Logique et démonstration automatique : Une introduction à la logique propositionnelle et à la logique du premier ordre

Version abrégée

Stéphane Devismes Pascal Lafourcade Michel Lévy

2022/2023

Ce document présente une version abrégée du cours de logique. Ainsi, la plupart des théorèmes sont admis sans démonstration, et certaines notions sont omises.
Le polycopié complet peut être retrouvé sur
`https://wackb.gricad-pages.univ-grenoble-alpes.fr/inf402/`.
Les références du document original ont été préservées, ce qui explique ici des « trous » dans la numérotation.

Table des matières

Introduction

Les mathématiciens ont raisonné *correctement* durant des siècles *sans connaître* la logique formelle. La logique moderne est née de l'ambition de formaliser (mécaniser) le raisonnement mathématique : ce projet a reçu une impulsion décisive quand il est apparu que l'absence de formalisation pouvait conduire à des contradictions. Le développement actuel de la logique continue avec

— le besoin de prouver la correction des programmes (particulièrement quand ces programmes sont utilisés dans des domaines où la sécurité est en jeu)

— l'ambition de représenter en machine l'ensemble des connaissances mathématiques

Cependant la terminologie des mathématiciens est restée différente de celle des logiciens. Pour exprimer que p implique q, le mathématicien dira par exemple :

p entraîne q
p est une condition suffisante de q
pour que q soit vrai, il suffit que p soit vrai
q est une condition nécessaire de p
pour que p soit vrai, il faut que q soit vrai

Nous restreindrons notre étude à la logique *classique* (par opposition à la logique intuitionniste). La logique classique est la logique à deux valeurs de vérité. De plus cette logique est celle des circuits combinatoires, ce qui explique sa grande importance pratique.

Plan : Nous présentons dans la première partie la logique propositionnelle. Plus précisément dans un premier chapitre nous donnons les définitions et résultats de base de la logique des prédicats. Dans le second chapitre nous parlons de la résolution propositionnelle en introduisant la résolution binaire, la stratégie complète et l'algorithme `DPLL`. Enfin nous illustrons une méthode de raisonnement logique en présentant la déduction naturelle. Dans le seconde partie du cours nous revisitons l'ensemble des notions, résultats et techniques présentés dans la première partie pour la logique du premier ordre.

Exercices : Nous proposons à la fin de chaque chapitre une série d'exercices portant sur le contenu du chapitre. Nous indiquons par des étoiles la difficulté des exercices proposés, plus il y a d'étoiles plus l'exercice est difficile. Nous indiquons par les exercices qui complètent les preuves vues durant le cours. Les exercices proposés sont de trois catégories. Nous avons des exercices :

— Basiques qui aident l'étudiant à se familiariser avec le vocabulaire et à manipuler les notions introduites en cours.
— Techniques qui sont des applications directes ou immédiates des résultats vus en cours.
— Réflexifs qui permettent à l'apprenant à raisonner, démontrer, prouver des résultats à partir des techniques et notions apprises grâce aux deux autres types d'exercices.

Objectifs : Les compétences et connaissances que nous souhaitons transmettre sont les suivantes :

— *Comprendre un raisonnement* : être capable de déterminer si un raisonnement logique est correct ou non.
— *Raisonner*, c'est-à-dire, construire un raisonnement correct utilisant les outils de la logique propositionnelle et du premier ordre.
— *Modéliser et formaliser un problème.*
— *Écrire une preuve rigoureuse.*

Première partie

Logique propositionnelle

Chapitre 1

Logique propositionnelle

Sommaire

ARISTOTE fut un des premiers à essayer de formaliser le raisonnement en utilisant la logique des syllogismes. La logique sert à préciser ce qu'est un raisonnement correct, indépendamment du domaine d'application. Un raisonnement est un moyen d'obtenir une conclusion à partir d'hypothèses données. Un raisonnement *correct* ne dit rien sur la vérité des hypothèses, il dit seulement qu'à partir de *la vérité des hypothèses, nous pouvons déduire la vérité de la conclusion*. Nous commençons l'étude de la logique par les lois de la logique propositionnelle. La logique propositionnelle est la logique *sans quantificateurs* qui s'intéresse uniquement aux lois gouvernant les opérations logiques suivantes : la négation ($\neg$) , la conjonction, autrement dit le « et » ($\wedge$), la disjonction, autrement dit le « ou » ($\vee$), l'implication ($\Rightarrow$) et l'équivalence ($\Leftrightarrow$). Ces opérations sont également appelées *connecteurs*. La logique propositionnelle permet de construire des raisonnements à partir de ces connecteurs. Considérons l'exemple suivant qui comporte trois hypothèses :

1. si Pierre est grand, alors Jean n'est pas le fils de Pierre,
2. si Pierre n'est pas grand, alors Jean est le fils de Pierre,
3. si Jean est le fils de Pierre alors Marie est la sœur de Jean.

Nous concluons que Marie est la sœur de Jean ou Pierre est grand.

Afin de pouvoir raisonner nous extrayons la structure logique des hypothèses. Nous désignons les phrases « Pierre est grand », « Jean est le fils de Pierre », « Marie est la sœur de Jean » respectivement par les lettres p, j, m. Les hypothèses peuvent donc s'écrire :

1. $p \Rightarrow \neg j$,
2. $\neg p \Rightarrow j$,
3. $j \Rightarrow m$,

et la conclusion se formalise en $m \vee p$. Nous montrons alors que les hypothèses impliquent la conclusion indépendamment de la nature des énoncés p, j, m. Pour cela nous prouvons que la formule suivante est vraie quelle que soit la vérité des propositions p, j, m.

$$((p \Rightarrow \neg j) \wedge (\neg p \Rightarrow j) \wedge (j \Rightarrow m)) \Rightarrow (m \vee p).$$

Plan : Nous débutons ce chapitre par la *syntaxe des formules logiques*, c'est-à-dire, les règles permettant d'écrire des formules. Une formule peut être vraie ou fausse, ainsi nous devons être capable d'évaluer le *sens d'une formule*. Pour cela nous introduisons le sens de chaque connecteur ainsi que le calcul de la valeur d'une formule qui en dérive. Nous montrons alors un résultat de compacité qui sera principalement utilisé dans la seconde partie de ce livre. Ensuite nous présentons des *équivalences remarquables* utiles pour simplifier les raisonnements logiques. D'autres méthodes permettent de simplifier les raisonnements logiques, par exemple le *remplacement* et la *substitution* de formule. Nous montrons ensuite comment construire les formes normales conjonctives ou disjonctives d'une formule en utilisant les équivalences remarquables. Ces formes normales permettent d'exhiber facilement les modèles ou les contre-modèles d'une formule. Nous montrons ensuite que la logique propositionnelle est une instance d'une *algèbre de Boole*. Nous introduisons la notion de *fonctions booléennes*. Enfin, nous présentons succinctement l'outil `BDDC`[1] développé par Pascal Raymond. Cet outil permet de manipuler les formules propositionnelles.

1.1 Syntaxe

Avant de raisonner, nous définissons le langage que nous utilisons. Ce langage est celui des formules construites à partir du *vocabulaire* suivant :

— Les constantes : $\top$ et $\bot$ représentant respectivement le *vrai* et le *faux*.
— Les variables : une variable est un identificateur, avec ou sans indice, par exemple x, y_1.
— Les parenthèses : ouvrante (et fermante).
— Les connecteurs : $\neg, \vee, \wedge, \Rightarrow, \Leftrightarrow$ respectivement appelés négation, disjonction (ou), conjonction (et), implication et équivalence.

La syntaxe définit les règles de construction d'une formule de la logique propositionnelle. Nous introduisons deux modes d'écriture d'une formule, l'un strict, l'autre plus souple dans le sens qu'il autorise plusieurs écritures d'une même formule. Cette souplesse est obtenue grâce à l'introduction de priorités entre les connecteurs logiques.

1.1.1 Formules strictes

Ci-dessous, nous donnons les règles de construction d'une formule stricte à partir du vocabulaire donné précédemment.

Définition 1.1.1 (Formule stricte) *Une* formule stricte *est définie de manière inductive comme suit :*

— *$\top$ et $\bot$ sont des formules strictes.*
— *Une variable est une formule stricte.*
— *Si A est une formule stricte alors $\neg A$ est une formule stricte.*
— *Si A et B sont des formules strictes et si $\circ$ est une des opérations $\vee, \wedge, \Rightarrow, \Leftrightarrow$ alors $(A \circ B)$ est une formule stricte.*

Dans la suite, nous désignons par $\circ$ tout connecteur binaire, et nous appellons simplement formule une formule stricte. Les formules différentes de $\top$, $\bot$ et des variables sont des formules décomposables.

Exemple 1.1.2 *L'expression $(a \vee (\neg b \wedge c))$ est une formule stricte construite suivant les règles précédentes. En revanche, $a \vee (\neg b \wedge c)$ et $(a \vee (\neg(b) \wedge c))$ ne sont pas des formules au sens de la définition 1.1.1.*

1. `http://www-verimag.imag.fr/~raymond/home/tools/bddc/`.

L'intérêt de la définition de formules strictes est que les parenthèses permettent de trouver sans ambiguïté la structure des formules. Nous représentons la structure des formules par un arbre, où les feuilles contiennent les constantes ou les variables et les nœuds les connecteurs logiques. Le nœud racine est le connecteur à appliquer en dernier.

Exemple 1.1.3 *La structure de la formule* $(a \vee (\neg b \wedge c))$ *est mise en évidence par l'arbre suivant :*

Une formule peut être vue comme une liste de symboles (connecteurs, parenthèses, variables, et constantes). Un facteur d'une telle liste est une suite de symboles consécutifs dans la liste.

Définition 1.1.4 (Sous-formule) *Nous appelons* sous-formule *d'une formule (stricte) A tout facteur de A qui est une formule (stricte).*

Exemple 1.1.5 $(\neg b \wedge c)$ *est une sous-formule de* $(a \vee (\neg b \wedge c))$.

Nous montrons que les formules sont décomposables d'une façon unique en leurs sous-formules. Ce résultat, précisé par le théorème 1.1.13, est évident sur les exemples. L'unicité de la décomposition implique que nous pouvons identifier une formule et son arbre de décomposition. Ainsi, une sous-formule de la formule A pourra être identifiée comme un sous-arbre de l'arbre représentant la formule A.

Afin de raisonner par induction sur la structure d'une formule, nous définissons la taille d'une formule. Nous remarquons que la taille d'une formule correspond au nombre de connecteurs qu'elle contient.

Définition 1.1.10 (Taille d'une formule) *La* taille d'une formule A*, notée* $|A|$*, est définie inductivement par :*

— $|\top| = 0$ *et* $|\bot| = 0$.
— *Si* A *est une variable alors* $|A| = 0$.
— $|\neg A| = 1 + |A|$.
— $|(A \circ B)| = |A| + |B| + 1$.

Exemple 1.1.11 $|(a \vee (\neg b \wedge c))| =$

La *taille* des formules définie dans la définition 1.1.10 est une mesure utile pour prouver par récurrence des propriétés sur les formules.

Théorème 1.1.13 *Pour toute formule A, un et un seul de ces cas se présente :*

— *A est une variable,*
— *A est une constante,*
— *A s'écrit d'une unique façon sous la forme* $\neg B$ *où B est une formule,*
— *A s'écrit d'une unique façon sous la forme* $(B \circ C)$ *où B et C sont des formules.*

Avec la définition de formules (strictes) nous écrivons de nombreuses parenthèses inutiles comme les parenthèses qui entourent chaque formule. Nous introduisons maintenant plus de souplesse dans notre syntaxe en définissant des priorités.

1.1.2 Formules à priorité

Pour éviter la surabondance des parenthèses, nous définissons les *formules à priorité*.

Définition 1.1.14 (Formule à priorité) *Une* formule à priorité *est définie inductivement par :*
- *$\top$ et $\bot$ sont des formules à priorité,*
- *une variable est une formule à priorité,*
- *si A est une formule à priorité alors $\neg A$ est une formule à priorité,*
- *si A et B sont des formules à priorité alors $A \circ B$ est une formule à priorité,*
- *si A est une formule à priorité alors (A) est une formule à priorité.*

Exemple 1.1.15 *Considérons la formule $a \vee \neg b \wedge c$ qui est une formule à priorité mais pas une formule.*

En général, une formule à priorité n'est pas une formule (stricte). Nous montrons dans l'exercice 2 page 31 que toute formule est une formule à priorité. Afin de pouvoir supprimer des parenthèses sans aucune ambiguïté nous définissons un ordre de priorité entre les différents connecteurs.

Définition 1.1.16 (Ordre de priorité des connecteurs) *La négation est prioritaire, puis dans l'ordre des priorités décroissantes, nous trouvons la conjonction ($\wedge$), la disjonction ($\vee$), l'implication ($\Rightarrow$) et l'équivalence ($\Leftrightarrow$).*

À priorité égale, le connecteur gauche est prioritaire, **sauf pour l'implication** *qui est associative à droite.*[2]

Nous considérons qu'une formule à priorité est *l'abréviation* de la formule reconstituable en utilisant les priorités. Sauf exception, nous identifions une formule et son abréviation. Autrement dit, ce qui nous intéresse dans une formule, ce n'est pas son écriture *superficielle*, c'est sa structure, qui est mise en évidence par la syntaxe « stricte ». Ainsi la taille d'une formule à priorité sera égale à la taille de la formule stricte dont elle est l'abréviation. De même, pour les sous-formules, nous considérerons toujours la formule stricte dont la formule à priorité est l'abréviation.

Exemple 1.1.17 *Nous donnons plusieurs exemples d'abréviation de formule par une formule à priorité :*
- *$a \wedge b \wedge c$ est l'abréviation de*
- *$a \wedge b \vee c$ est l'abréviation de*
- *$a \vee b \wedge c$ est l'abréviation de*

Maintenant que la syntaxe est définie, nous définissons le sens des formules.

1.2 Sens des formules

Nous cherchons à déterminer si une formule est vraie ou fausse indépendamment des valeurs affectées à ses variables. Nous définissons d'abord le sens des connecteurs logiques. Ensuite nous expliquons comment calculer la valeur d'une formule et montrons le théorème de compacité. Nous terminons cette section par la présentation de définitions de notions de base de la logique qui constituent le langage commun des logiciens.

1.2.1 Sens des connecteurs

Nous désignons les valeurs de vérité par 0 pour faux et par 1 pour vrai. La constante $\top$ vaut 1 et la constante $\bot$ vaut 0, ce qui nous conduit, le plus souvent, à confondre les constantes et leurs valeurs, et à utiliser indifféremment $\top$, 1 et vrai, respectivement $\bot$, 0 et faux. Le sens des connecteurs logiques est donné par la table 1.1 page ci-contre qui indique les valeurs des formules de la première ligne suivant les valeurs *assignées* aux variables x et y.

2. C'est-à-dire $a \Rightarrow b \Rightarrow c$ est l'abréviation de $(a \Rightarrow (b \Rightarrow c))$.

x	y	$\neg x$	$x \vee y$	$x \wedge y$	$x \Rightarrow y$	$x \Leftrightarrow y$
0	0					
0	1					
1	0					
1	1					

TABLE 1.1 – Table de vérité des connecteurs.

1.2.2 Valeur d'une formule

Chacun sait évaluer une formule : nous associons à chaque variable de la formule une valeur dans l'ensemble $\mathbb{B} = \{0,1\}$. La valeur de la formule est obtenue en remplaçant les variables par leurs valeurs et en effectuant les opérations suivant la table 1.1. Néanmoins, pour raisonner sur les formules, nous définissons formellement la valeur d'une formule.

Définition 1.2.1 (Assignation) *Une* assignation *est une application de l'ensemble de toutes les variables d'une formule dans l'ensemble* $\mathbb{B}$.

Définition 1.2.2 (Valeur d'une formule) *Soient A une formule et v une assignation,* $[A]_v$ *dénote la valeur de la formule A dans l'assignation v. La* valeur *de* $[A]_v$ *est définie par récurrence sur l'ensemble des formules. Soient A, B des formules, x une variable et v une assignation.*
— $[x]_v = v(x)$.
— $[\top]_v = 1$, $[\bot]_v = 0$.
— $[\neg A]_v = 1 - [A]_v$ *autrement dit pour calculer la valeur de* $\neg A$*, nous soustrayons à* 1 *la valeur de A.*
— $[(A \vee B)]_v = \max\{[A]_v, [B]_v\}$.
— $[(A \wedge B)]_v = \min\{[A]_v, [B]_v\}$.
— $[(A \Rightarrow B)]_v =$ *si* $[A]_v = 0$ *alors* 1 *sinon* $[B]_v$.
— $[(A \Leftrightarrow B)]_v =$ *si* $[A]_v = [B]_v$ *alors* 1 *sinon* 0.

D'après le théorème 1.1.13 page 13, toute formule (stricte) se décompose de façon unique en l'un des cas ci-dessus. Ainsi l'extension de v aux formules est une application des formules dans $\mathbb{B}$. En effet, soient 4 formules A, A', B, B' et deux opérations $\circ$ et $\circ'$ telles que $(A \circ B) = (A' \circ' B')$. Par unicité de la décomposition, $A = A', B = B', \circ = \circ'$, donc la valeur de la formule $(A \circ B)$ est définie uniquement par une et une seule des lignes de la définition de la valeur. Il est clair que la valeur d'une formule ne dépend que de ses variables et de sa structure, aussi l'évaluation d'une formule est présentée sous la forme d'une *table de vérité*.

Définition 1.2.3 (Table de vérité d'une formule) *Une* table de vérité d'une formule *A est un tableau qui représente la valeur de A pour toutes les valeurs possibles des variables de A.*

Chaque ligne de la table de vérité définit une assignation pour les variables des formules présentes dans les colonnes de la table et chaque colonne donne la valeur d'une formule.

Exemple 1.2.4 *Nous donnons la table de vérité des formules suivantes :* $x \Rightarrow y$, $\neg x$, $\neg x \vee y$, $(x \Rightarrow y) \Leftrightarrow (\neg x \vee y)$ *et* $x \vee \neg y$.

x	y	$x \Rightarrow y$	$\neg x$	$\neg x \vee y$	$(x \Rightarrow y) \Leftrightarrow (\neg x \vee y)$	$x \vee \neg y$
0	0					
0	1					
1	0					
1	1					

1.2.3 Définitions et notions élémentaires de logique

Nous listons les notions élémentaires de la logique. Nous illustrons par des exemples et contre-exemples chacune des définitions introduites afin d'en faire comprendre les subtilités.

