. On verra plus loin que $U \vdash A$ a lieu si et seulement si on a $U \models A$; en particulier, $\vdash A$ a lieu si et seulement si A est une tautologie.

Remarques. Il est souvent plus facile d'établir $A, B, C \vdash D$ que $\vdash A \Rightarrow (B \Rightarrow (C \Rightarrow D))$, mais on montrera qu'une dérivation du premier énoncé se convertit automatiquement en une preuve du second ; la dérivation ci-dessus établit donc indirectement

$$\vdash (p \Rightarrow (q \Rightarrow r)) \Rightarrow (q \Rightarrow (p \Rightarrow r)) \, .$$

[47] Après en avoir longtemps été une conjecture, dont les mathématiciens ont finalement déterminé qu'elle ne pouvait être déduite des autres axiomes.

1.	$p \Rightarrow (q \Rightarrow r),\ q,\ p \vdash p \Rightarrow (q \Rightarrow r)$	(Hypothèse)
2.	$p \Rightarrow (q \Rightarrow r),\ q,\ p \vdash p$	(Hypothèse)
3.	$p \Rightarrow (q \Rightarrow r),\ q,\ p \vdash q \Rightarrow r$	(1, 2, MP)
4.	$p \Rightarrow (q \Rightarrow r),\ q,\ p \vdash q$	(Hypothèse)
5.	$p \Rightarrow (q \Rightarrow r),\ q,\ p \vdash r$	(3, 4, MP)

FIG. 48 – Exemple de dérivation dans le système de Hilbert.

La représentation arborescente, style séquent, reste possible. Les hypothèses deviennent les éléments des antécédents. (Le sens du mot “hypothèse” a donc changé.)

4.5 Quelques résultats utiles

Nous voyons ici trois résultats permettant de raccourcir les preuves ou, plus exactement, d’écrire des “textes” qui ne sont pas, à strictement parler, des preuves, mais des argumentations (souvent plus courtes que les preuves elles-mêmes) établissant l’existence d’une preuve d’un théorème donné.

4.5.1 Principes de composition et de substitution uniforme

Théorème. Tout théorème peut être utilisé comme un axiome dans une preuve.[48]
Démonstration. On obtient une preuve (au sens strict) en remplaçant chaque théorème utilisé comme un axiome par une preuve de ce théorème.

Théorème. Si C est un théorème et si $p_1, \ldots, p_n$ sont des propositions deux à deux distinctes, alors $C(p_1/A_1, \ldots, p_n/A_n)$ est un théorème.
Démonstration. L’application d’une substitution uniforme à toutes les lignes d’une preuve produit toujours une preuve.

4.5.2 Règles d’inférence dérivées

L’écriture $\dfrac{U_1 \vdash A_1\,, \cdots,\ U_n \vdash A_n}{U \vdash B}$

est une *règle d’inférence dérivée* ; elle exprime que s’il existe des dérivations pour chacune des n prémisses, alors il existe une dérivation pour la conclusion. Une règle est *adéquate* ou *correcte* si elle exprime une vérité.

Remarque. Une règle dérivée est correcte … si l’on peut s’en passer, c’est-à-dire si toute dérivation l’utilisant peut être convertie en une dérivation ne l’utilisant pas. Une fois que nous aurons montré que $U \vdash A$ est assimilable à $U \models A$, on pourra prouver facilement qu’une règle est correcte. Par exemple, la règle

$$\frac{U\,,\ \neg X \vdash X}{U \vdash X}$$

est correcte, parce que si X est conséquence logique de $U \cup \{\neg X\}$, alors X est est conséquence logique de U. Nous devrons cependant établir directement certaines règles, nécessaires pour démontrer que les relations $\vdash$ et $\models$ sont coextensives.
Remarque. Dans ce contexte, “U, A” abrège “$U \cup \{A\}$”.

On notera que les principes de composition et de substitution uniforme peuvent se traduire par des règles dérivées, de même que le principe de monotonie, selon lequel une hypothèse supplémentaire n’altère pas les dérivations faites sans elle. On a

$$\frac{U \vdash A \qquad U, A \vdash B}{U \vdash B}$$

$$\frac{U \vdash A}{U[p/B] \vdash A[p/B]}$$

$$\frac{U \vdash A}{U, B \vdash A}$$

4.6 Règle de déduction

Cette règle dérivée essentielle s’écrit $\dfrac{U, A \vdash B}{U \vdash A \Rightarrow B}$.