Définition 1.2.5 (Formules équivalentes) *Deux formules A et B sont* équivalentes *si elles ont la même valeur pour toute assignation.*

Exemple 1.2.6 *Les colonnes des deux formules $x \Rightarrow y$ et $\neg x \vee y$ sont identiques dans l'exemple 1.2.4 page précédente. Ces deux formules sont donc équivalentes. Par contre les formules $x \Rightarrow y$ et $x \vee \neg y$ ne sont pas équivalentes car elles n'ont pas les mêmes tables de vérité.*

Remarque 1.2.7 *Nous n'utilisons pas le symbole du connecteur logique $\Leftrightarrow$ pour dire que A et B sont équivalentes. Nous notons que les formules A et B sont équivalentes par $A \equiv B$ ou simplement $A = B$ si le contexte nous permet de comprendre que le signe égal indique l'équivalence. Ainsi, $x \Rightarrow y = \neg x \vee y$ signifie que la formule $x \Rightarrow y$ est équivalente à la formule $\neg x \vee y$.*

Définition 1.2.8 (Valide, tautologie) *Une formule est* valide *si elle a la valeur* 1 *pour toute assignation. Une formule valide est aussi appelée une* tautologie.

Exemple 1.2.9 *En regardant la table de vérité de l'exemple 1.2.4 page précédente nous obtenons que :*
- *la formule $(x \Rightarrow y) \Leftrightarrow (\neg x \vee y)$ est valide ;*
- *la formule $x \Rightarrow y$ n'est pas valide car*

Notation : $\models A$ dénote le fait que la formule A est valide. Nous pouvons écrire $\models x \vee \neg x$, car $x \vee \neg x$ est une tautologie.

Propriété 1.2.10 *Les formules A et B sont équivalentes si et seulement si la formule $A \Leftrightarrow B$ est valide.*

Preuve : La propriété est une conséquence de la table 1.1 page précédente et des définitions précédentes.

$\Rightarrow$ Si les formules A et B sont équivalentes cela signifie qu'elles ont la même table de vérité, ainsi d'après la définition du connecteur $\Leftrightarrow$ donnée dans la table 1.1 page précédente la table de vérité de $A \Leftrightarrow B$ contient uniquement des 1 donc $A \Leftrightarrow B$ est valide.

$\Leftarrow$ Si la formule $A \Leftrightarrow B$ est valide, nous déduisons que la table de vérité de $A \Leftrightarrow B$ contient uniquement des 1, ainsi d'après la définition du connecteur $\Leftrightarrow$ donnée dans la table 1.1 page précédente les tables de vérité de A et de B coïncident donc les formules A et B sont équivalentes.

$\square$

Définition 1.2.11 (Modèle d'une formule) *Une assignation v qui donne la valeur* 1 *à une formule est un* modèle *de la formule. Nous dirons aussi que v* satisfait *la formule ou que v rend* vraie *la formule.*

Exemple 1.2.12 *$x \Rightarrow y$ a pour modèle*

Cette notion de modèle s'étend aux ensembles de formules comme suit.

Définition 1.2.13 (Modèle d'un ensemble de formules) *Une assignation est un* modèle d'un ensemble de formules *si et seulement si elle est un modèle de chaque formule de l'ensemble.*

Exemple 1.2.14

Propriété 1.2.15 *Une assignation est un modèle d'un ensemble de formules si et seulement si elle est un modèle de la conjonction des formules de l'ensemble.*

Preuve : La preuve est demandée dans l'exercice 11 page 33. $\square$

Exemple 1.2.16 *L'ensemble de formules* $\{a \Rightarrow b, b \Rightarrow c\}$ *et la formule* $(a \Rightarrow b) \wedge (b \Rightarrow c)$ *ont les mêmes modèles.*

Définition 1.2.17 (Contre-Modèle) *Une assignation v qui donne la valeur* 0 *à une formule est un* contre-modèle *de la formule. Nous dirons que v ne satisfait pas la formule ou v rend la formule* fausse.

Exemple 1.2.18 $x \Rightarrow y$ *a pour contre-modèle*

Remarque 1.2.19 (Contre-modèle d'un ensemble de formules) *La notion de contre-modèle s'étend aux ensembles de formules de la même manière que la notion de modèle.*

Définition 1.2.20 (Formule satisfaisable) *Une formule (respectivement un ensemble de formules) est* satisfaisable *s'il existe une assignation qui en est un modèle.*

Définition 1.2.21 (Formule insatisfaisable) *Une formule (respectivement un ensemble de formules) est* insatisfaisable *si elle (respectivement s'il) n'est pas satisfaisable.*

Une formule (respectivement un ensemble de formules) insatisfaisable ne possède pas de modèle. Sa table de vérité ne comporte que des 0. La négation d'une tautologie est donc une formule insatisfaisable.

Exemple 1.2.22

Remarque 1.2.23 *Les logiciens utilisent le mot* consistant *comme synonyme de satisfaisable et* contradictoire *comme synonyme d'insatisfaisable.*

Définition 1.2.24 (Conséquence) *Soient* Γ *un ensemble de formules et A une formule : A est* conséquence *de l'ensemble* Γ *d'hypothèses si tout modèle de* Γ *est modèle de A. Le fait que A soit conséquence de* Γ *est noté par* $\Gamma \models A$.

Exemple 1.2.25 *D'après la table de vérité suivante, la formule* $a \Rightarrow c$ *est conséquence des hypothèses* $a \Rightarrow b$ *et* $b \Rightarrow c$.

a	b	c	$a \Rightarrow b$	$b \Rightarrow c$	$a \Rightarrow c$
0	0	0			
0	0	1			
0	1	0			
0	1	1			
1	0	0			
1	0	1			
1	1	0			
1	1	1			

Remarque 1.2.26 (Validité et conséquence) *Nous notons A est valide par* $\models A$*, car A est valide si et seulement si A est conséquence de l'ensemble vide.*

Maintenant, nous établissons l'équivalence de la validité d'une formule composée d'hypothèses et d'une conclusion avec la conséquence de la conclusion à partir des hypothèses mais aussi avec l'insatifaisabilité des hypothèses et de la négation de la conclusion. Ces relations sont constamment utilisées dans les exercices et les démonstrations.

Propriété 1.2.27 *Soient* $n+1$ *formules* $A_1, \ldots, A_n$, B. *Soit* H_n *la* conjonction *des formules* $A_1, \ldots, A_n$. *Les trois formulations suivantes sont équivalentes :*

1. $A_1, \ldots, A_n \models B$, *c'est-à-dire B est conséquence des hypothèses* $A_1, \ldots, A_n$.
2. *La formule* $H_n \Rightarrow B$ *est valide.*

3. $H_n \wedge \neg B$ *est insatisfaisable.*

Preuve : La propriété est une conséquence de la table 1.1 page 15 et des définitions précédentes. Soient $n+1$ formules $A_1, \ldots A_n, B$ et $H_n = A_1 \wedge \ldots \wedge A_n$.

1 ⇒ 2 : Supposons que $A_1, \ldots, A_n \models B$, c'est-à-dire tout modèle de $A_1, \ldots, A_n$ est aussi modèle de B.

Nous obtenons donc dans tous les cas que $H_n \Rightarrow B$ est valide.

2 ⇒ 3 : Supposons que $H_n \Rightarrow B$ est valide, cela signifie que pour toute assignation v, $[H_n \Rightarrow B]_v = 1$. D'après la table 1.1 page 15 nous avons deux possibilités soit $[H_n]_v = 0$, soit $[H_n]_v = 1$ et $[B]_v = 1$. Or $[H_n \wedge \neg B]_v = \min([H_n]_v, [\neg B]_v) = \min([H_n]_v, 1 - [B]_v)$. Dans les deux cas, nous obtenons $[H_n \wedge \neg B]_v = 0$. Nous concluons donc que $H_n \wedge \neg B$ est insatisfaisable.

3 ⇒ 1 : Supposons que $H_n \wedge \neg B$ est insatisfaisable, c'est-à-dire que pour toute assignation la formule $H_n \wedge \neg B$ est contradictoire. Montrons que les modèles de $A_1, \ldots, A_n$ sont aussi des modèles de B.

Soit v une assignation modèle de $A_1, \ldots, A_n$, il en découle que $[A_i]_v = 1$ pour $i = 1, \ldots, n$ donc $[H_n]_v = [A_1 \wedge \ldots \wedge A_n]_v = 1$. D'après notre hypothèse, nous en déduisons que $[\neg B]_v = 0$. D'où, $1 - [B]_v = 0$. Nous avons donc $[B]_v = 1$, c'est-à-dire que v est aussi un modèle de B.

En utilisant le résultat de l'exercice 7 page 32 nous concluons. □

Exemple 1.2.28 *Nous considérons les deux formules* $a \Rightarrow b$ *et* $b \Rightarrow c$*, nous illustrons le théorème précédent en prouvant que* $a \Rightarrow b, b \Rightarrow c \models a \Rightarrow c$*, soit en montrant que* $(a \Rightarrow b) \wedge (b \Rightarrow c) \Rightarrow (a \Rightarrow c)$ *est une tautologie, soit que* $(a \Rightarrow b) \wedge (b \Rightarrow c) \wedge \neg(a \Rightarrow c)$ *est insatisfaisable. Pour cela nous donnons la table de vérité associée à ces formules.*

a	b	c	$a \Rightarrow b$	$b \Rightarrow c$	$a \Rightarrow c$	$(a \Rightarrow b) \wedge (b \Rightarrow c) \Rightarrow (a \Rightarrow c)$	$(a \Rightarrow b) \wedge (b \Rightarrow c) \wedge \neg(a \Rightarrow c)$
0	0	0	1	1	1		
0	0	1	1	1	1		
0	1	0	1	0	1		
0	1	1	1	1	1		
1	0	0	0	1	0		
1	0	1	0	1	1		
1	1	0	1	0	0		
1	1	1	1	1	1		

1.2.5 Équivalences remarquables

Raisonner par équivalence c'est utiliser les propriétés de l'équivalence (réflexivité, symétrie, transitivité), et la propriété de remplacement d'une formule par une autre formule équivalente pour obtenir de nouvelles équivalences à partir des équivalences déjà prouvées ou admises. Ci-dessous, nous listons des équivalences remarquables de la logique.

1. La disjonction est :
   — Associative, c'est-à-dire, $x \vee (y \vee z) \equiv (x \vee y) \vee z$.
   — Commutative, c'est-à-dire, $x \vee y \equiv y \vee x$.
   — 0 est l'élément neutre de la disjonction, c'est-à-dire, $0 \vee x \equiv x$.
2. La conjonction est :
   — Associative, c'est-à-dire, $x \wedge (y \wedge z) \equiv (x \wedge y) \wedge z$.

— Commutative, c'est-à-dire, $x \wedge y \equiv y \wedge x$.
— 1 est l'élément neutre de la conjonction, c'est-à-dire, $1 \wedge x \equiv x$.

3. La conjonction est distributive sur la disjonction : $x \wedge (y \vee z) \equiv (x \wedge y) \vee (x \wedge z)$.
4. La disjonction est distributive sur la conjonction : $x \vee (y \wedge z) \equiv (x \vee y) \wedge (x \vee z)$.
5. Les lois de la négation :
   — $x \wedge \neg x \equiv 0$.
   — $x \vee \neg x \equiv 1$ (Le *tiers-exclus*).
6. $\neg\neg x \equiv x$.
7. $\neg 0 \equiv 1$.
8. $\neg 1 \equiv 0$.
9. Les lois de De Morgan :
   — $\neg(x \wedge y) \equiv \neg x \vee \neg y$.
   — $\neg(x \vee y) \equiv \neg x \wedge \neg y$.
10. 0 est l'élément absorbant de la conjonction : $0 \wedge x \equiv 0$.
11. 1 est l'élément absorbant de la disjonction : $1 \vee x \equiv 1$.
12. idempotence de la disjonction : $x \vee x \equiv x$.
13. idempotence de la conjonction : $x \wedge x \equiv x$.

Les équivalences 1 page ci-contre à 5 se démontrent à l'aide de tables de vérité [3]. Les équivalences 6 à 13 se déduisent des équivalences 1 page ci-contre à 5, comme nous le montrons dans la section 1.5 page 24. Ci-dessous, nous donnons quelques lois de simplification qui permettent d'alléger les raisonnements logiques.

Propriété 1.2.31 (Lois de simplification) *Pour tout x, y nous avons :*
— $x \vee (x \wedge y) \equiv x$.
— $x \wedge (x \vee y) \equiv x$.
— $x \vee (\neg x \wedge y) \equiv x \vee y$.
— $x \wedge (\neg x \vee y) \equiv x \wedge y$.

Preuve : En utilisant les équivalences remarquables nous prouvons les trois lois de simplification. La preuve est demandée dans l'exercice 12 page 33. □

1.3 Substitution et remplacement

Dans ce paragraphe, nous étendons l'ensemble des formules valides par substitution et remplacement.

1.3.1 Substitution

Définition 1.3.1 (Substitution) *Une* substitution *est une application de l'ensemble des variables dans l'ensemble des formules. L'application d'une substitution σ à une formule consiste à remplacer dans la formule toute variable x par la formule $\sigma(x)$.*

Notation : Soit A une formule, nous notons $A\sigma$ ou $\sigma(A)$ l'application de la substitution σ à la formule A.

Définition 1.3.2 (Support d'une substitution, substitution à support fini) *Le* support *d'une substitution σ est l'ensemble des variables x telles que $x\sigma \neq x$. Une substitution σ à* support fini *est notée $< x_1 := A_1, \ldots, x_n := A_n >$, où $A_1, \ldots, A_n$ sont des formules, $x_1, \ldots, x_n$ sont des variables* distinctes *et la substitution vérifie :*
— *pour i de 1 à n, $x_i\sigma = A_i$*
— *pour toute variable y telle que $y \notin \{x_1, \ldots, x_n\}$, nous avons : $y\sigma = y$*

3. Nous laissons le soin au lecteur de se convaincre de la véracité de ces équivalences en construisant les tables correspondantes.

Exemple 1.3.3 *Soient la formule $A = x \vee x \wedge y \Rightarrow z \wedge y$ et la substitution $\sigma =< x := a \vee b, z := b \wedge c >$, σ appliquée à A donne :*

Le support de σ est fini et comporte les variables x et z.

Propriété 1.3.4 *Soient A une formule, v une assignation et σ une substitution, nous avons $[A\sigma]_v = [A]_w$ où pour toute variable x, $w(x) = [\sigma(x)]_v$.*

Preuve : Soient A une formule, v une assignation et σ une substitution, nous voulons montrer que $[A\sigma]_v = [A]_w$ où pour toute variable x, $w(x) = [\sigma(x)]_v$. Par une récurrence sur la taille des formules, nous obtenons :

Cas de base : $|A| = 0$.

Induction :

Supposons que la propriété soit vraie pour toute formule de taille inférieure ou égale à n, montrons qu'elle est vraie pour une formule A de taille $n+1$:

□

Exemple 1.3.5 *Soit $A = x \vee y \vee d$ une formule. Soit $\sigma =< x := a \vee b, y := b \wedge c >$ une substitution. Soit v une assignation telle que $v(a) = 1$, $v(b) = 0$, $v(c) = 0$, $v(d) = 0$. Nous avons :*

Théorème 1.3.6 *L'application d'une substitution à une formule valide donne une formule valide.*

Preuve : Soient A une formule valide, σ une substitution et v une assignation quelconque. D'après la propriété 1.3.4 : $[A\sigma]_v = [A]_w$ où pour toute variable x, $w(x) = [\sigma(x)]_v$. Puisque A est valide, $[A]_w = 1$. Par suite $A\sigma$ vaut 1 dans toute assignation, c'est donc une formule valide. □

Exemple 1.3.7 *Soit A la formule $\neg(p \wedge q) \Leftrightarrow (\neg p \vee \neg q)$. Cette formule est valide, c'est une équivalence remarquable. Soit σ la substitution suivante : $< p := (a \vee b), q := (c \wedge d) >$. La formule*

La substitution est définie sur les formules, pour l'appliquer sans erreur aux formules à priorité, il suffit que l'image, par la substitution, de toute variable soit une variable, une constante ou une formule entre parenthèses.

1.3.2 Remplacement

La notion de substitution ne nous permet pas de remplacer une formule par une formule, nous introduisons la notion de remplacement à cette fin.

Définition 1.3.8 (Remplacement) *Soient A, B, C, D des formules. La formule D est obtenue en remplaçant dans C certaines occurrences de A par B, s'il existe une formule E et une variable x telles que, $C = E < x := A >$ et $D = E < x := B >$.*

Exemple 1.3.9 *Considérons la formule $C = ((a \Rightarrow b) \vee \neg(a \Rightarrow b))$.*

- *La formule obtenue en remplaçant toutes les occurrences de $(a \Rightarrow b)$ par $(a \wedge b)$ dans C est*

  elle est obtenue en considérant la formule $E = (x \vee \neg x)$ et les substitutions $< x := (a \wedge b) >$ et $< x := (a \Rightarrow b) >$.
- *La formule obtenue en remplaçant la première occurrence de $(a \Rightarrow b)$ par $(a \wedge b)$ dans C est*

  elle est obtenue en considérant la formule $E = (x \vee \neg(a \Rightarrow b))$ et les substitutions $< x := (a \wedge b) >$ et $< x := (a \Rightarrow b) >$.

La différence entre une substitution et un remplacement est qu'une substitution remplace un ensemble de variables par des formules alors qu'un remplacement remplace les occurrences de certaines formules par une autre formule en utilisant des substitutions.