Remarque. On voit que cette règle provient directement d’une règle (réversible et de type α) du calcul des séquents. Elle formalise une démarche courante en mathématiques :
- on doit démontrer $A \Rightarrow B$;
- on suppose A ;
- on en dérive B.

On a déjà vu l’utilité potentielle de la règle (pour peu que son adéquation soit établie) :
- $p \Rightarrow (q \Rightarrow r),\ q,\ p \vdash r$ est trivial ;
- $\vdash (p \Rightarrow (q \Rightarrow r)) \Rightarrow (q \Rightarrow (p \Rightarrow r))$ ne l’est pas.

Notons enfin que la règle est réversible, la règle inverse étant une simple variante du Modus Ponens.

4.6.1 Adéquation de la règle de déduction

On doit démontrer que toute preuve utilisant la règle de déduction peut se récrire en une preuve ne l’utilisant pas. Cela revient à transformer, étape par étape, une preuve de $U, A \vdash B$ en une preuve de $U \vdash A \Rightarrow B$.

Démonstration. On suppose donnée une preuve Π_1 de $U, A \vdash B$, dont chaque étape est du type $U, A \vdash X$. On construit une preuve Π_2 de $U \vdash (A \Rightarrow B)$ en remplaçant chaque étape de Π_1

$$n.\ U, A \vdash X$$

par une ou plusieurs nouvelles étapes dont la dernière est

[48] On notera la différence de sens entre les deux emplois du mot “théorème” ; la première occurrence aurait pu être “métathéorème”.

$n'.\ U \vdash A \Rightarrow X$.

On distingue quatre cas :

1. X est *un axiome* ;
2. X est *une hypothèse de l'ensemble* U ;
3. X est *la nouvelle hypothèse* A ;
4. X est *inféré par Modus ponens*.

– **Dans les cas 1 et 2,** l'étape
$n.\quad U, A \vdash X$ (Ai ou H)
est remplacée par les trois étapes suivantes :
$n'-2.\quad U \vdash X$ (Ai ou H)
$n'-1.\quad U \vdash X \Rightarrow (A \Rightarrow X)$ (A1)
$n'.\quad U \vdash A \Rightarrow X$ ($n'-2$, $n'-1$, MP)

– **Dans le cas 3,** l'étape $U, A \vdash A$ est remplacée par cinq étapes calquées sur la démonstration de $\vdash (p \Rightarrow p)$ donnée plus haut ; la dernière de ces cinq nouvelles étapes est naturellement $U \vdash (A \Rightarrow A)$.

– **Dans le cas 4,** la preuve Π_1 comporte les étapes
$i.\quad U, A \vdash Y$ (...)
$j.\quad U, A \vdash Y \Rightarrow X$ (...)
$n.\quad U, A \vdash X$ (i, j, MP)
Le préfixe déjà construit de Π_2 comportera
$i'.\quad U \vdash A \Rightarrow Y$ (...)
$j'.\quad U \vdash A \Rightarrow (Y \Rightarrow X)$ (...)
Le fragment de Π_2 relatif à la $n$ième étape de Π_1 sera :
$n'-2.\quad U \vdash (A \Rightarrow (Y \Rightarrow X)) \Rightarrow ((A \Rightarrow Y) \Rightarrow (A \Rightarrow X))$ (A2)
$n'-1.\quad U \vdash (A \Rightarrow Y) \Rightarrow (A \Rightarrow X)$ (j', $n'-2$, MP)
$n'.\quad U \vdash (A \Rightarrow X)$ (i', $n'-1$, MP)

Ceci achève la démonstration.

4.7 Théorèmes et règles dérivées supplémentaires

Dans le système de Hilbert, les preuves sont très faciles à *vérifier* mais nettement plus difficiles à *construire* ; cette situation est habituelle avec les méthodes synthétiques. Nous donnons ici quelques théorèmes utiles et quelques règles dérivées supplémentaires.

4.7.1 Théorèmes supplémentaires

Nous donnons ici neuf théorèmes importants :

1. $\vdash (A \Rightarrow B) \Rightarrow ((B \Rightarrow C) \Rightarrow (A \Rightarrow C))$
2. $\vdash (A \Rightarrow (B \Rightarrow C)) \Rightarrow (B \Rightarrow (A \Rightarrow C))$
3. $\vdash \neg A \Rightarrow (A \Rightarrow B)$
4. $\vdash A \Rightarrow (\neg A \Rightarrow B)$
5. $\vdash \neg\neg A \Rightarrow A$
6. $\vdash A \Rightarrow \neg\neg A$
7. $\vdash (A \Rightarrow B) \Rightarrow (\neg B \Rightarrow \neg A)$
8. $\vdash B \Rightarrow (\neg C \Rightarrow \neg(B \Rightarrow C))$
9. $\vdash (B \Rightarrow A) \Rightarrow ((\neg B \Rightarrow A) \Rightarrow A)$

Il s'agit en fait de schémas de théorèmes ; chacune des lettres A, B et C désigne ici n'importe quelle formule.