Théorème 1.3.10 *Soient C une formule et D la formule obtenue en remplaçant, dans C, des* occurrences *de la formule A par la formule B, alors $(A \Leftrightarrow B) \models (C \Leftrightarrow D)$.*

Preuve : Par définition du remplacement, il existe une formule E et une variable x telles que, $C = E < x := A >$ et $D = E < x := B >$. Supposons que v est une assignation modèle de $(A \Leftrightarrow B)$. Nous avons donc $[A]_v = [B]_v$. D'après la propriété 1.3.4 page ci-contre :

- $[C]_v = [E]_w$ où w est identique à v sauf que $w(x) = [A]_v$
- $[D]_v = [E]_{w'}$ où w' est identique à v sauf que $w'(x) = [B]_v$

Puisque $[A]_v = [B]_v$, les assignations w et w' sont identiques, donc $[C]_v = [D]_v$. Par suite v est modèle de $(C \Leftrightarrow D)$. □

Corollaire 1.3.11 *Soient C une formule et D la formule obtenue en remplaçant, dans C, une* occurrence *de la formule A par la formule B, alors $A \equiv B$ implique $C \equiv D$.*

Preuve : Si $A \equiv B$, alors la formule $(A \Leftrightarrow B)$ est valide (propriété 1.2.10), donc la formule $(C \Leftrightarrow D)$ également puisqu'elle est, d'après le théorème ci-dessus, la conséquence de $(A \Leftrightarrow B)$, par suite $C \equiv D$. □

Exemple 1.3.12 *Le remplacement d'une occurrence d'une formule A par une occurrence de B est mis en évidence par des boîtes marquant ces occurrences.*

- *D'après théorème 1.3.10 : $p \Leftrightarrow q \models (p \vee (\boxed{p} \Rightarrow r)) \Leftrightarrow (p \vee (\boxed{q} \Rightarrow r))$.*
- *D'après le corollaire 1.3.11 : $(\neg(p \vee q) \Rightarrow (\boxed{\neg(p \vee q)} \vee r)) \equiv (\neg(p \vee q) \Rightarrow (\boxed{(\neg p \wedge \neg q)} \vee r))$, puisque $\neg(p \vee q) \equiv (\neg p \wedge \neg q)$.*

Remarque 1.3.13 *Le théorème précédent et son corollaire s'appliquent aux formules. Quand nous faisons un remplacement directement sur une formule à priorité, nous devons nous assurer que ce remplacement reste correct sur les formules (strictes) sous peine de commettre des erreurs. Par exemple, considérons les deux équivalences $a \wedge \boxed{b} \equiv a \wedge \boxed{b}$ et $\neg c \Rightarrow d \equiv c \vee d$. Remplaçons b à gauche par $\neg c \Rightarrow d$ et à droite par $c \vee d$. Nous observons que bien que $\neg c \Rightarrow d \equiv c \vee d$, $a \wedge \neg c \Rightarrow d \not\equiv a \wedge c \vee d$, car pour $a = c = d = 0$, la formule de gauche vaut 1 et celle de droite vaut 0. Ici le corollaire ne doit pas être appliqué à l'occurrence encadrée car $a \wedge \neg c \Rightarrow d$ est une abréviation de $((a \wedge \neg c) \Rightarrow d)$, donc $\neg c \Rightarrow d$ n'apparaît pas comme une occurrence possible de $a \wedge \neg c \Rightarrow d$.*

Nous avons défini les remplacements à partir des substitutions, nous allons maintenant appliquer les remplacements à une formule afin de la transformer en une formule en forme normale équivalente.

1.4 Formes Normales

Mettre une formule en forme normale consiste à la transformer en une formule équivalente ayant des propriétés structurelles. Nous introduisons deux notions de formes normales : la forme normale disjonctive qui permet de mettre en évidence les modèles et la forme normale conjonctive qui exhibe les contre-modèles. La définition de forme normale nécessite l'introduction des concepts de littéral, monôme et clause.

Définition 1.4.1 (Littéral, monôme, clause)
- *Un* littéral *est une variable ou la négation d'une variable.*
- *Un* monôme *est une conjonction de littéraux.*
- *Une* clause *est une disjonction de littéraux.*

Exemple 1.4.2 *Nous illustrons par quelques exemples simples ces nouvelles notions :*
- $x, y, \neg z$ *sont des littéraux.*
- $x \wedge \neg y \wedge z$ *est un monôme dont l'unique modèle est* $x = 1, y = 0, z = 1$.
- *Le monôme* $x \wedge \neg y \wedge z \wedge \neg x$ *comporte une variable et sa négation : il vaut* 0.
- $x \vee \neg y \vee z$ *est une clause dont l'unique contre modèle est* $x = 0, y = 1, z = 0$.
- *La clause* $x \vee \neg y \vee z \vee \neg x$ *comporte une variable et sa négation : elle vaut* 1.

1.4.1 Transformation en forme normale

Nous introduisons la notion de forme normale et montrons qu'il est toujours possible de transformer une formule en une formule équivalente en forme normale.

Définition 1.4.3 (Forme normale) *Une formule est en* forme normale *si elle n'utilise que les opérateurs* $\wedge, \vee, \neg$ *et que les négations sont uniquement appliquées aux variables.*

Exemple 1.4.4 *La formule* $\neg a \vee b$ *est en forme normale, alors que la formule* $a \Rightarrow b$ *n'est pas en forme normale bien qu'elle soit équivalente à la première.*

Nous expliquons maintenant comment transformer toute formule en une formule en forme normale équivalente grâce à des remplacements. Appliquer l'équivalence $A \equiv B$ à la formule C, c'est remplacer dans C une occurrence de A par une occurrence de B : nous avons prouvé dans le théorème 1.3.10 page précédente qu'un tel remplacement change C en une formule équivalente. Ainsi pour transformer une formule en une formule équivalente de forme normale, nous appliquons les transformations suivantes :

1. **Élimination des équivalences** : remplacer une occurrence de $A \Leftrightarrow B$ par l'une des sous-formules :
   (a) $(\neg A \vee B) \wedge (\neg B \vee A)$.
   (b) $(A \wedge B) \vee (\neg A \wedge \neg B)$.
2. **Élimination des implications** : remplacer une occurrence de $A \Rightarrow B$ par $\neg A \vee B$.
3. **Déplacement des négations** : remplacer une occurrence de
   (a) $\neg\neg A$ par A.
   (b) $\neg(A \vee B)$ par $\neg A \wedge \neg B$.
   (c) $\neg(A \wedge B)$ par $\neg A \vee \neg B$.
   ainsi les négations ne portent que sur des variables.

En appliquant ces trois transformations dans l'ordre indiqué, il est clair que la formule initiale a été transformée en une formule en forme normale équivalente. Dans l'exercice 25 page 34, nous prouvons que l'ordre des transformations n'est pas important : en effectuant les transformations ci-dessus dans un ordre quelconque, nous obtenons finalement une formule équivalente en forme normale.

Remarque 1.4.5 *Il est par exemple recommandé de remplacer une sous-formule de la forme* $\neg(A \Rightarrow B)$ *par* $A \wedge \neg B$, *ce qui en fait combine une élimination de l'implication et un déplacement de la négation. En pratique, il est plus efficace de simplifier le plus tôt possible de la façon suivante :*
- *Remplacer par* 0 *une conjonction qui comporte soit comporte une formule et sa négation, soit un* 0.
- *Remplacer par* 1 *une disjonction qui comporte soit une formule et sa négation, soit un* 1.
- *Remplacer* $\neg 1$ *par* 0 *et* $\neg 0$ *par* 1.
- *Enlever les* 0 *des disjonctions et les* 1 *des conjonctions.*
- *Appliquer les simplifications* $x \vee (x \wedge y) \equiv x$, $x \wedge (x \vee y) \equiv x$, $x \vee (\neg x \wedge y) \equiv x \vee y$.
- *Appliquer l'idempotence de la disjonction et de la conjonction.*

1.4.2 Transformation en forme normale disjonctive (somme de monômes)

La forme normale disjonctive permet de trouver facilement des modèles.

Définition 1.4.6 (Forme normale disjonctive) *Une formule est une* forme normale disjonctive *(en bref fnd) si et seulement si elle est une disjonction (somme) de monômes.*

L'intérêt des formes normales disjonctives est de mettre en évidence les modèles.

Exemple 1.4.7 $(x \land y) \lor (\neg x \land \neg y \land z)$ *est une fnd composée de deux monômes, chacun d'eux nous donne un des deux modèles possibles :*

En partant d'une forme normale et en distribuant toutes les conjonctions sur les disjonctions, nous obtenons une disjonction de monômes. Pour cela il faut aussi savoir regrouper plusieurs applications de la distributivité.

Exemple 1.4.8 *Dans une transformation en disjonction de monômes, nous pouvons appliquer de gauche à droite l'équivalence* $(a \lor b) \land (c \lor d \lor e) \equiv$

La transformation par équivalence d'une formule en une disjonction de monômes peut être utilisée pour déterminer si une formule est valide ou non. Soit A une formule dont nous souhaitons déterminer la validité : Nous transformons $\neg A$ en une disjonction de monômes *équivalente* à $\neg A$. Si $\neg A = 0$ alors $A = 1$ donc A est valide, sinon, toutes simplifications étant faites, $\neg A$ est égal à une disjonction de monômes non nuls, qui nous donnent des modèles de $\neg A$, donc des contre-modèles de A.

Exemple 1.4.9 *Soit* $A = (p \Rightarrow (q \Rightarrow r)) \Rightarrow (p \land q \Rightarrow r)$. *Déterminons si A est valide par transformation de* $\neg A$ *en disjonction de monômes.*

Exemple 1.4.10 *Soit* $A = ((a \Rightarrow b) \land c) \lor (a \land d)$. *Déterminer si A est valide par transformation de* $\neg A$ *en disjonction de monômes.*

$\neg A$	
$\equiv \neg((a \Rightarrow b) \land c) \land \neg(a \land d)$	*par déplacement des négations*
$\equiv (\neg(a \Rightarrow b) \lor \neg c) \land (\neg a \lor \neg d)$	*par déplacement des négations*
$\equiv (a \land \neg b \lor \neg c) \land (\neg a \lor \neg d)$	*par combinaison du déplacement d'une négation et élimination de l'implication*
$\equiv (a \land \neg b \land \neg a) \lor (a \land \neg b \land \neg d) \lor (\neg c \land \neg a) \lor (\neg c \land \neg d)$	*par distributivité de la disjonction sur la conjonction*
$\equiv (a \land \neg b \land \neg d) \lor (\neg c \land \neg a) \lor (\neg c \land \neg d)$	*par simplification*

Nous obtenons 3 *modèles de* $\neg A$ *(*$a = 1, b = 0, d = 0$ *;* $a = 0, c = 0$ *;* $c = 0, d = 0$*), c'est-à-dire, des contre-modèles de A. Donc A n'est pas valide.*

1.4.3 Transformation en forme normale conjonctive (produit de clauses)

La forme normale conjonctive permet d'exhiber facilement des contre-modèles.

Définition 1.4.11 (Forme normale conjonctive) *Une formule est une* forme normale conjonctive *(en bref fnc) si et seulement si elle est une conjonction (produit) de clauses.*

L'intérêt des formes normales conjonctives est de mettre en évidence des contre-modèles.

Exemple 1.4.12 $(x \lor y) \land (\neg x \lor \neg y \lor z)$ *est une fnc, qui a deux contre-modèles*

Par convention, nous considérons que 0 et 1 sont des disjonctions de monômes et des conjonctions de clauses. Nous appliquons la distributivité (inhabituelle) de la disjonction sur la conjonction, autrement dit nous remplaçons toute sous-formule $A \lor (B \land C)$ par $(A \lor B) \land (A \lor C)$, et toute sous-formule $(B \land C) \lor A$ par $(B \lor A) \land (C \lor A)$.

Exemple 1.4.13 *Nous reprenons l'exemple 1.4.8 page précédente et obtenons* $(a \land b) \lor (c \land d \land e) \equiv$

La transformation par équivalence d'une formule en une conjonction de clauses de littéraux peut aussi être utilisée pour déterminer si une formule est valide ou non. Soit A une formule dont nous souhaitons déterminer la validité : Nous transformons A en une conjonction de clauses *équivalente* à A. Si $A = 1$ alors A est valide, sinon, toutes simplifications étant faites, A est égale à une conjonction de clauses non égales à 1, et chacune de ces disjonctions nous donne un contre-modèle de A.

1.5 Algèbre de Boole

Cette notion fut introduite par le mathématicien britannique George Boole au milieu du XIX^{e} siècle. Elle permet notamment de traduire les propositions en équations (généralement, cette écriture est plus concise). Nous rappelons d'abord la définition d'une algèbre de Boole. Nous déduisons alors que la logique propositionnelle est une algèbre de Boole. Ensuite nous prouvons certaines propriétés usuelles des algèbres de Boole. Nous terminons cette section en présentant la notion de dualité.

1.5.1 Définition et notations

La définition d'algèbre de Boole donnée ci-dessous est minimale en nombre d'axiomes.

Définition 1.5.1 *Une* algèbre de Boole *est un ensemble d'au moins deux éléments,* 0, 1, *et trois opérations, complément (le complément de x est noté* $\bar{x}$*), somme* $(+)$ *et produit* $(.)$*, qui vérifient les axiomes suivants :*

1. *la somme est :*
   - *associative :* $x + (y + z) = (x + y) + z$,
   - *commutative :* $x + y = y + x$,
   - 0 *est élément neutre de la somme :* $0 + x = x$,
2. *le produit est :*
   - *associatif :* $x.(y.z) = (x.y).z$,
   - *commutatif :* $x.y = y.x$,
   - 1 *est élément neutre du produit :* $1.x = x$,
3. *le produit est distributif sur la somme :* $x.(y + z) = (x.y) + (x.z)$,
4. *la somme est distributive sur le produit :* $x + (y.z) = (x + y).(x + z)$,
5. *les lois de la négation :*
   - $x + \bar{x} = 1$,
   - $x.\bar{x} = 0$.

Attention, le fait que la distributivité de la somme sur le produit n'est pas usuelle rend son application propice aux erreurs.

Comme indiqué précédemment dans la sous-section 1.2.5 page 18, nous pouvons prouver en utilisant des tables de vérité que la logique propositionnelle vérifie l'ensemble des axiomes d'une algèbre de Boole. Ainsi nous pouvons considérer la logique propositionnelle comme la plus petite algèbre de Boole, car elle contient deux éléments. De ce fait, nous pouvons utiliser les notations booléennes (plus condensées) en lieu et place des notations propositionnelles, comme indiqué dans la table de correspondance donnée dans la figure 1.1. Nous conseillons de les utiliser pour effectuer de gros calculs. Par ailleurs, observez que ces notations sont fréquemment utilisées dans le domaine du matériel.

notations logiques	notations booléennes
$\neg a$	$\overline{a}$
$a \wedge b$	$a.b$
$a \vee b$	$a+b$
$a \Rightarrow b$	$\overline{a}+b$
$a \Leftrightarrow b$	$a.b+(\overline{a}.\overline{b})$ ou $(a+\overline{b}).(\overline{a}+b)$

FIGURE 1.1 – Correspondance entre l'algèbre de Boole et la logique propositionnelle.

Remarque 1.5.2 (Conventions d'écriture)

— *Parfois, la négation est utilisée à la place du complémentaire. Par exemple, $-a$ dénote $\overline{a}$ et $-(a+b)$ représente $\overline{a+b}$.*

— *Afin de permettre l'omission du point, dans les notations booléennes, les variables sont souvent dénotées par une seule lettre (avec ou sans indice). Ainsi lorsque a et b sont des variables, des formules entre parenthèses ou sous une barre de négation, nous abrégeons a.b en ab suivant une pratique usuelle en mathématiques.*

— *Enfin l'équivalence de deux formules est systématiquement notée avec le symbole $=$ au lieu du symbole $\equiv$.*

Il est important de noter que la logique propositionnelle n'est pas l'unique algèbre de Boole. Par exemple, l'ensemble $\mathcal{P}(X)$ des sous-ensembles d'un ensemble X est une algèbre de Boole, comme nous l'indique la mise en correspondance des opérateurs ensemblistes avec leur désignation dans l'algèbre de Boole dans la figure 1.2.

Algèbre de Boole	$\mathcal{P}(X)$
1	X
0	$\emptyset$
$\overline{p}$	$X-p$
$p+q$	$p \cup q$
$p.q$	$p \cap q$

FIGURE 1.2 – Correspondance entre l'algèbre de Boole et $\mathcal{P}(X)$.

Puisque les lois de simplification sont déduites des lois de l'algèbre de Boole, comme nous allons le montrer dans la suite, elles sont vérifiées dans toute algèbre de Boole. En particulier dans l'algèbre de Boole $\mathcal{P}(X)$ nous avons $A \cup (A \cap B) = A$. Une *preuve dans l'algèbre de Boole* est une suite d'égalités, permettant de passer d'une formule à une autre formule équivalente en utilisant les axiomes ou simplifications de l'algèbre de Boole.

1.5.2 Propriétés

Nous présentons certaines des propriétés qui découlent de la définition de l'algèbre de Boole et qui sont couramment utilisées dans les raisonnements. Pour raccourcir les preuves de ces propriétés, nous utilisons implicitement l'associativité et la commutativité de la somme et du produit.

Propriété 1.5.3 *Dans une algèbre de Boole, pour tout x, il y a un et un seul y tel que $x+y=1$ et $xy=0$, autrement dit la négation est unique.*

Preuve : Il y a un élément y vérifiant $x+y=1$ et $xy=0$, c'est $\overline{x}$ d'après les axiomes de l'algèbre de Boole. Montrons qu'il est unique. Raisonnons par contradiction, supposons qu'il n'est pas unique, nous avons donc qu'il existe y et z deux éléments distincts tels que $x+y=1$, $xy=0$, $x+z=1$, et $xz=0$. Nous considérons $u=xy+z$. Puisque $xy=0$, nous avons : $u=z$. Par distributivité de la somme sur le produit, nous avons : $u=(x+z)(y+z)$. Puisque $x+z=1$ et que 1 est l'élément neutre du produit, nous avons : $u=y+z$. Nous concluons donc que $z=y+z$. En considérant $v=xz+y$ et échangeant les rôles de y et z, nous obtenons de façon analogue $y=y+z$. Par suite $y=z$, ce qui prouve donc l'unicité de la négation. □

Propriété 1.5.4 *Dans une algèbre de Boole, les équivalences suivantes sont vraies pour tout x et tout y :*

1. $\overline{1}=0$.
2. $\overline{0}=1$.
3. $\overline{\overline{x}}=x$.
4. *Idempotence du produit :* $x.x=x$.
5. *Idempotence de la somme :* $x+x=x$.
6. 1 *est élément absorbant de la somme :* $1+x=1$.
7. 0 *est élément absorbant du produit :* $0.x=0$.
8. *Lois de De Morgan :*
   — $\overline{xy}=\overline{x}+\overline{y}$.
   — $\overline{x+y}=\overline{x}.\overline{y}$.

Preuve :

1. $\overline{1}=0$.

2. $\overline{0}=1$.

3. $\overline{\overline{x}}=x$.

4. Idempotence du produit : $x.x=x$.

5. Idempotence de la somme : $x+x=x$.

6. 1 est élément absorbant de la somme : $1+x=1$.

7. 0 est élément absorbant du produit : $0.x=0$.

8. Lois de De Morgan :

— $\overline{x.y}=\overline{x}+\overline{y}$:

— $\overline{x+y} = \bar{x}.\bar{y}$.

□

1.6 Fonctions booléennes

Nous regardons maintenant nos formules comme des fonctions mathématiques sur le domaine $\mathbb{B} = \{0,1\}$. Nous définissons formellement ce qu'est une fonction booléenne, puis nous revisitons les formes normales conjonctives et disjonctives avec ce formalisme. Remarquons que les fonctions booléennes sont des notions utilisées en cryptologie par exemple dans la construction de chiffrements symétriques (le chiffrement One-Time-Pad, un des chiffrements les plus sûrs est basé sur la fonction booléenne du « ou-exclusif »).

Définition 1.6.1 (Fonction booléenne) *Une* fonction booléenne f *est une fonction dont les arguments et le résultat sont dans le domaine* $\mathbb{B} = \{0,1\}$.

$$f : \mathbb{B}^n \to \mathbb{B}$$

Exemple 1.6.2 *Nous donnons des exemples et contre-exemples de fonctions booléennes :*

— *La fonction* $f : \mathbb{B} \to \mathbb{B} : f(x) = \neg x$ *est une fonction booléenne.*
— *La fonction* $f : \mathbb{N} \to \mathbb{B} : f(x) = x \bmod 2$ *n'est pas une fonction booléenne.*
— *La fonction* $f : \mathbb{B} \to \mathbb{N} : f(x) = x+1$ *n'est pas une fonction booléenne.*
— *La fonction* $f : \mathbb{B} \times \mathbb{B} \to \mathbb{B} : f(x,y) = \neg(x \wedge y)$ *est une fonction booléenne.*

1.6.1 Fonctions booléennes et somme de monômes

Grâce aux fonctions booléennes, nous sommes capables à partir des valeurs de vérité d'une formule de reconstruire la somme de monômes correspondante. Pour toute variable x, nous posons $x^0 = \overline{x}$ et $x^1 = x$.

Théorème 1.6.3 *Soit* f *une fonction booléenne à* n *arguments. Cette fonction est représentée à l'aide de* n *variables* $x_1, \dots, x_n$. *Soit* A *la formule suivante :*

$$A = \sum_{f(a_1,\dots,a_n)=1} x_1^{a_1} \dots x_n^{a_n}.$$

Les a_i *sont des valeurs booléennes et* A *est la somme des monômes* $x_1^{a_1} \dots x_n^{a_n}$ *tels que* $f(a_1, \dots, a_n) = 1$. *Par convention, si la fonction* f *vaut toujours* 0 *alors* $A = 0$.

Pour toute assignation v *telle que* $v(x_1) = a_1, \dots, v(x_n) = a_n$, *nous avons* $f(a_1, \dots, a_n) = [A]_v$. *En omettant la distinction entre une variable et sa valeur, ce résultat s'écrit plus brièvement :* $f(x_1, \dots x_n) = A$.

Exemple 1.6.4 *La fonction* maj *à 3 arguments vaut* 1 *lorsqu'au moins 2 de ses arguments valent* 1. *Nous donnons la table représentant cette fonction :*

x_1	x_2	x_3	$maj(x_1,x_2,x_3)$					
0	0	0	0	0	0	0	0	0
0	0	1	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0
0	1	1	**1**	**1**	0	0	0	**1**
1	0	0	0	0	0	0	0	0
1	0	1	**1**	0	**1**	0	0	**1**
1	1	0	**1**	0	0	**1**	0	**1**
1	1	1	**1**	0	0	0	**1**	**1**

Nous notons sur la table de vérité, que chaque monôme ne vaut 1 *que sur une ligne de la table et que la somme des monômes est équivalente à* $maj(x_1,x_2,x_3)$. *La fonction* maj *peut donc s'exprimer par*

1.6.2 Fonctions booléennes et produit de clauses

Nous effectuons le même raisonnement pour la forme duale.

Théorème 1.6.5 *Soit f une fonction booléenne à n arguments. Cette fonction est représentée à l'aide de n variables $x_1,\ldots,x_n$. Soit A la formule suivante :*

$$A = \prod_{f(a_1,\ldots,a_n)=0} x_1^{\overline{a_1}} + \ldots + x_n^{\overline{a_n}}.$$

Les a_i sont des valeurs booléennes et A est le produit des clauses $x_1^{\overline{a_1}} + \ldots + x_n^{\overline{a_n}}$ telles que $f(a_1,\ldots,a_n) = 0$. Par convention si la fonction f vaut toujours 1 alors $A = 1$.

Pour toute assignation v telle que $v(x_1) = a_1, \ldots, v(x_n) = a_n$, nous avons $f(a_1,\ldots,a_n) = [A]_v$. En omettant la distinction entre une variable et sa valeur, ce résultat s'écrit plus brièvement : $f(x_1,\ldots,x_n) = A$.

Exemple 1.6.6 *Nous représentons la fonction maj définie dans l'exemple 1.6.4 page précédente par un produit de clauses :*

x_1	x_2	x_3	$maj(x_1,x_2,x_3)$					
0	0	0	**0**	**0**	1	1	1	**0**
0	0	1	**0**	1	**0**	1	1	**0**
0	1	0	**0**	1	1	**0**	1	**0**
0	1	1	1	1	1	1	1	1
1	0	0	**0**	1	1	1	**0**	**0**
1	0	1	1	1	1	1	1	1
1	1	0	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1

La fonction maj peut donc s'exprimer en produit de clauses comme suit :

1.7 L'outil BDDC

`BDDC` (*Binary Decision Diagram based Calculator*) est un outil pour la manipulation de formules propositionnelles développé par Pascal Raymond et disponible à l'adresse suivante :

`http://www-verimag.imag.fr/~raymond/home/tools/bddc/.`

Cet outil est une « calculette propositionnelle » basée sur les *diagrammes de décision binaires*. Ces diagrammes permettent d'avoir une représentation interne des formules logiques sous la forme d'un graphe orienté acyclique ayant pour propriété d'être canonique. Cette calculette offre notamment la possibilité :

- d'évaluer si une formule est une tautologie,
- d'évaluer si une formule a un modèle,
- de transformer une formule en forme normale conjonctive,
- de transformer une formule en forme normale disjonctive,
- ...

Ainsi, il est possible d'utiliser `BDDC` pour résoudre plusieurs des exercices proposés ci-après, parmi lesquels les exercices 8 page 32 à 16 page 33 et 23 page 34 à 24 page 34.

1.8 Exercices

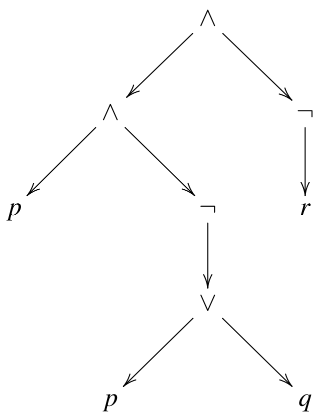
Exercice 1 (Formules strictes, formules à priorité) *Parmi les expressions suivantes, indiquer celles qui sont et celles qui ne sont pas des formules, au sens de la syntaxe stricte décrite dans la définition 1.1.1 page 12 :*

1. x
2. $(x$
3. (x)
4. $(x \Rightarrow (y \wedge z)$
5. $(x \Rightarrow (y \wedge z))$
6. $\neg(x \vee y)$
7. $(\neg(x \vee y))$
8. $\neg(x \Rightarrow y) \vee \neg(y \wedge z)$
9. $\neg(x)$
10. $(\neg((x \vee y) \wedge z) \Leftrightarrow (u \Rightarrow v)))$

$\square$

Exercice 2 (☞Formules et priorités) *Montrer que toute formule stricte est une formule à priorité.* $\square$

Exercice 3 (Formules et priorités) *L'arbre suivant représente la structure de la formule $((p \wedge \neg(p \vee q)) \wedge \neg r)$.*



En utilisant les priorités, elle peut s'écrire avec le moins de parenthèses possibles sous l'une des formes : $p \wedge \neg(p \vee q) \wedge \neg r$ avec les notations logiques, $p.\overline{p+q}.\overline{r}$ avec les notations booléennes. Comme sur cet exemple, pour chacune des formules suivantes, indiquer sa structure sous forme d'arbre, sa forme la moins parenthésée avec les opérations logiques, sa forme la moins parenthésée avec les opérations booléennes :

1. $(\neg(a \wedge b) \Leftrightarrow (\neg a \vee \neg b))$.
2. $((\neg a \vee b) \wedge (\neg b \vee a))$.
3. $(((a \wedge b) \wedge c) \vee ((\neg a \wedge \neg b) \wedge \neg c))$.
4. $(p \Rightarrow (q \Rightarrow r))$.
5. $((p \Rightarrow q) \Rightarrow r)$.

$\square$

Exercice 4 (Formules et priorités) *Indiquer sous forme d'arbres, les structures des formules à priorité suivantes :*

1. $p \Leftrightarrow \neg q \vee r$.
2. $p \vee q \Rightarrow r \wedge s$.
3. $p \vee q \Rightarrow r \Leftrightarrow s$.
4. $p \vee q \wedge r \Rightarrow \neg s$.
5. $p \Rightarrow r \wedge s \Rightarrow t$.

6. $p \vee q \wedge s \vee t$.
7. $p \wedge q \Leftrightarrow \neg r \vee s$.
8. $\neg p \wedge q \vee r \Rightarrow s \Leftrightarrow t$.

$\square$

Exercice 5 (Hauteur,*) *Nous définissons la hauteur h d'une formule à priorité par :*
— $h(\top) = 0$ *et* $h(\bot) = 0$.
— *Si A est une variable* $h(A) = 0$.
— $h(\neg A) = 1 + h(A)$.
— $h((A)) = h(A)$.
— $h(A \circ B) = \max(h(A), h(B)) + 1$, *si* $\circ$ *est une des opérations* $\vee, \wedge, \Rightarrow, \Leftrightarrow$.

1. *Montrer que cette définition est ambigüe, c'est-à-dire trouver une formule qui peut avoir deux hauteurs différentes.*
2. *Proposer une nouvelle définition récursive de la hauteur n'ayant pas ce problème.*

$\square$

Exercice 6 (Validité) *Donner les tables de vérités des formules suivantes :*

1. $p \Rightarrow (q \Rightarrow p)$.
2. $p \Rightarrow (q \Rightarrow r)$.
3. $(p \Rightarrow q) \Rightarrow (p \Rightarrow r)$.
4. $(p \Rightarrow (q \Rightarrow r)) \Rightarrow ((p \Rightarrow q) \Rightarrow (p \Rightarrow r))$.

Indiquer celles qui sont valides. Indiquer les équivalences. $\square$

Exercice 7 (☞Raisonnement circulaire) *Donner les tables de vérité des formules suivantes :* $a \Rightarrow b$, $b \Rightarrow c$, $c \Rightarrow a$, $a \Leftrightarrow b$, $b \Leftrightarrow c$ *et* $c \Leftrightarrow a$. *Conclure que* $(a \Rightarrow b) \wedge (b \Rightarrow c) \wedge (c \Rightarrow a) \equiv (a \Leftrightarrow b) \wedge (b \Leftrightarrow c) \wedge (c \Leftrightarrow a)$. $\square$

Exercice 8 (Équivalence) *Donner les tables de vérités des formules suivantes :*

1. $p \wedge (p \vee q)$.
2. $\neg p \wedge \neg q$.
3. $\neg(p \wedge q)$.
4. $\neg(p \vee q)$.
5. $p \vee (p \wedge q)$.
6. $\neg p \vee \neg q$.
7. p.

Parmi ces formules, indiquer les formules équivalentes . $\square$

Exercice 9 (Équivalence) *Donner les tables de vérités des formules suivantes :*

1. $(p \Rightarrow q) \wedge (q \Rightarrow p)$.
2. $p \Rightarrow q$.
3. $p \Leftrightarrow q$.
4. $\neg q \Rightarrow \neg p$.
5. $(p \wedge q) \vee (\neg p \wedge \neg q)$.

Parmi ces formules, Indiquer les formules équivalentes. $\square$

Exercice 10 (Équivalence) *Parmi les formules suivantes, indiquer celles qui sont équivalentes à* $p \Rightarrow q \vee r$ *?*

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

$\square$

Exercice 11 (☞Modèle d'un ensemble de formules,*) *Montrer qu'une assignation est un modèle d'un ensemble de formules, si et seulement si elle est un modèle de la conjonction des formules de l'ensemble.* □

Exercice 12 (☞Lois de simplification) *Prouver que pour tout* x, y *:*

— $x \vee (x \wedge y) \equiv x$.
— $x \wedge (x \vee y) \equiv x$.
— $x \vee (\neg x \wedge y) \equiv x \vee y$.
— $x \wedge (\neg x \vee y) \equiv x \wedge y$.

□

Exercice 13 (☞Simplification de formule,*) *Montrer par simplification que la formule suivante est une tautologie.*

$$(a \vee b) \wedge (\neg b \vee c) \Rightarrow (a \vee c)$$

□

Exercice 14 (Modèles et formes normales) *Soit A la formule suivante :*

$$(((a \Rightarrow \neg b) \Leftrightarrow \neg c) \wedge (c \vee d)) \wedge (a \Leftrightarrow d).$$

1. *A est-elle une tautologie ? (justifier)*
2. *A est-elle une contradiction ? (justifier)*
3. *Donner la forme normale conjonctive de A (détailler).*
4. *Donner la forme normale disjonctive de A (détailler).*

□

Exercice 15 (☞Récurrence,*) *Montrer que toute formule construite avec une seule variable, uniquement les connecteurs binaires* $\vee$ *et* $\wedge$ *et sans négation est équivalente à une formule de taille* 0. □

Exercice 16 (Algèbre de Boole) *À l'aide de tables de vérité, indiquer pour les opérations suivantes* $\Rightarrow, \Leftrightarrow$ *si elles sont commutatives, associatives, idempotentes, transitives.* □

Exercice 17 (Algèbre de Boole) *Montrer qu'une formule avec une seule variable, notée p, est équivalente à l'une des formules,* $0, 1, p, \neg p$. □

Exercice 18 (Algèbre de Boole) *Donner* 16 *formules telles que toute formule avec deux variables, notées p et q, soit équivalente à l'une des* 16 *formules.*

□

Exercice 19 (Fonction booléenne) *Indiquer en fonction de k combien il y a de fonctions booléennes à k arguments.* □

Exercice 20 (Conséquence) *Au cours de l'enquête dont il est chargé, l'adjudant Tifrice effectue le raisonnement suivant :*

— *Si le meurtre a eu lieu le jour, le meurtrier est l'ami de la victime*
— *Or le meurtre a eu lieu la nuit*

Donc le meurtrier n'est pas l'ami de la victime. Le raisonnement de l'adjudant Tifrice est-il correct ? Pour cela nous procédons en trois étapes :

1. *Formalisation des faits.*
2. *Formalisation du raisonnement permettant de déduire la conclusion à partir des hypothèses.*
3. *Démontrer que le raisonnement est correct ou non.*

□

Exercice 21 (Conséquence) *Pinocchio, Quasimodo et Roméo s'apprêtent à chanter en canon. Ils décident entre eux que :*

1. *Si Pinocchio ne chante pas alors Quasimodo chantera.*
2. *Si Quasimodo chante alors Pinocchio et Roméo chanteront.*
3. *Si Roméo chante alors Quasimodo ou Pinocchio, l'un des deux au moins, ne chantera pas.*

Peut-on en conclure que Pinocchio chantera ? Justifier votre réponse en formalisant le raisonnement. □

Exercice 22 (Conséquence) *Formaliser les énoncés suivants en dégageant les connecteurs logiques.*

(a) *Si Pierre est rentré chez lui, alors Jean est allé au cinéma.*

(b) *Marie est à la bibliothèque ou Pierre est rentré chez lui ou les deux.*

(c) *Si Jean est allé au cinéma, alors Marie est à la bibliothèque ou Pierre est rentré chez lui ou les deux.*

(d) *Marie n'est pas à la bibliothèque et Jean est allé au cinéma.*

(e) *Pierre est rentré chez lui.*

Le dernier énoncé est-il conséquence des énoncés qui le précèdent ? □

Exercice 23 (Formes normales) *Pour chaque formule donnée ci-après, donner sa forme normale disjonctive et prouver si elle est ou non satisfaisable (en donnant si besoin un modèle de la formule).*

— $\neg(a \Leftrightarrow b) \vee (b \wedge c) \Rightarrow c$.

— $(a \Rightarrow b) \wedge (b \Rightarrow \neg a) \wedge (\neg a \Rightarrow b) \wedge (b \Rightarrow a)$.