Dans la suite, on évitera la *construction* des preuves ; on préférera démontrer simplement l'*existence* d'une preuve. Les notions de *dérivation* et de *règle dérivée* ont pour but de faciliter ces démonstrations d'existence.

A titre d'exemple, une dérivation du théorème 6 est donnée à la figure 49, les théorèmes 1 à 5 étant supposés déjà démontrés. On voit ici l'utilité capitale du principe de composition ; la règle de déduction facilite le travail de justification ... qui n'est quand même pas évident.

1. $A, \neg\neg\neg A \vdash \neg\neg\neg A \Rightarrow \neg A$	(Composition, th. 5)
2. $A, \neg\neg\neg A \vdash \neg\neg\neg A$	(Hypothèse)
3. $A, \neg\neg\neg A \vdash \neg A$	(1, 2, MP)
4. $A \vdash \neg\neg\neg A \Rightarrow \neg A$	(Déduction, 3)
5. $A \vdash (\neg\neg\neg A \Rightarrow \neg A) \Rightarrow (A \Rightarrow \neg\neg A)$	(Axiome 3)
6. $A \vdash A \Rightarrow \neg\neg A$	(4, 5, MP)
7. $A \vdash A$	(Hypothèse)
8. $A \vdash \neg\neg A$	(6, 7, MP)
9. $\vdash A \Rightarrow \neg\neg A$	(Déduction, 8)

FIG. 49 – Justification de $\vdash A \Rightarrow \neg\neg A$.

Remarque. Une règle dérivée évidente, le plus souvent utilisée implicitement, est la *règle d'augmentation* ; elle s'écrit

$$\frac{U \vdash A}{U, B \vdash A}$$

et exprime qu'une hypothèse disponible ne doit pas nécessairement être utilisée.

4.7.2 Quelques autres règles dérivées

Notons d'emblée un puissant moyen de construire des règles dérivées.

Métarègle. Si $\vdash A \Rightarrow B$, alors $\dfrac{U \vdash A}{U \vdash B}$ est une règle dérivée correcte.

Cela formalise une démarche intuitive :

1. Ayant démontré A en supposant U, soit $U \vdash A$,
2. on utilise le théorème $\vdash A \Rightarrow B$
3. et on applique la règle du Modus ponens à (1) et (2) pour obtenir $U \vdash B$.

On obtient ainsi diverses règles dérivées utiles, dont voici quatre exemples :

$$\frac{U \vdash \neg B \Rightarrow \neg A}{U \vdash A \Rightarrow B} \qquad \textit{Contraposée}$$

$$\frac{U \vdash A \Rightarrow B \qquad U \vdash B \Rightarrow C}{U \vdash A \Rightarrow C} \qquad \textit{Transitivité}$$

$$\frac{U \vdash \neg\neg A}{U \vdash A} \qquad \textit{Double négation}$$

$$\frac{U \vdash A \Rightarrow (B \Rightarrow C)}{U \vdash B \Rightarrow (A \Rightarrow C)} \qquad \textit{Echange d'antécédents}$$

La règle de *contraposition* formalise le raisonnement par l'absurde. La règle de *transitivité* formalise l'emploi de lemmes "en cascade" : pour démontrer $\vdash A \Rightarrow B$, on démontre les lemmes $\vdash A \Rightarrow C_1, \vdash C_1 \Rightarrow C_2, \ldots, \vdash C_n \Rightarrow B$. Par application répétée de la règle de transitivité, on déduit $\vdash A \Rightarrow B$. La règle d'*échange d'antécédents* indique que l'ordre dans lequel des hypothèses sont faites n'est pas important. La règle de la double négation est couramment utilisée en mathématique. Elle n'est pas innocente dans le cadre de la philosophie, de la linguistique, de l'informatique. Par exemple, la phrase

> Il n'est pas vrai que je suis mécontent.

n'est pas tout à fait équivalente à

> Je suis content.

De même, un programme qui ne produit pas deux valeurs $x \neq y$, n'est pas nécessairement un programme qui produit deux valeurs $x = y$.

Signalons encore la règle de disjonction des cas, qui s'écrit

$$\frac{U, B \vdash A \qquad U, \neg B \vdash A}{U \vdash A}$$

Voici un schéma de démonstration de cette règle importante.

$U, B \vdash A$	Prémisse
$U \vdash B \Rightarrow A$	Déduction
$\vdash (B \Rightarrow A) \Rightarrow ((\neg B \Rightarrow A) \Rightarrow A)$	Théorème 9, § 4.7.1
$U \vdash (\neg B \Rightarrow A) \Rightarrow A$	MP
$U, \neg B \vdash A$	Prémisse
$U \vdash \neg B \Rightarrow A$	Déduction
$U \vdash A$	MP

Remarque. Un peu de créativité ... ou de patience est nécessaire pour démontrer le théorème utilisé dans le développement ci-dessus.

4.8 Adéquation et complétude du système de Hilbert

Le système de Hilbert est un mécanisme permettant d'engendrer des théorèmes ; on souhaite naturellement que tout théorème soit une tautologie (adéquation) et que toute tautologie soit un théorème (complétude). On peut établir un résultat plus général : si A est une formule et si U est un ensemble de formules, alors

$$U \models A$$

est vrai si et seulement si

$$U \vdash A$$

est vrai. (Seuls les connecteurs de négation et d'implication sont admis.)

4.8.1 Adéquation du système de Hilbert

Montrer que tous les théorèmes sont des tautologies revient à montrer que tous les éléments d'une preuve Π sont des tautologies. C'est une conséquence immédiate des lemmes suivants, eux-mêmes évidents.

Lemme. Toute instance des schémas d'axiomes est une tautologie.

Lemme. Si A et $A \Rightarrow B$ sont des tautologies, alors B est une tautologie.

On démontre de même que si l'assertion $U \vdash A$ apparaît dans une dérivation, alors la formule A est conséquence logique de l'ensemble U.

4.8.2 Lemme de Kalmar

Le système de Hilbert est, nous venons de le prouver, un mécanisme de production de tautologies. Il faut aussi prouver que ce système produit *toutes* les tautologies. Le point de vue constructif adopté dans ce texte nous incite donc à élaborer une stratégie d'utilisation du système de Hilbert, permettant de construire à la demande une preuve dont le dernier élément est une tautologie donnée.

Le système le plus simple de vérification de tautologie est sans doute la méthode des tables de vérité.[49] Le lemme de Kalmar spécifie qu'à chaque ligne d'une table de vérité correspond une dérivation dans le système de Hilbert.

Lemme. Soit A une formule construite à partir des propositions $p_1, \ldots, p_n$ et des connecteurs "$\neg$" et "$\Rightarrow$". Soit v une interprétation. Si on définit p'_k comme p_k ou $\neg p_k$ selon que $v(p_k)$ est **V** ou **F**, et si on définit A' comme A ou $\neg A$ selon que $v(A)$ est **V** ou **F**, on a

$$\{p'_1, \ldots, p'_n\} \vdash A'$$

Exemple. Du fragment de table de vérité

p	q	r	s	$(p \Rightarrow q) \Rightarrow \neg(\neg r \Rightarrow s)$
F	**F**	**V**	**V**	**F**

le lemme de Kalmar permet de déduire

$$\{\neg p, \neg q, r, s\} \ \vdash \ \neg[(p \Rightarrow q) \Rightarrow \neg(\neg r \Rightarrow s)] \,.$$

Remarque. Le lemme de Kalmar permet de "coder" une ligne de table de vérité dans le système de Hilbert ; il contribue donc à établir que $U \models A$ implique $U \vdash A$.

4.8.3 Démonstration du lemme de Kalmar

Remarque préliminaire. On va utiliser les lemmes suivants :

$C \ \vdash \ B \Rightarrow C \,,$

$B \ \vdash \ \neg\neg B \,,$

$\neg B \ \vdash \ B \Rightarrow C \,,$

$\neg C, \, B \ \vdash \ \neg(B \Rightarrow C) \,.$

dont les démonstrations sont évidentes si l'on tient compte respectivement de l'axiome 1 (§ 4.2) et des théorèmes 6, 3 et 8 (§ 4.7.1)

Démonstration. On raisonne par induction sur la structure de la formule A.

Cas de base : la formule A est une proposition p_k.
Si $v(p_k) = \mathbf{V}$, l'énoncé se réduit à $\{\ldots, p_k, \ldots\} \vdash p_k$.
Si $v(p_k) = \mathbf{F}$, l'énoncé se réduit à $\{\ldots, \neg p_k, \ldots\} \vdash \neg p_k$.