□

Exercice 24 (Formes normales) *Soit A la formule $p \vee (q \wedge r) \Leftrightarrow (p \vee q) \wedge (p \vee r)$. Transformer A en une fnc. La formule A est-elle valide ?*

□

Exercice 25 (Formes normales,)** *Nous revenons sur les preuves des transformations en forme normale.*

1. *Montrer que l'application de l'élimination des équivalences, des implications et de déplacement des négations sur une formule donne une formule en forme normale.*
2. *Montrer aussi que quel que soit l'ordre d'application de l'élimination des équivalences, des implications et de déplacement des négations sur une formule, l'algorithme de transformation en forme normale termine.*

□

Exercice 26 (Conséquence et formes normales) *L'adjudant Tifrice enquête une nouvelle fois. Ses hypothèses sont les suivantes :*

— *Si Jean n'a pas rencontré Pierre l'autre nuit, c'est que Pierre est le meurtrier ou que Jean est un menteur*

— *Si Pierre n'est pas le meurtrier, alors Jean n'a pas rencontré Pierre l'autre nuit et le crime a eu lieu après minuit*

— *Si le crime a eu lieu après minuit, alors Pierre est le meurtrier ou Jean est un menteur.*

Il en conclut que Pierre est le meurtrier. Son raisonnement est-il correct ? Donner la réponse en construisant la conjonction des hypothèses et de la négation de la conclusion *et en mettant cette conjonction sous forme de somme de monômes. Nous rappelons que le raisonnement est incorrect si et seulement si un des monômes de la somme ne comporte pas de littéraux complémentaires : ce monôme donne un modèle des hypothèses, qui est un contre-modèle de la conclusion.*

□

Exercice 27 (Formalisation, somme de monômes,*) *Nous nous trouvons sur une île dont les indigènes sont répartis en deux tribus, les Purs et les Pires. Les Purs disent toujours la vérité tandis que les Pires mentent toujours. Nous rencontrons deux indigènes, Aha et Bébé*[4].

(a) *Aha déclare : « L'un de nous au moins est un Pire ». Pouvons-nous en déduire ce que sont Aha et Bébé ?*

(b) *Aha déclare : « L'un de nous deux au plus est un Pire ». Pouvons-nous en déduire ce que sont Aha et Bébé ?*

(c) *Aha déclare :« Nous sommes tous les deux de la même tribu ». Pouvons-nous en déduire ce que sont Aha et Bébé ?*

□

Exercice 28 (Ensemble complet et incomplet de connecteurs,)** *Un ensemble de constantes et de connecteurs est dit* complet*, si toute fonction booléenne est exprimable avec ces constantes et ces connecteurs. D'après le théorème 1.6.3 page 29, l'ensemble $\{0, 1, -, +, \bullet\}$ est complet.*

1. *Montrer que l'ensemble $\{-, +\}$ est complet.*
   Indication : il suffit de montrer que $\{0, 1, \bullet\}$ sont définissables à l'aide de $-$ et $+$.

4. Cette énigme provient du livre de Raymond M. Smullyan : « Quel est le titre de ce livre ? », qui contient bien d'autres problèmes amusants à propos de ces indigènes.

2. *Montrer que l'ensemble* $\{0, \Rightarrow\}$ *est complet.*
3. *Soit* $|$ *l'opération suivante :* $x \mid y$ *est vrai si et seulement si ni l'un, ni l'autre ne sont vrais, autrement dit* $x \mid y = 1$ *si et seulement si* $x = 0$ *et* $y = 0$*. Cette opération est aussi appelée* ni *car* $x \mid y$ *est vrai si et seulement si ni x, ni y ne sont vrais.*
   Montrer que cette opération est complète.
4. *(**) Montrer que l'ensemble* $\{0, 1, +, \bullet\}$ *est incomplet.*
   Indication : nous trouvons une propriété caractéristique des fonctions booléennes définies avec ces opérations et qui n'est pas vérifiée par toute fonction booléenne.
   Nous appelons monotone, une fonction f telle que si $a_1 \leq b_1, \dots a_n \leq b_n$ *alors* $f(a_1, \dots a_n) \leq f(b_1, \dots b_n)$*.*
   Montrer que toute fonction booléenne définie avec $\{0, 1, +, \bullet\}$ *est monotone.*
   Proposer une fonction booléenne non monotone.
5. *(*) L'ensemble* $\{1, \Rightarrow\}$ *est-il complet ?*
6. *(**) L'ensemble* $\{0, \Leftrightarrow\}$ *est-il complet ?*

□

Exercice 29 (Dualité, récurrence,)** *Considérons des formules dont les opérations sont limitées à* $0, 1, \neg, \wedge, \vee$*. Nous rappelons que la formule duale d'une formule est obtenue en échangeant les conjonctions et les disjonctions ainsi que les* 0 *et les* 1 *et que nous notons* A^* *la duale de la formule A.*

1. *Montrer que si 2 formules sont équivalentes, il en est de même de leurs duales. Nous donnons une indication qui réduit la question à une récurrence simple. Soit v une assignation. Soit* $\overline{v}$ *l'assignation complémentaire de v : pour toute variable x,* $\overline{v}(x) = \overline{v(x)}$*. Par exemple l'assignation complémentaire de* $a = 1, b = 0$ *est* $a = 0, b = 1$*.*
   (a) *Montrer que pour toute formule A :* $[A^*]_v = \overline{[A]_{\overline{v}}}$*.*
   (b) *En déduire que si 2 formules sont équivalentes, il en est de même de leurs duales.*
2. *En déduire que si une formule est valide, sa duale est contradictoire.*

□

Exercice 30 (Représentation canonique,*)** *Soit* $x_{i(0 \leq i \leq n)}$ *une liste de n variables distinctes. Dans cet exercice, les variables des formules sont toutes dans cette liste. À cette liste et à une formule A, nous associons la fonction booléenne f à n arguments définie par* $f(x_1, \dots, x_n) = A$*. Cette notation signifie que pour* $a_1, \dots a_n \in \mathbb{B}$*, la valeur de* $f(a_1, \dots, a_n)$ *est la valeur de la formule A quand* $x_1 = a_1, \dots, x_n = a_n$*.*

1. *Combien y a-t-il de classes de formules ?*
2. *Notons par* $\oplus$ *le ou-exclusif. Nous avons* $x \oplus y = 1$ *si et seulement soit* $x = 1$*, soit* $y = 1$ *mais pas* $x = y = 1$ *(d'où le nom de ou-exclusif). Nous pouvons aussi définir cette opération par* $x \oplus y = \neg(x \Leftrightarrow y)$*. Il est clair que cette opération est commutative, associative, que l'opération* $\wedge$ *est distributive sur* $\oplus$ *et que* 0 *est l'élément neutre de* $\oplus$*. Exprimer la négation, la disjonction, l'implication et l'équivalence comme somme ou-exclusive de* 1 *et de conjonctions de variables.*
3. *Montrer que toute formule booléenne est équivalente à une somme ou-exclusive de 1 et de produits de variables.*
4. *Par convention* 1 *est considéré comme le produit de zéro variable et* 0 *comme la somme ou-exclusive de zéro produit de variables. Montrer que toute formule booléenne est équivalente à une somme ou-exclusive de produits dont les ensembles de variables sont tous distincts.*
5. *Fixons un ordre arbitraire des n variables utilisées dans les formules (lorsque les variables sont des identificateurs, cet ordre est souvent l'ordre lexicographique). Une somme ou-exclusive de produits de variables est canonique (forme de Reed-Muller) si et seulement si :*
   — *ses produits ne comportent pas de variables répétées et dans un produit les variables sont ordonnées de gauche à droite dans l'ordre croissant,*
   — *les produits sont ordonnés de gauche à droite dans l'ordre lexicographique décroissant déduit de l'ordre des variables.*
   Montrer que toute formule booléenne est équivalente à une somme ou-exclusive canonique. En déduire l'unicité de cette forme canonique.

□

Chapitre 2

Résolution propositionnelle

Sommaire

LES tables de vérité et les transformations en sommes de monômes ou produits de clauses permettent de répondre aux questions : une formule est-elle valide ? et un raisonnement est-il correct ? Cependant le temps de réponse de ces méthodes augmente exponentiellement avec le nombre de variables. Pour illustrer cela, considérons la conséquence suivante :

$$a \Rightarrow b, b \Rightarrow c, c \Rightarrow d, d \Rightarrow e, e \Rightarrow f, f \Rightarrow g, g \Rightarrow h, h \Rightarrow i, i \Rightarrow j \models a \Rightarrow j$$

En utilisant une table de vérité il faut tester $2^{10} = 1024$ lignes. Or par *déduction*, en utilisant la transitivité de l'implication, nous savons immédiatement que le raisonnement ci-dessus est correct. Ainsi, en logique propositionnelle, les déductions peuvent s'avérer très utiles, voire nécessaires lorsque la taille du problème à traiter est importante. Nous étudions maintenant un système de déduction très simple. Dans ce système, les hypothèses et les formules déduites ne sont pas des formules quelconques mais des *clauses*, c'est-à-dire des disjonctions de littéraux. De plus, le système comporte une seule règle appelée *la résolution*.

Plan : Nous introduisons d'abord la méthode de *résolution* inventée en 1965 par J. Alan Robinson [23] dans le cadre plus général de la recherche de preuve au premier ordre. Nous présentons ensuite une procédure pour calculer tous les résolvants possibles, c'est-à-dire toutes les applications possibles de la règle de résolution : la *stratégie complète*. Puis, nous étudions l'algorithme `DPLL`, proposé dans les années soixante par Martin Davis, Hilary Putnam, George W. Logemann et Donald W. Loveland [10, 9]. En utilisant la notion de résolution, des méthodes de simplifications et le branchement/retour arrière, cet algorithme prouve efficacement si une formule a un modèle ou non. Bien qu'inventé il y a plus de quarante ans, il est toujours à la base de la plupart des solveurs SAT complets actuels. Notons que dans ce chapitre nous adoptons la notation booléenne plus concise.

2.1 Résolution

Nous souhaitons prouver que $a+c$ est une conséquence de $a+b$ et $\overline{b}+c$, c'est-à-dire $a+b,\ \overline{b}+c \models a+c$. En utilisant la propriété 1.2.27 page 17, il suffit de démontrer que la formule $(a+b).(\overline{b}+c) \Rightarrow (a+c)$ est valide. Pour cela, nous pouvons utiliser une table de vérité (8 lignes) ou alors une simplification de formules comme dans l'exercice 13 page 33. Ces deux méthodes sont fastidieuses même pour des formules aussi simples (3 variables uniquement). En utilisant la règle de résolution présentée dans ce chapitre, nous déduisons directement la clause $a+c$ à partir des deux clauses $a+b$ et $\overline{b}+c$.

Nous donnons maintenant la définition de la résolution et certaines propriétés utiles qui en dérivent. Nous prouvons ensuite la *cohérence* et la *complétude pour la réfutation* de notre système basé sur la déduction. La cohérence nous permet d'affirmer que si une clause C est obtenue par résolution à partir d'un ensemble Γ de clauses alors elle en est conséquence, c'est-à-dire tout modèle de Γ est modèle de C. La complétude pour la réfutation nous permet d'affirmer que si $\perp$ est conséquence de Γ, autrement dit si Γ est insatisfaisable, alors $\perp$ peut être retrouvé par résolution à partir de Γ.

2.1.1 Définitions

Dans un premier temps nous introduisons les notions nécessaires à la définition de la résolution. Nous rappelons que les notions de clauses et littéraux sont définies dans la section 1.4 du chapitre précédent (page 22). Comme nous le verrons dans la suite deux clauses qui ont le même ensemble de littéraux jouent le même rôle dans la résolution. Aussi, nous définissons deux clauses égales comme étant deux clauses avec le même ensemble de littéraux.

Définition 2.1.1 (Éléments d'une clause, clauses égales et sous-clauses)
— *Un littéral est* élément d'une clause*, s'il est élément de l'ensemble des littéraux de la clause.*
— *Une clause A est* incluse dans une clause B*, si tous les littéraux de la clause A sont éléments de la clause B. Dans ce cas, A est une* sous-clause *de B.*
— *Deux clauses sont* égales *si elles ont le même ensemble de littéraux.*

Exemple 2.1.2 *Nous illustrons les notions introduites par des exemples simples :*
— *Le littéral p est élément de la clause $\overline{q}+p+r+p$.*
— *La clause $p+\overline{q}$ est incluse dans la clause $\overline{q}+p+r+p$.*
— *Enlever le littéral p de la clause $\overline{q}+p+r+p$ donne la clause $\overline{q}+r$, enlever le littéral p de la clause $p+p+p$ donne la clause $\perp$.*
— *Ajouter le littéral r à la clause p donne la clause $p+r$, ajouter le littéral p à la clause $\perp$ donne la clause p.*
— *Les clauses $p+\overline{q}$, $\overline{q}+p$, et $p+\overline{q}+p$ sont égales.*

Notation : Nous notons $s(A)$ l'ensemble des littéraux de la clause A. Par convention $\perp$ est la clause vide et $s(\perp) = \emptyset$.

Exemple 2.1.3 $s(\overline{q}+p+r+p+\overline{p}) = \{\overline{q}, p, r, \overline{p}\}$.

Dans l'exercice 37 page 53 nous demandons de formaliser ce qu'est une clause et son ensemble de littéraux. Nous introduisons la notion de littéral *complémentaire*, noté L^c pour un littéral L, afin de manipuler dans nos raisonnements uniquement des littéraux. En effet, la négation ne suffit pas car $\overline{\overline{L}}$ est équivalent à L mais n'est pas un littéral : un littéral est une variable ou la négation d'une variable mais la double négation d'une variable n'est pas un littéral.

Définition 2.1.4 (Littéral complémentaire) *Notons L^c le* littéral complémentaire *d'un littéral L, défini par :*
— *Si L est une variable, L^c est la négation de L.*
— *Si L est la négation d'une variable, L^c est obtenue en enlevant la négation de L.*

Exemple 2.1.5 $x^c = \overline{x}$ *et* $\overline{x}^c = x$.

Nous présentons le calcul de résolvant entre deux clauses, ce qui constitue l'unique règle du système de résolution.

Définition 2.1.6 (Résolvant) *Soient A et B deux clauses. La clause C est un résolvant de A et B si et seulement s'il y a un littéral L tel que $L \in s(A)$, $L^c \in s(B)$ et $s(C) = (s(A) - \{L\}) \cup (s(B) - \{L^c\})$. Nous utilisons la représentation suivante lorsque C est un résolvant de A et B :*

$$\frac{A \quad B}{C}$$

Si la clause C est un résolvant de A et B, alors nous disons que C est engendrée *par A et B, et que A et B sont les* parents *de la clause C.*

Exemple 2.1.7 *Nous proposons quelques exemples de résolution :*

Propriété 2.1.8 *Si l'un des parents d'un résolvant est valide, le résolvant est valide ou contient l'autre parent.*

Preuve : Cette preuve est demandée dans l'exercice 39 page 53. □

Étant données deux clauses A et B, la formule $A+B$ n'est pas une clause si l'un des deux opérandes de la disjonction est la clause vide. Par exemple $\perp + p$ n'est pas une clause, bien que les deux opérandes de la disjonction soient des clauses. Aussi nous introduisons un nouvel opérande, $\tilde{+}$, afin de combiner la disjonction et l'élimination de $\perp$. Cet opérande va nous permettre de simplifier la définition de résolvant.

Définition 2.1.9 ($C\tilde{+}D$) *Soient C et D deux clauses. Nous notons $C\tilde{+}D$ la clause suivante :*

— *Si $C = \perp$ alors $C\tilde{+}D = D$,*
— *sinon si $D = \perp$ alors $C\tilde{+}D = C$ sinon $C\tilde{+}D = C+D$.*

Ajouter le littéral L à la clause C, c'est construire $C\tilde{+}L$. Munis de cette définition, nous pouvons reformuler plus simplement la règle de résolution.

Définition 2.1.10 (Résolvant) *Soient A et B deux clauses. La clause C est un* résolvant *de A et B si et seulement s'il y a un littéral L tel que :*

— *L est élément de la clause A, L^c est élément de la clause B*
— *C est égale à la clause $A'\tilde{+}B'$ où $A' = A - \{L\}$ est obtenue en enlevant L de A et $B' = B - \{L^c\}$ est obtenue en enlevant L^c de B.*

La combinaison de résolutions successives permet de générer de nouvelles clauses. Une clause issue de ces résolvants sera une preuve par résolution de cette clause à partir d'un ensemble de clauses.

Définition 2.1.11 (Preuve) *Soient Γ un ensemble de clauses et C une clause. Une* preuve *de C à partir de Γ est une liste de clauses qui se termine par C. Toute clause de la preuve est égale à un élément de Γ ou est un résolvant de deux clauses la précédant dans la preuve. La clause C est déduite de Γ, notée $\Gamma \vdash C$, s'il y a une preuve de C à partir de Γ.*

Exemple 2.1.12 *Soit Γ l'ensemble de clauses $\overline{p}+q, p+\overline{q}, \overline{p}+\overline{q}, p+q$. Nous montrons que $\Gamma \vdash \perp$ par la preuve suivante :*

Nous pouvons aussi voir une preuve comme un *arbre* dont les feuilles sont les hypothèses et dont les sommets internes sont obtenus par construction des résolvants.

Définition 2.1.13 (Taille de preuve) *Une preuve P de C à partir d'un ensemble de clauses Γ est de taille n si elle contient n lignes.*

La taille de la preuve donnée dans l'example 2.1.12 page précédente est 8, ce qui n'est pas forcément intuitif sur la représentation en arbre. Nous utiliserons les deux propriétés énoncées ci-dessous souvent de manière implicite dans les preuves.

Propriété 2.1.14 (Monotonie et composition) *Soient Γ, Δ deux ensembles de clauses et A, B deux clauses.*

1. *Monotonie de la déduction : Si $\Gamma \vdash A$ et si Γ est inclus dans Δ alors $\Delta \vdash A$*
2. *Composition des déductions : Si $\Gamma \vdash A$, $\Gamma \vdash B$ et si C est un résolvant de A et B alors $\Gamma \vdash C$.*

Preuve : Demandée dans l'exercice 38 page 53. □

2.1.2 Cohérence

La cohérence de la résolution signifie que si une clause C est obtenue après une à plusieurs applications de la règle de résolution à un ensemble de clauses Γ alors tout modèle de Γ est aussi modèle de C (autrement dit, C est une conséquence de Γ). La première partie de la preuve consiste à démontrer la cohérence d'une application de la règle de résolution (théorème 2.1.15). Nous généralisons ensuite par induction dans le théorème 2.1.16 page suivante.

Théorème 2.1.15 (Cohérence de la règle de résolution) *Si C est un résolvant de A et B alors $A, B \models C$.*

Preuve : Supposons que C soit un résolvant de A et B. Par définition il y a un littéral L tel que $L \in s(A), L^c \in s(B), s(C) = (s(A) - \{L\}) \cup (s(B) - \{L^c\})$. Soit v une assignation modèle de A et B, c'est-à-dire, $[A]_v = 1$ et $[B]_v = 1$. Montrons que v est modèle de C. Nous distinguons deux cas possibles :

□

Théorème 2.1.16 (Cohérence de la déduction) *Soient Γ un ensemble de clauses et C une clause. Si $\Gamma \vdash C$ alors $\Gamma \models C$.*

Preuve : Supposons que $\Gamma \vdash C$. Par définition, il existe une preuve P de C à partir de Γ. Supposons que pour toute preuve de D à partir de Γ, *plus courte* que P, nous avons $\Gamma \models D$. Montrons que $\Gamma \models C$. Nous avons deux cas possibles :

□

Corollaire 2.1.17 *D'après le théorème 2.1.16, si $\Gamma \vdash \perp$ alors $\Gamma \models \perp$, autrement dit Γ est insatisfaisable.*

2.1.3 Complétude pour la réfutation

La complétude pour la réfutation est la propriété suivante : si Γ est un ensemble insatisfaisable de clauses, alors il y a une preuve de la clause vide à partir de Γ. Nous montrons dans la suite de cette section la complétude pour la réfutation lorsque Γ est un ensemble *fini* de clauses.

Nous prouvons la complétude réfutationnelle par récurrence sur le nombre de variables. Pour cela, nous construisons à partir de Γ deux ensembles de clauses contenant tous deux une variable en moins. Plus précisément, nous construisons les ensembles $\Gamma[L := 1]$ et $\Gamma[L := 0]$ en assignant un littéral L de Γ à vrai et à faux, respectivement. Nous étudions ensuite les propriétés de ces deux ensembles par rapport à Γ (lemmes 2.1.22 page suivante et 2.1.23 page suivante). Ces propriétés seront utilisées lors du pas de récurrence.

Définition 2.1.18 ($\Gamma[L := 1]$) *Soient Γ un ensemble de clauses et L un littéral. $\Gamma[L := 1]$ est l'ensemble de clauses obtenu en supprimant les clauses dont L est élément et en enlevant L^c des autres clauses. Nous posons $\Gamma[L := 0] = \Gamma[L^c := 1]$.*

$\Gamma[L := x]$ est égal à l'ensemble de clauses obtenu en donnant à L la valeur x ($x = 0$ ou $x = 1$) et en simplifiant le résultat obtenu.

Exemple 2.1.19 *Soit Γ l'ensemble de clauses $\overline{p}+q, \overline{q}+r, p+q, p+r$. Nous avons :*
- $\Gamma[p := 1] =$
- $\Gamma[p := 0] =$

Afin de donner une intuition de comment calculer $\Gamma[p := 1]$ et $\Gamma[p := 0]$, nous considérons le produit de clauses suivant $(\overline{p}+q)(\overline{q}+r)(p+q)(p+r)$ et observons :
- $(\overline{1}+q)(\overline{q}+r)(1+q)(1+r) =$
- $(\overline{0}+q)(\overline{q}+r)(0+q)(0+r) =$

Nous notons $v[L := 1]$ l'assignation qui donne à L la valeur 1, à L^c la valeur 0 et ne change pas la valeur des autres littéraux.

Définition 2.1.20 ($v[L := 1]$) *Soit une assignation v, l'assignation $v[L := 1]$ est l'assignation identique à v sauf éventuellement pour x la variable de L. Si $L = x$ alors $v[L := 1](x) = 1$, si $L = \overline{x}$ alors $v[L := 1](x) = 0$.*

Nous posons $v[L := 0] = v[L^c := 1]$. La propriété suivante se déduit du fait que si Γ a un modèle v, alors v donne soit la valeur 1 soit la valeur 0 au littéral L.

Propriété 2.1.21 *Soient Γ un ensemble de clauses et L un littéral. Γ a un modèle si et seulement si $\Gamma[L := 1]$ ou $\Gamma[L := 0]$ a un modèle.*

Lemme 2.1.22 *Soient Γ un ensemble de clauses, C une clause et L un littéral. Si $\Gamma[L := 1] \vdash C$ alors $\Gamma \vdash C$ ou $\Gamma \vdash C \tilde{+} L^c$.*

Rappel : L'opération $\tilde{+}$ est introduite dans la définition 2.1.9 page 39.

Lemme 2.1.23 *Soient Γ un ensemble de clauses, C une clause et L un littéral. Si $\Gamma[L := 0] \vdash C$ alors $\Gamma \vdash C$ ou $\Gamma \vdash C \tilde{+} L$.*

Preuve : Supposons $\Gamma[L := 0] \vdash C$. Puisque $\Gamma[L := 0] = \Gamma[L^c := 1]$ et que $L^{cc} = L$, d'après le lemme 2.1.22 nous avons $\Gamma \vdash C$ ou $\Gamma \vdash C \tilde{+} L$. □

Théorème 2.1.24 *Soit Γ un ensemble fini de clauses. Si Γ est insatisfaisable alors $\Gamma \vdash \perp$.*

Preuve : Supposons que Γ est insatisfaisable. Soit n le nombre de variables de Γ. Nous montrons que $\Gamma \vdash \perp$ par récurrence sur n. Supposons que pour tout ensemble Δ de clauses insatisfaisable avec moins de n variables, nous avons $\Delta \vdash \perp$.

- **Cas de base :** Supposons que n soit nul. Donc $\Gamma = \emptyset$ ou $\Gamma = \{\perp\}$. Le premier cas est impossible, car l'ensemble vide est valide (toute assignation en est modèle). Donc $\Gamma = \{\perp\}$ et par suite $\Gamma \vdash \perp$.
- **Induction :** Supposons que n soit non nul. Soit x une variable figurant dans Γ. D'après la propriété 2.1.21, $\Gamma[x := 0]$ et $\Gamma[x := 1]$ sont insatisfaisables. Puisque la variable x ne figure pas dans ces deux ensembles de clauses, l'hypothèse de récurrence s'applique, donc : $\Gamma[x := 0] \vdash \perp$ et $\Gamma[x := 1] \vdash \perp$. Des lemmes 2.1.22 et 2.1.23, nous déduisons soit $\Gamma \vdash \perp$, soit $\Gamma \vdash \overline{x}$ et $\Gamma \vdash x$. Dans le premier cas, la preuve est terminée. Dans le deuxième cas, puisque $\perp$ est un résolvant de $\overline{x}$ et x, nous avons également $\Gamma \vdash \perp$.

□

Corollaire 2.1.25 *Soit Γ un ensemble fini de clauses. Γ est insatisfaisable si et seulement si $\Gamma \vdash \perp$.*

Preuve : Ce corollaire est une conséquence immédiate du lemme ci-dessus et de la remarque faite au début du paragraphe 2.1.3 page précédente. □

2.1.4 Réduction d'un ensemble de clauses

Réduire un ensemble de clauses, c'est lui enlever les clauses valides et les clauses contenant une *autre* clause de l'ensemble. Un ensemble de clauses est réduit s'il n'est plus réductible.

Définition 2.1.26 (Ensemble de clauses réduit) *Un ensemble de clauses est* réduit *s'il ne contient aucune clause valide et aucune clause n'est incluse dans une autre clause.*

Exemple 2.1.27 *La réduction de l'ensemble de clauses $\{p + q + \overline{p}, p + r, p + r + \overline{s}, r + q\}$ donne l'ensemble réduit :*

Propriété 2.1.28 *Un ensemble de clauses E est équivalent à l'ensemble de clauses réduit obtenu à partir de E.*

Preuve :

□

2.2 Stratégie complète

Pour savoir si un ensemble fini Γ de clauses est contradictoire, il y a une méthode simple mais peu efficace : construire *toutes* les clauses que nous pouvons déduire de Γ. D'après le corollaire 2.1.25 page ci-contre, Γ est contradictoire si et seulement si la clause vide a été construite. Il faut préciser ce que nous entendons par construire toutes les clauses déduites de Γ. Si nous considérons comme différentes les clauses qui s'écrivent différemment, deux clauses ont une infinité de résolvants. Par exemple les clauses $p+\overline{q}$ et $r+q$ ont comme résolvants la clause $p+r$ mais aussi les clauses $r+p$, $p+r+p$, $p+\ldots+p+r+\cdots+r$. Donc l'ensemble des clauses déduites de Γ est infini. Aussi, il faut, comme précédemment, considérer comme égales deux clauses, qui ont le même ensemble de littéraux (voir définition 2.1.1 page 38) : avec cette *définition de l'égalité de deux clauses*, l'ensemble des résolvants de deux clauses est fini ainsi que l'ensemble des clauses déduites de Γ. Supposons que l'ensemble des littéraux de Γ ait n éléments, comme les clauses déduites de Γ ne comportent que des littéraux de Γ (voir exercice 40 page 53), il y a donc au plus 2^n clauses déduites de Γ. Pour simplifier la suite, comme ci-dessus, nous confondons une clause et son ensemble de littéraux.

2.2.1 Algorithme de la stratégie complète

Soit Γ un ensemble de clauses. Pour obtenir l'ensemble des clauses minimales déduites de Γ, on construit deux suites d'ensembles de clauses $\Delta_{i(i\geq 0)}$ et $\Theta_{i(i\geq 0)}$ en s'arrêtant au premier indice k tel que $\Delta_k = \emptyset$.

1. Δ_0 est obtenu en réduisant Γ et Θ_0 est l'ensemble vide.
2. Δ_{i+1} est obtenu de la façon suivante
   (a) Nous construisons l'ensemble des résolvants entre les clauses de Δ_i et celles de $\Delta_i \cup \Theta_i$ puis nous réduisons cet ensemble.
   (b) Nous enlevons de cet ensemble de résolvants, les clauses qui contiennent une clause de $\Delta_i \cup \Theta_i$.
3. Θ_{i+1} est obtenu en enlevant de l'ensemble $\Delta_i \cup \Theta_i$ les clauses qui contiennent une clause de Δ_{i+1}.

Nous illustrons notre algorithme par deux exemples.

Exemple 2.2.1 *Nous appliquons l'algorithme à l'ensemble de clauses $a+b+\overline{a}, a+b, a+b+c, a+\overline{b}, \overline{a}+b, \overline{a}+\overline{b}$. L'algorithme effectue alors le calcul suivant. L'ensemble Δ_0 comprend les clauses :*

1	$a+b$
2	$a+\overline{b}$
3	$\overline{a}+b$
4	$\overline{a}+\overline{b}$

L'ensemble Θ_0 est vide. L'algorithme calcule les résolvants non valides suivants :

5	a	*résolvant de 1 et 2*
6	b	*résolvant de 1 et 3*
7	$\overline{b}$	*résolvant de 2 et 4*
8	$\overline{a}$	*résolvant de 3 et 4*

Donc l'ensemble Δ_1 comprend les mêmes clauses que nous numérotons de 5 *à* 8*.*

L'ensemble Θ_1 est vide.

L'ensemble Δ_2 comprend la clause :

9	$\perp$	*de 5 et 8*

Car l'algorithme obtient deux fois $\perp$.

L'ensemble Θ_2 est vide.

L'ensemble Δ_3 est vide.

L'ensemble Θ_3 est égal à Δ_2.

Nous notons que l'algorithme a construit la preuve suivante :

1	$a+b$	
2	$a+\overline{b}$	
5	a	*résolvant de 1 et 2*
3	$\overline{a}+b$	
4	$\overline{a}+\overline{b}$	
8	$\overline{a}$	*résolvant de 3 et 4*
9	$\perp$	*résolvant de 5 et 8*

Nous pouvons utiliser la représentation ci-dessous sous forme de tableau représentant la construction de ces ensembles.

k	Δ_k	Θ_k	$\Delta_k \cup \Theta_k$	*Résolvants de Δ_k et $\Delta_k \cup \Theta_k$*
0				
1				
2				
3				

Comme le montre cet exemple, les clauses de Δ_i ont des preuves de hauteur i, et celles de Θ_i des preuves de hauteur inférieure à i.

Exemple 2.2.2 *L'application de la stratégie complète sur l'ensemble $\{a, c, \overline{a}+\overline{b}, \overline{c}+e\}$ nous donne l'ensemble $\{a, c, e, \overline{b}\}$.*

k	Δ_k	Θ_k	$\Delta_k \cup \Theta_k$	*Résolvants de Δ_k et $\Delta_k \cup \Theta_k$*
0				
1				
2				

Nous montrons que l'algorithme se termine et qu'il est correct, c'est-à-dire que l'ensemble Θ_k est l'ensemble des clauses minimales déduites de Γ. Γ est insatisfaisable si et seulement si la clause vide est élément de Θ_k.

2.2.2 Arrêt de l'algorithme

Soit n le nombre de littéraux de Γ. Les clauses éléments des Δ_i sont des clauses déduites de Γ, donc, comme nous l'avons rappelé ci-dessus, elles ne comportent que des littéraux de Γ. Par suite, l'ensemble $\bigcup_{i<k} \Delta_i$ comprend au plus 2^n clauses. D'après l'algorithme, les ensembles Δ_i sont non vides pour $i < k$. Nous montrons ci-dessous que ces ensembles sont disjoints. Par conséquent $k \leq 2^n + 1$, autrement dit l'algorithme s'arrête.

Propriété 2.2.3 *Soit $i \leq k$. Toute clause de $\bigcup_{j \leq i} \Delta_j$ contient une clause de $\Delta_i \cup \Theta_i$.*

Preuve : Nous effectuons une preuve par induction.
— Pour $i = 0$ la propriété est triviale car $\bigcup_{j \leq 0} \Delta_j = \Delta_0$ et $\Delta_i \cup \Theta_i = \Delta_0$ ($\Theta_0 = \emptyset$).
— Supposons la propriété vraie au rang i et montrons qu'elle reste vraie au rang $i+1$. Soit $C \in \bigcup_{j \leq i+1} \Delta_j$. Montrons que C contient une clause de $\Delta_{i+1} \cup \Theta_{i+1}$. Nous examinons tous les cas possibles pour C.
  1. $C \in \Delta_{i+1}$. Donc C contient une clause de $\Delta_{i+1} \cup \Theta_{i+1}$.
  2. $C \in \bigcup_{j \leq i} \Delta_j$. Par hypothèse de récurrence, C contient une clause $D \in \Delta_i \cup \Theta_i$. Nous distinguons deux situations pour D.
     (a) $D \in \Theta_{i+1}$. Donc C contient une clause de $\Delta_{i+1} \cup \Theta_{i+1}$.
     (b) $D \notin \Theta_{i+1}$. Par construction de Θ_{i+1}, puisque $D \in \Delta_i \cup \Theta_i$ et que $D \notin \Theta_{i+1}$, c'est que D contient une clause de Δ_{i+1}. Or C contient D, donc C contient aussi une clause de $\Delta_{i+1} \cup \Theta_{i+1}$.

□

Propriété 2.2.4 *Pour tout $i \leq k$, les ensembles Δ_i sont disjoints entre eux.*

Preuve : Nous effectuons une récurrence sur les ensembles Δ_j avec $0 \leq j \leq i$ et $i \leq k$.

Cas de base : Si $i = 0$, il n'y a qu'un seul ensemble, donc la propriété est vérifiée.

Récurrence : Soit $i < k$. Supposons que tous les ensembles Δ_j où $j \leq i$ sont disjoints entre eux.
Montrons que Δ_{i+1} est disjoint de ces ensembles.
Soit $C \in \Delta_{i+1}$. Supposons, au contraire, que $C \in \bigcup_{j \leq i} \Delta_j$. D'après la propriété précédente, C inclut une clause de $\Delta_i \cup \Theta_i$. Donc par construction de Δ_{i+1}, $C \notin \Delta_{i+1}$, contradiction. Par suite, $C \notin \bigcup_{j \leq i} \Delta_j$.

□

2.2.3 Le résultat de l'algorithme est équivalent à l'ensemble initial de clauses

Nous montrons ci-dessous que l'ensemble Θ_k est équivalent à Γ.

Propriété 2.2.5 *Pour tout $i < k$, les ensembles $\Delta_i \cup \Theta_i$ et $\Delta_{i+1} \cup \Theta_{i+1}$ sont équivalents.*

Preuve :

1. Toute clause de $\Delta_{i+1} \cup \Theta_{i+1}$ est conséquence de $\Delta_i \cup \Theta_i$. En effet toute clause de $\Delta_{i+1} \cup \Theta_{i+1}$ est élément de $\Delta_i \cup \Theta_i$ ou résolvant de deux éléments de cet ensemble, donc est conséquence de cet ensemble.
2. Toute clause de $\Delta_i \cup \Theta_i$ est conséquence de $\Delta_{i+1} \cup \Theta_{i+1}$. Soit $C \in \Delta_i \cup \Theta_i$. Nous distinguons deux cas possibles :
   (a) $C \in \Theta_{i+1}$, ainsi C est conséquence de $\Delta_{i+1} \cup \Theta_{i+1}$.
   (b) $C \notin \Theta_{i+1}$, ainsi C contient une clause de Δ_{i+1} donc est conséquence de $\Delta_{i+1} \cup \Theta_{i+1}$.

□

Propriété 2.2.6 *Les ensembles Γ et Θ_k sont équivalents.*

Preuve : Par définition Δ_0 est l'ensemble obtenu par réduction de Γ, d'après la propriété 2.1.28 page 42, ces deux ensembles sont équivalents. Puisque Θ_0 est vide, Γ est équivalent à $\Delta_0 \cup \Theta_0$. D'après la propriété 2.2.5 et par récurrence, $\Delta_0 \cup \Theta_0$ est équivalent à l'ensemble de clauses $\Delta_k \cup \Theta_k$. Puisque l'algorithme termine quand Δ_k est l'ensemble vide, les ensembles Γ et Θ_k sont équivalents. □

2.3 Algorithme de Davis-Putnam-Logemann-Loveland

La méthode de Davis-Putnam-Logemann-Loveland (`DPLL`) fut crée en 1960 par Martin Davis et Hillary Putnam, puis améliorée en 1962 par Martin Davis, George Logemann et Donald Loveland. L'algorithme `DPLL` permet de décider si un ensemble de clauses est satisfaisable. Bien que cet algorithme date de plus de 50 ans, de nombreuses variantes en sont encore étudiées, car il est bien meilleur que ceux précédemment étudiés (table de vérité, transformation en somme de monômes ou produit de sommes, stratégie complète). Nous remarquons d'abord qu'il y a deux types de transformations de formules :

1. Celles qui préservent le sens, c'est-à-dire transforment une formule en une autre formule équivalente.
2. Celles qui préservent seulement la satisfaisabilité, c'est-à-dire transforment une formule satisfaisable en une autre formule satisfaisable.