Premier cas inductif : la formule A est la négation $\neg B$.

[49] Ne confondons pas "le plus simple" et "le plus efficace".

Premier sous-cas.
$v(B) = \mathbf{F}$ et $v(A) = \mathbf{V}$.
$\{p'_1, \ldots, p'_n\} \vdash B' \,,$
$\{p'_1, \ldots, p'_n\} \vdash \neg B \,,$
$\{p'_1, \ldots, p'_n\} \vdash A \,,$
$\{p'_1, \ldots, p'_n\} \vdash A' \,.$

Deuxième sous-cas.
$v(B) = \mathbf{V}$ et $v(A) = \mathbf{F}$.
$\{p'_1, \ldots, p'_n\} \vdash B'$
$\{p'_1, \ldots, p'_n\} \vdash B \,,$
$B \vdash \neg\neg B \,,$
$\{p'_1, \ldots, p'_n\} \vdash \neg\neg B \,,$
$\{p'_1, \ldots, p'_n\} \vdash \neg A \,,$
$\{p'_1, \ldots, p'_n\} \vdash A' \,.$

Second cas inductif : la formule A est l'implication $B \Rightarrow C$.

Premier sous-cas.
$v(C) = \mathbf{V}$ et $v(A) = \mathbf{V}$.
$\{p'_1, \ldots, p'_n\} \vdash C' \,,$
$\{p'_1, \ldots, p'_n\} \vdash C \,,$
$C \vdash (B \Rightarrow C) \,,$
$\{p'_1, \ldots, p'_n\} \vdash (B \Rightarrow C) \,,$
$\{p'_1, \ldots, p'_n\} \vdash A \,,$
$\{p'_1, \ldots, p'_n\} \vdash A' \,.$

Deuxième sous-cas.
$v(B) = \mathbf{F}$ et $v(A) = \mathbf{V}$.
$\{p'_1, \ldots, p'_n\} \vdash B' \,,$
$\{p'_1, \ldots, p'_n\} \vdash \neg B \,,$
$\neg B \vdash (B \Rightarrow C) \,,$
$\{p'_1, \ldots, p'_n\} \vdash (B \Rightarrow C) \,,$
$\{p'_1, \ldots, p'_n\} \vdash A \,,$
$\{p'_1, \ldots, p'_n\} \vdash A' \,.$

Troisième sous-cas.
$v(B) = \mathbf{V}$, $v(C) = \mathbf{F}$ et $v(A) = \mathbf{F}$.
$\{p'_1, \ldots, p'_n\} \vdash B' \,,$
$\{p'_1, \ldots, p'_n\} \vdash B \,,$
$\{p'_1, \ldots, p'_n\} \vdash C' \,,$
$\{p'_1, \ldots, p'_n\} \vdash \neg C \,,$
$\neg C, \, B \ \vdash \ \neg(B \Rightarrow C) \,,$
$\{p'_1, \ldots, p'_n\} \vdash \neg(B \Rightarrow C) \,,$
$\{p'_1, \ldots, p'_n\} \vdash \neg A \,,$
$\{p'_1, \ldots, p'_n\} \vdash A' \,.$

Remarque. On a raisonné par cas pour établir $\{p'_1, \ldots, p'_n\} \vdash A'$, sans pour autant utiliser la règle dérivée de disjonction des cas.[50] En fait, cette règle dérivée est, comme les autres, une version particulière et formelle d'une technique de raisonnement (informel).

4.8.4 Complétude du système de Hilbert

Soit A une tautologie construite à partir des propositions $p_1, \ldots, p_n$ et des connecteurs "$\neg$" et "$\Rightarrow$". D'après le lemme de Kalmar, on a

$$\{p'_1, \ldots, p'_n\} \vdash A \,,$$

où p'_k est indifféremment p_k ou $\neg p_k$.
(On a toujours $A' = A$.)

De ces 2^n théorèmes on tire, par disjonction des cas sur p'_n, les 2^{n-1} théorèmes suivants :

$$\{p'_1, \ldots, p'_{n-1}\} \vdash A \,,$$

[50] En revanche, cette règle sera utilisée explicitement au paragraphe suivant.

et, plus généralement, les 2^k théorèmes suivants, pour tout $k \in \{0, 1, \ldots, n\}$:

$$\{p'_1, \ldots, p'_k\} \vdash A\,,$$

et donc, en particulier ($k = 0$) le théorème

$$\vdash A\,,$$

ce qui achève la démonstration.