L'efficacité de l'algorithme `DPLL` vient de l'utilisation des transformations préservant seulement la satisfaisabilité, car ces transformations sont moins coûteuses que celles préservant le sens. Nous introduisons d'abord la suppression des clauses contenant des littéraux isolés, puis la résolution unitaire et enfin la simplification des clauses valides. Ces transformations constituent les briques de base de l'algorithme `DPLL`. L'idée principale de cet algorithme, partant d'un ensemble de clauses « irréductibles », est de choisir une variable et de considérer les deux situations possibles : le cas où elle est vraie et le cas où elle est fausse. Ainsi les deux ensembles de clauses obtenus peuvent éventuellement se simplifier, sinon le choix d'une nouvelle variable est nécessaire. Dans le pire des cas il s'agit de construire la table de vérité correspondante aux clauses, mais en pratique de nombreuses simplifications apparaissent et permettent à cet algorithme de conclure efficacement.

2.3.1 Suppression des clauses contenant des littéraux isolés

Comme nous ne cherchons pas à préserver l'équivalence de formule mais la satisfaisabilité, nous avons la liberté de choisir la valeur de certaines variables. En particulier une variable qui n'apparaît que sous la forme d'un littéral positif ou négatif sera dite *isolée*. Nous pouvons alors affecter la valeur adéquate à cette variable afin de rendre vraies toutes les occurrences du littéral associé, ceci sans changer la satisfaisabilité de l'ensemble de clauses considéré.

Définition 2.3.1 (Littéral isolé) *Soient Γ un ensemble de clauses et L un littéral élément d'une clause de Γ, L est un* littéral isolé *(relativement à Γ), si aucune clause de Γ ne comporte de littéral complémentaire de L.*

Lemme 2.3.2 *La suppression des clauses qui comportent un littéral isolé, préserve la satisfaisabilité.*

La preuve est demandée dans l'exercice 49 page 54.

Exemple 2.3.3 *Soit Γ l'ensemble de clauses $\{p+q+r, \overline{q}+\overline{r}, q+s, \overline{s}+t\}$.*

2.3.2 Résolution unitaire

Définition 2.3.4 *Une* clause unitaire *est une clause qui ne comporte qu'un littéral.*

Les clauses unitaires sont aussi des clauses particulières qui sont :
— soit isolées et donc seront traitées par la simplification précédente,
— soit de potentiels résolvants, dans ce cas il faut effectuer la résolution dite « unitaire » de ces clauses.

Lemme 2.3.5 *Soit Γ un ensemble de clauses. Soit L l'ensemble des littéraux des clauses unitaires de Γ. Soit Θ l'ensemble de clauses ainsi obtenu à partir de Γ :*

— *Si L comporte deux littéraux complémentaires, alors $\Theta = \{\perp\}$.*
— *Sinon Θ est obtenu ainsi :*
  - *Nous enlevons les clauses qui comportent un élément de L.*
  - *À l'intérieur des clauses restantes nous enlevons les littéraux complémentaires des éléments de L.*

Γ a un modèle si et seulement si Θ en a un.

La preuve est demandée dans l'exercice 50 page 54.

Exemple 2.3.6

— *Soit Γ l'ensemble de clauses : $p+q, \overline{p}, \overline{q}$.*
— *Soit Γ l'ensemble de clauses : $a+b+\overline{d}, \overline{a}+c+\overline{d}, \overline{b}, d, \overline{c}$.*

— *Soit* Γ *l'ensemble de clauses :* $p, q, p+r, \overline{p}+r, q+\overline{r}, \overline{q}+s$.

2.3.3 Suppression des clauses valides

Enfin la dernière transformation utilisée dans l'algorithme `DPLL` est de simplement enlever les clauses valides d'un ensemble de clauses. Ceci peut paraître une évidence mais cette transformation est une des idées centrales de cet algorithme, une fois une variable affectée à une valeur, elle est implicitement utilisée dans la suppression de clauses contenant des littéraux isolés.

Lemme 2.3.7 *Soit* Γ *un ensemble de clauses. Soit* Θ *l'ensemble de clauses obtenu en supprimant les clauses valides de* Γ*.* Γ *a un modèle si et seulement si* Θ *en a un.*

Preuve :

— Supposons que Γ a un modèle v, comme Θ est un sous-ensemble des clauses de Γ, v est aussi modèle de Θ. Donc, Θ a un modèle.

— Supposons que Θ a un modèle v. Soit v' une assignation de Γ telle que $v'(x) = v(x)$ pour toute variable x présente à la fois dans Γ et Θ. Soit C une clause de Γ. Si C est aussi une clause de Θ, alors v' est un modèle de C car v et v' donnent la même valeur à C. Si C n'est pas une clause de Θ, alors C est valide, en conséquence toute assignation, v' en particulier, est modèle de C. D'où, Γ a un modèle : v'.

□

2.3.4 L'algorithme

Nous donnons une version de l'algorithme `DPLL` dans la figure 2.1 page suivante dans la fonction Algo_DPLL. Nous pouvons constater dans l'algorithme que nous supprimons une seule fois les clauses valides avant de réellement commencer les simplifications. Cela se justifie par le fait que l'algorithme ne pourra pas produire de nouvelles clauses valides. En effet, il ne fait qu'enlever des littéraux des clauses initialement données. Par suite, il est inutile d'enlever des clauses valides lors de l'étape de réduction, puisqu'il n'y aura plus de telles clauses. La fonction `DPLL` teste d'abord si la clause vide a été générée, puis effectue les simplifications possibles avant de choisir une variable pour appliquer récursivement la même fonction dans le cas où cette variable est affectée à vraie et dans le cas où elle est affectée à faux.

Pour obtenir une trace de l'algorithme, nous supprimons les clauses valides, en abrégé VAL, puis nous dessinons l'arbre des appels avec l'argument de la fonction ainsi que les ensembles obtenus par les étapes 2 (réduction, en abrégé RE), 3 (enlèvement des clauses ayant des littéraux isolés, en abrégé ELI) et 4 (résolution unitaire, abrégé en RU). À cause du « ou alors », il est inutile de poursuivre la construction de cet arbre dès que la réponse vrai (attachée à un ensemble vide) a été obtenue. Dans ce cas, un modèle peut être exhibé en remontant la récursion, en tenant compte des simplifications : le modèle doit rendre vrai chaque littéral isolé ou clause unitaire trouvé.

Exemple 2.3.8 *Nous illustrons l'application de cet algorithme sur deux ensembles de clauses.*

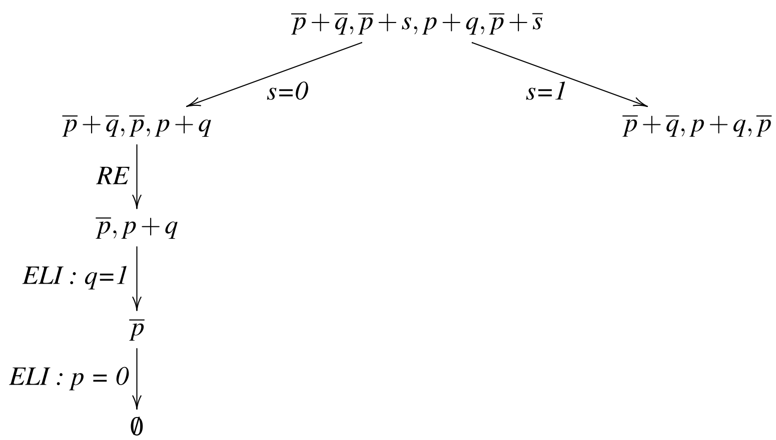
— *Soit* Γ *l'ensemble de clauses :* $\overline{a}+\overline{b}, a+b, \overline{a}+\overline{c}, a+c, \overline{b}+\overline{c}, b+c$. *Nous donnons la trace de l'algorithme.*

bool fonction Algo_DPLL(Γ : ensemble de clauses)
0 Supprimer les clauses valides de Γ.
Si $\Gamma = \emptyset$, retourner (`vrai`).
Sinon retourner (`DPLL`(Γ))

bool fonction `DPLL`(Γ : ensemble de clauses non valides)
La fonction retourne vrai si et seulement si Γ est satisfaisable
1 **Si** $\perp \in \Gamma$, retourner(`faux`).
Si $\Gamma = \emptyset$, retourner (`vrai`).
2 Réduire Γ : il suffit d'enlever toute clause contenant une *autre* clause.
3 Enlever de Γ les clauses comportant des littéraux isolés (cf. paragraphe 2.3.1 page 46).
Si l'ensemble Γ a été modifié, aller en 1.
4 Appliquer à Γ la résolution unitaire (cf paragraphe 2.3.2 page 46).
Si l'ensemble Γ a été modifié, aller en 1.
5 Choisir x une variable quelconque de Γ
retourner (`DPLL`($\Gamma[x := 0]$) ou alors `DPLL`($\Gamma[x := 1]$))

FIGURE 2.1 – Algorithme de Davis-Putnam-Logemann-Loveland.

— *Soit* Γ *l'ensemble de clauses :* $\overline{p}+\overline{q}, \overline{p}+s, p+q, \overline{p}+\overline{s}$. *Nous donnons la trace de l'algorithme.*



Puisqu'une feuille porte l'ensemble vide, l'ensemble Γ *est satisfaisable. Il est inutile de poursuivre la construction* *de la branche droite.*

Théorème 2.3.9 *L'algorithme Algo_DPLL est correct.*

Preuve : La réduction et les transformations 2.3.1 page 46, 2.3.2 page 46 et 2.3.3 page 47 préservent l'existence d'un modèle, donc au pas 0 et 1, l'algorithme vérifie l'invariant suivant : la valeur courante de l'ensemble de clauses de Γ a un modèle si et seulement si Γ a un modèle. Nous en déduisons immédiatement que les réponses de l'algorithme, fournies au pas 0 ou 1, sont correctes. Au pas 5, pour les mêmes raisons, la valeur courante de l'ensemble de clauses de Γ, lors de l'appel de `DPLL`, a un modèle si et seulement si Γ a un modèle. Supposons les appels récursifs corrects :

— Si `DPLL`$(\Gamma[x := 0])$ est vrai, par récurrence $\Gamma[x := 0]$ est satisfaisable donc Γ est satisfaisable (nous utilisons le résultat 2.1.21 page 42 : Γ est satisfaisable ssi $\Gamma[x := 0]$ est satisfaisable ou $\Gamma[x := 1]$ est satisfaisable), ce qui correspond à la valeur vrai de `DPLL`(Γ).

— Si `DPLL`$(\Gamma[x := 0])$ est faux, par récurrence $\Gamma[x := 0]$ est insatisfaisable. Dans ce cas, `DPLL`(Γ) vaut `DPLL`$(\Gamma[x := 1])$:
  — Supposons que `DPLL`$(\Gamma[x := 1])$ soit vrai alors par récurrence $\Gamma[x := 1]$ est satisfaisable donc Γ est satisfaisable, ce qui correspond à la valeur vrai de `DPLL`(Γ).
  — Supposons que `DPLL`$(\Gamma[x := 1])$ soit faux, alors par récurrence $\Gamma[x := 1]$ est insatisfaisable. Donc Γ est insatisfaisable, ce qui correspond à la valeur faux de `DPLL`(Γ).

□

Théorème 2.3.10 *L'algorithme Algo_DPLL se termine.*

Preuve : Au pas 0, la fonction Algo_DPLL consiste en un test, puis soit le renvoi de la valeur vrai soit l'appel de la fonction `DPLL`(Γ). Le pas 0 consiste en la suppression de clauses de l'ensemble de départ, cet ensemble étant fini, le pas 0 termine. Donc, la fonction Algo_DPLL termine si la fonction `DPLL`(Γ) termine.

Considérons un appel de la fonction `DPLL`(Γ). Le pas 1 est constitué d'un test et d'instructions élémentaires, donc il termine. Le pas 2 consiste à réduire Γ qui est un ensemble fini. La réduction consistant à supprimer des clauses, elle termine. Le pas 3 consiste à supprimer des clauses, puis éventuellement retourner au pas 1. Le nombre de clauses étant fini, soit l'algorithme termine en 1, soit le pas 4 débute. De la même manière, le pas 4 consiste à supprimer des clauses et des variables, qui sont en nombres finis, puis éventuellement retourner au pas 1. Le nombre de clauses et de variables étant fini, soit l'algorithme termine en 1, soit le pas 5 débute. Le pas 5 consiste en un ou deux appels récursifs sur un ensemble de clauses où une variable a disparu (la valeur d'une variable est fixée). Ainsi, la récurrence en 5 se termine car à chaque appel récursif le nombre de variables diminue strictement. D'où, l'algorithme Algo_DPLL termine. □

Remarque 2.3.11 (Oubli de simplifications) *L'algorithme Algo_DPLL a été donné avec les étapes de « simplification » : suppression des clauses valides (0), réduction (2), enlèvement des littéraux isolés (3) et réduction unitaire (4). L'algorithme reste correct si nous omettons ces étapes, ou si nous ne faisons qu'une partie des simplifications, ce qui est souvent le cas lorsque nous faisons une trace de l'exécution sans machine. Car sans machine il est fréquent d'oublier des simplifications, ce qui ne nuit pas car l'algorithme reste correct.*

Remarque 2.3.12 (Choix de la variable) *Un bon choix pour la variable x de l'étape (5), consiste à choisir la variable qui apparaît le plus souvent. Un meilleur choix consiste à choisir la variable qui va entraîner par la suite le plus de simplifications : il faut faire des compromis entre le temps passé à choisir la « bonne » variable et l'importance des simplifications induites par ce choix. Plusieurs heuristiques classiques de choix de variable sont présentées dans le paragraphe 2.3.5.1 page suivante.*

2.3.5 Solveurs SAT

Le problème *SAT* est un problème de décision qui consiste à déterminer si une formule propositionnelle en forme normale conjonctive admet ou non un modèle. Ce problème est le problème NP-complet de référence [5]. Généralement, seule une version de ce problème, appelée 3-SAT, où toutes les clauses de la formule sont constituées d'exactement trois littéraux, est considérée, car des *réductions linéaires* existent pour passer de SAT à 3-SAT. Un exemple de réduction SAT vers 3-SAT est étudié dans l'exercice 51 page 54.

Des programmes performants sont dédiés à la résolution du problème SAT [1]. Ces programmes sont appelés des *solveurs SAT* et sont généralement classés en deux catégories :

1. Dans ces programmes, le formalisme 3-SAT est généralement préféré au formalisme SAT, car son aspect régulier permet d'obtenir des solutions plus efficaces.

— Les solveurs SAT « complets » sont généralement des versions améliorées de l'algorithme DPLL. Parmi les solveurs SAT complets les plus utilisés à l'heure actuelle, nous pouvons citer par exemple *zchaff*, *satz*, *march* ou encore *GRASP*.
— Les solveurs SAT « incomplets » sont des algorithmes de semi-décision qui finissent par trouver un modèle à la formule, s'il existe et qui ne terminent pas dans le cas contraire. Ces algorithmes sont fondés sur des marches aléatoires dans l'espace d'état. Dans cette catégorie, nous pouvons citer les algorithmes par exemple *WalkSAT* et *GSAT*.

Nous décrivons maintenant, de manière non-exhaustive, les principales améliorations de l'algorithme DPLL utilisées dans les solveurs SAT complets.

2.3.5.1 Heuristique de branchement

Le choix de la prochaine variable à affecter a un impact important sur la taille de l'arbre de recherche développé par DPLL. Ainsi, le choix de l'heuristique de branchement est déterminant dans les algorithmes fondés sur DPLL. Les heuristiques de branchement sont souvent basées sur des arguments syntaxiques tels que la taille des clauses, le nombre d'occurences, *etc*. Parmi les heuristiques les plus connues, nous pouvons citer :

MOMS : L'heuristique *MOMS* (pour Maximum Occurences in Clauses of Minimum Size) consiste, comme son nom l'indique, à sélectionner la variable ayant le plus d'occurences dans les clauses les plus courtes. Les choix effectués par cette heuristique favorisent la propagation unitaire.

JW : L'heuristique *JW* (pour Jeroslow-Wang) utilise également la longueur des clauses. En effet, un poids est affecté à chaque littéral en fonction de la taille des clauses où il apparaît. La possibilité qu'une variable soit sélectionnée par JW est inversement proportionnelle à la somme des tailles des clauses où elle apparaît.

UP : Contrairement aux deux heuristiques précédentes, l'heuristique *UP* (pour Unit Propagation) n'est pas fondée sur des arguments syntaxiques. Cette heuristique essaie de sélectionner une variable en prévoyant à l'avance son effet sur la recherche. Elle consiste à calculer un poids pour chaque variable en fonction des simplifications qu'elle génère par propagation unitaire.

2.3.5.2 Ajout de clauses

L'idée est d'ajouter des clauses, en pré-traitement, à l'ensemble initial de clauses . Le nouvel ensemble de clauses reste équivalent à l'ensemble initial. Par exemple, nous pouvons citer la *résolution restreinte* qui consiste à ajouter des résolvants de taille bornée. Ce type de méthode permet de réduire la taille de l'arbre recherche en amenant aux contradictions plus tôt dans la recherche.

2.3.5.3 Analyse des conflits et retour-arrière non chronologique

Lorsque l'algorithme DPLL atteint une contradiction (un conflit), il revient (chronologiquement) au dernier branchement de variable effectué. Or, ce branchement n'est pas nécessairement l'une des décisions à l'origine de la contradiction. L'analyse des conflits remédie à ce problème en tenant compte des décisions à l'origine du conflit. Cette analyse permet ainsi d'effectuer un retour-arrière non-chronologique, c'est-à-dire un retour direct au niveau, plus haut dans l'arbre, qui est à l'origine du conflit sans tester les niveaux intermédiaires.

2.3.5.4 Apprentissage

L'apprentissage consiste à analyser et apprendre les raisons qui ont amené à une contradiction : lorsqu'une assignation partielle amène à une contraction, il est possible d'isoler un sous-ensemble d'assignations et un sous-ensemble de clauses, qui sont responsables du conflit. À partir de ces assignations, il est possible de construire (d'apprendre) une clause qui est impliquée par ces clauses. Cette nouvelle clause est ajoutée à l'ensemble de clauses du problème. Les assignations pertinentes sont déterminées par un graphe de dépendance entre les clauses et les assignations, appelé *graphe d'implication*. Les clauses apprises permettent de ne pas refaire plusieurs fois les mêmes « erreurs » dans l'arbre de recherche.

2.3.5.5 Redémarrage

La plupart des solveurs SAT actuels utilisent une stratégie de redémarrage, qui consiste à redémarrer la recherche en prenant en compte de nouvelles clauses apprises durant la recherche précédente. La plupart des solveurs effectuent un redémarrage après qu'un seuil de clauses apprises soit atteint. Ainsi, à chaque redémarrage, le parcours de l'espace de recherche change car l'ensemble de clauses est modifié. Il faut noter que la plupart des solveurs effacent régulièrement les clauses apprises. Ainsi, le mixage des redémarrages et des oublis peut dans certains cas mettre en cause la complétude des solveurs. Un paramétrage des redémarrages est donc nécessaire pour garder cette dernière propriété. Le solveur Chaff, par exemple, augmente à chaque redémarrage le seuil au-delà duquel une clause est oubliée. Ainsi, de plus en plus de clauses sont gardées, ce qui assure la complétude.

2.3.5.6 Structures de données paresseuses

Dans les solveurs SAT modernes, 80% du temps de calcul est consommé par la propagation unitaire. Ainsi, des structures de données dites « paresseuses » ont été introduites dans le but d'optimiser la propagation unitaire.

2.4 Exercices

Exercice 31 (Résolvant) *Par résolution, nous avons :*

$$\frac{a+b \quad \overline{a}+\overline{b}}{b+\overline{b}}$$

Montrer que : $b+\overline{b} \not\models (a+b).(\overline{a}+\overline{b})$.

□

Exercice 32 (Résolvant) *Nous rappelons que deux clauses sont égales si et seulement si elles ont les mêmes ensembles de littéraux.*

— *Les clauses* $p+q+\overline{r}+q+p+s+q+\overline{r}$ *et* $s+q+\overline{r}+p$ *sont-elles égales ?*

— *Les clauses* $p+q+r+p$ *et* $q+r+q+r+q+r$ *sont-elles égales ? L'une est-elle incluse dans l'autre ? L'une est-elle la conséquence de l'autre ? Sont-elles équivalentes ?*

— *Indiquer tous les résolvants des clauses* $a+\overline{b}+c$ *et* $a+b+\overline{c}$*. Ces résolvants sont-ils valides ?*

□

Exercice 33 (Preuve) *Les ensembles de formules suivants sont insatisfaisables.*

— $\{a, a \Rightarrow b, \overline{b}\}$.

— $\{a+b, \overline{a}+c, \overline{a}+\overline{d}, d+\overline{c}, \overline{b}+a\}$.

— $\{a+b+c, \overline{a}+b, \overline{b}+c, \overline{c}+a, \overline{a}+\overline{b}+\overline{c}\}$.

En donner une preuve par résolution.

□

Exercice 34 (Formalisation et résolution,*) *Remarquons que : « x à moins que y » se formalise en* $\neg(x \Leftrightarrow y)$*. Dans une maison hantée, les esprits se manifestent sous deux formes différentes, un* **chant obscène** *et un* **rire sardonique***, cependant nous pouvons influencer le comportement en jouant de l'orgue ou en brûlant de l'encens. Compte-tenu des données suivantes :*

(i) Le chant ne se fait pas entendre, à moins que nous jouions de l'orgue sans que le rire ne se fasse entendre.
(ii) Si nous brûlons de l'encens, le rire se fait entendre si et seulement si le chant se fait entendre.
(iii) (En ce moment) Le chant se fait entendre et le rire est silencieux.
Et de la conclusion :
(iv) (En ce moment) Nous jouons de l'orgue et ne brûlons pas d'encens.
Nous posons :

— *c : le chant se fait entendre.*

— *o : nous jouons de l'orgue.*

— *r : le rire se fait entendre.*

— *e : nous brûlons de l'encens.*

1. *Simplifier en produit de clauses* $\neg(x \Leftrightarrow y)$.
2. *Formaliser sous forme de produit de clauses les hypothèses et la négation de la conclusion.*
3. *Prouver par résolution que le raisonnement est correct.*

Autrement dit transformer le produit des hypothèses et de la négation de la conclusion en un produit de clauses, et en déduire la clause vide. □

Exercice 35 (Preuve,*) *Montrer, à l'aide d'une preuve par résolution, la correction du raisonnement suivant :*

$$r+q \Rightarrow t, t.q \Rightarrow r, q \models t \Leftrightarrow r.$$

□

Exercice 36 (Formalisation et preuve,*) *Montrer par résolution que le raisonnement suivant est contradictoire :*

— *Il fait beau à moins qu'il neige.*

— *Il pleut à moins qu'il neige.*

— *Il fait beau à moins qu'il pleuve.*

□

Exercice 37 (Définir une clause) *Une clause est la clause vide ou une disjonction (non vide) de littéraux. Donner une définition formelle de ce qu'est une clause et définir par récurrence la fonction s telle que $s(C)$ est l'ensemble des littéraux de la clause C.*

$\square$

Exercice 38 (Preuve) *Prouver la propriété 2.1.14 page 40 sur la monotonie et composition.* $\square$

Exercice 39 (Propriété de la résolution) *Montrer la propriété 2.1.8 page 39.*

$\square$

Exercice 40 (La résolution n'ajoute aucun littéral,*) *Soit Γ un ensemble de clauses. Un littéral de Γ est un littéral d'une clause de Γ. Montrer que toute clause déduite de Γ ne comporte que des littéraux de Γ.* $\square$

Exercice 41 (Formalisation et résolution) *Considérons les hypothèses suivantes :*

1. *Si Pierre rate son tournoi alors Pierre sera déprimé.*
2. *S'il fait beau alors Pierre ira à la piscine.*
3. *Si Pierre ne va pas à la piscine il sera déprimé.*
4. *À la piscine, Pierre ne s'entraîne pas.*
5. *Pierre ratera son tournoi s'il ne s'entraîne pas.*

Nous souhaitons démontrer que des hypothèses précédentes, on peut déduire la conclusion suivante :

— *Pierre sera déprimé.*

Vous procéderez comme suit :

— *Formaliser les hypothèses et la négation de la conclusion.*
— *Déduire de vos énoncés formels un ensemble de clauses équivalent.*
— *Prouver qu'il est correct de déduire la conclusion à partir des hypothèses en démontrant avec une preuve* **par résolution** *que l'ensemble de clauses est contradictoire.*

$\square$

Exercice 42 (Formalisation et résolution)

Les Beatles étaient un groupe de rock qui s'est formé dans les années soixantes à Liverpool. Ce groupe était composé de quatres garçons : Ringo Starr, Paul MacCartney, John Lennon et Georges Harrison. À l'époque où s'est formé le groupe, il n'a pas été évident de décider qui allait jouer de quel instrument. Pour preuve, voici un extrait de leurs discussions :

Paul dit : *« Si Ringo ne joue pas de la guitare, alors je jouerai de la basse et John jouera de la guitare »,*

Georges dit : *« Je jouerai de la guitare si et seulement si John en joue »,*

John dit : *« Si Paul joue de la basse, alors Georges jouera de la guitare »,*

Ringo dit : *« Je jouerai de la batterie et donc pas de la guitare ».*

Après cette discussion, ils décidèrent que :

— *Ringo jouerait à la batterie,*
— *Paul jouerait de la basse, et que*
— *John et Georges joueraient tous les deux de la guitare.*

Nous allons maintenant montrer que cette conclusion a satisfait tous les membres du groupe.

1. *Formalisez les quatres hypothèses et la conclusion en utilisant les variables propositionnelles suivantes :*
   — *RB : « Ringo joue de la batterie »,*
   — *RG : « Ringo joue de la guitare »,*
   — *PB : « Paul joue de la basse »,*
   — *JG : « John joue de la guitare », et*
   — *GG : « Georges joue de la guitare ».*
2. *Transformez en clauses les hypothèses et la négation de la conclusion.*
3. *Démontrez avec une preuve par résolution que la conclusion est conséquence des hypothèses.*

$\square$

Exercice 43 (Réduction, X1603 Andrews) *Soit l'ensemble de clauses :*

$$\{p+q, \overline{p}+r+\overline{q}+p, p+\overline{r}, q+\overline{p}+\overline{q}, q+\overline{r}+p, r+q+\overline{p}+\overline{r}, \overline{r}+q\}.$$

1. *Réduire cet ensemble (la réduction est définie en 2.1.4 page 42)*
2. *Indiquer si l'ensemble réduit est ou non satisfaisable.*

□

Exercice 44 (DPLL) *Considérons l'ensemble de clauses suivant :*

$$\{\overline{a}+\overline{b}+\overline{f}, a+b+f, e+\overline{a}, \overline{a}+\overline{b}, \overline{a}+c, d+a+\overline{d}, a+b, \overline{a}+\overline{c}+\overline{d}, d\}.$$

— *Appliquer l'algorithme Algo_DPLL sur cet ensemble de clauses.*
— *Conclure si cet ensemble est satisfaisable ou non.*
— *Donner un modèle ou un contre-modèle obtenu à partir de la trace de l'algorithme.*
Dans l'arbre d'appel vous étiquetterez les étapes comme suit :
— *Suppression des clauses valides, en abrégé VAL.*
— *Réduction, en abrégé RE.*
— *Suppression des clauses ayant des littéraux isolés, en abrégé ELI.*
— *Résolution unitaire, abrégé en RU.*
De plus, noter les affectations effectuées à chaque étape de l'algorithme afin de retrouver facilement le modèle construit.

□

Exercice 45 (DPLL) *Utiliser l'algorithme Algo_DPLL pour déterminer si les ensembles suivants de clauses sont satisfaisables ou insatisfaisables :*
— $\{a+b+c+d+e+f, \overline{a}+b, \overline{b}+a, \overline{c}+d, \overline{d}+c, \overline{b}+\overline{c}, \overline{b}+c, b+\overline{c}, \overline{e}, \overline{f}\}$.
— $\{a+b+c+d+f, \overline{a}+b, \overline{b}+a, \overline{c}+d, \overline{d}+c\}$.
— $\{b+j+\overline{a}, a+j+\overline{b}, b+a+j, a+j, j+b, \overline{b}+\overline{j}, \overline{j}+b, j+s, \overline{s}+\overline{b}\}$.
— $\{a+\overline{c}+d, \overline{b}+c+f, b+\overline{e}+f, \overline{c}+e+\overline{f}, e+f, c+d, \overline{a}, \overline{e}+\overline{f}\}$.
Donner une trace de l'algorithme.

□

Exercice 46 (Stratégie complète) *Soit* Γ *le produit suivant de clauses :*

$$(a+b+\overline{c}).(b+\overline{c}).(c+\overline{c}).(b+c).(\overline{a}+\overline{b}+c).(a+\overline{b}+c).$$

Déterminer par la stratégie complète de résolution si Γ *est insatisfaisable ou possède un modèle. Donner la trace de l'algorithme. Indiquer la ou les preuves obtenues. Si* Γ *a un modèle indiquez-le.* □

Exercice 47 (Stratégie complète) *Soient l'ensemble de clauses suivant :*

$$\{p+q, \overline{p}+s, \overline{s}+t, \overline{t}, \overline{q}+r, \overline{r}, \overline{r}+p+t, q+z+\overline{z}, \overline{q}+r+s\}.$$

Appliquer l'algorithme de la stratégie complète sur cet ensemble de clauses et conclure si cet ensemble est satisfaisable ou non.

□

Exercice 48 (Stratégie complète) *Considérons la fonction* f *telle que* $f(x,y,z)=0$ *si et seulement si le nombre d'arguments valant* 1 *est pair. Exprimer* f *comme un produit de clauses suivant la méthode décrite dans la sous-section 1.6 page 29 puis simplifier* f *en utilisant la stratégie complète.*

□

Exercice 49 (Preuve) *Prouver le lemme 2.3.2 page 46.* □

Exercice 50 (Preuve) *Prouver le lemme 2.3.5 page 46.* □

Exercice 51 (De SAT à 3-SAT,*)** *SAT est un problème de décision qui peut être énoncé comme suit : « étant donné un ensemble de clauses* Γ*, existe-t-il une assignation qui soit modèle de* Γ *? » 3-SAT est une restriction de ce problème où toutes les clauses de* Γ *ont exactement trois littéraux.*

SAT est un problème NP-complet [5]. Dans cet exercice, nous proposons d'étudier une réduction polynomiale de SAT vers 3-SAT, prouvant ainsi que 3-SAT est aussi NP-complet[2].

Une réduction polynomiale de SAT vers 3-SAT consiste à transformer en temps polynomial un ensemble de clauses Γ *en un ensemble de clauses* Γ' *vérifiant les deux propriétés suivantes :*

2. Il faut noter qu'en général, un problème k-SAT n'est pas nécessairement NP-complet. Par exemple, 2-SAT a été prouvé polynomial.

(a) Γ' *est uniquement constitué de clauses ayant trois littéraux distincts.*

(b) Γ *a un modèle si et seulement si* Γ' *a un modèle.*

Généralement, de telles réductions vérifient une propriété supplémentaire :

(c) Tout modèle de Γ' *est un modèle de* Γ.

Le but de cet exercice est de montrer que la transformation polynomiale présentée ci-dessous vérifie les trois propriétés énoncées ci-dessus. Noter que cette transformation introduit des variables supplémentaires.

Soit $\Gamma = \{c_1, \ldots, c_m\}$ *un ensemble de clauses. La réduction consiste à remplacer toute clause* $c_i = z_1 + \ldots + z_k$ *de* Γ *(ATTENTION :* $z_1, \ldots, z_k$ *sont des littéraux) par un ensemble de clauses* C_i' *construit en fonction de la valeur de k, comme suit :*

$k = 1$ **:** $C_i' = \{z_1 + y_1 + y_2,\ z_1 + y_1 + \overline{y_2},\ z_1 + \overline{y_1} + y_2,\ z_1 + \overline{y_1} + \overline{y_2}\}$ *où* y_1 *et* y_2 *sont des variables supplémentaires.*

$k = 2$ **:** $C_i' = \{z_1 + z_2 + y_1,\ z_1 + z_2 + \overline{y_1}\}$ *où* y_1 *est une variable supplémentaire.*

$k = 3$ **:** $C_i' = \{c_i\}$.

$k > 3$ **:** $C_i' = \{z_1 + z_2 + y_1,\ \overline{y_1} + z_3 + y_2,\ \overline{y_2} + z_4 + y_3,\ \overline{y_3} + z_5 + y_4,\ \ldots,\ \overline{y_{k-3}} + z_{k-1} + z_k\}$ *où* $y_1, \ldots, y_{k-3}$ *sont des variables supplémentaires.*

Ainsi, $\Gamma' = \bigcup_{i=1}^{m} C_i'$.

Par construction, la réduction proposée vérifie la propriété (a). *Les réponses aux questions suivantes permettent de prouver les propriétés* (b) *et* (c) *:*

1. *Montrer (sans table de vérité) que toute assignation donne la même valeur à* c_i *et* C_i' *lorsque* c_i *est constituée d'un seul littéral.*
2. *Montrer (sans table de vérité) que toute assignation donne la même valeur à* c_i *et* C_i' *lorsque* c_i *est constituée de deux littéraux.*
3. *Soit* c_i *une clause de plus de 3 littéraux. Montrer que si* c_i *a un modèle, alors* C_i' *a aussi un modèle.*
4. *Soit* c_i *une clause de plus de 3 littéraux. Montrer que tout modèle de* C_i' *est modèle de* c_i.
5. *La réduction proposée maintient uniquement la satisfaisabilité. Montrer qu'elle ne maintient pas le sens.*

□

Chapitre 3

Déduction Naturelle

Sommaire

UNE preuve dans les cours de Mathématiques est une décomposition du raisonnement en pas élémentaires évidents, nous pratiquons alors sans le savoir la *déduction naturelle*. Dans ce chapitre, nous présentons un système formel qui est une modélisation particulière de la déduction naturelle. La déduction naturelle et le calcul de séquent furent introduits en 1934 par Gerhard Gentzen (1909-1945) [12, 13]. Les preuves dans notre système ne sont pas des arbres comme dans le système original de Gehard Gentzen mais ressemblent plus aux preuves des mathématiciens. Nous expliquons aussi comment faire le lien entre ces deux formalismes en construisant des arbres proches de ceux introduits originellement. Nous choisissons de ne parler que de la *logique classique*, par opposition à la *logique intuitionniste* qui est obtenue en *omettant* la règle de réduction à l'absurde (la dernière règle de la table 3.1 page 59).

Remarque préliminaire : Pour simplifier l'étude de la déduction naturelle, nous considérons que le vrai, la négation et l'équivalence sont des *abréviations* ainsi définies :

— $\top$ abrège $\bot \Rightarrow \bot$.
— $\neg A$ abrège $A \Rightarrow \bot$.
— $A \Leftrightarrow B$ abrège $(A \Rightarrow B) \wedge (B \Rightarrow A)$.

Deux formules seront considérées comme *égales*, si les formules obtenues en éliminant les abréviations sont identiques. Par exemple, les formules $\neg\neg a$, $\neg a \Rightarrow \bot$ et $(a \Rightarrow \bot) \Rightarrow \bot$ sont égales. Il est clair que deux formules égales, au sens ci-dessus, sont équivalentes (voir exercice 55 page 67) et nous utilisons implicitement cette propriété quand nous remplaçons une formule par une autre formule égale aux abréviations près.

Plan : Nous décrivons tout d'abord notre système de règles de déduction naturelle et introduisons la notion de preuve dans ce contexte. Ensuite, nous présentons des tactiques pour aider le lecteur à la rédaction de preuve. Nous montrons la cohérence et la complétude de notre système. Enfin nous présentons un outil permettant à partir d'une formule de construire automatiquement une preuve ou bien d'exhiber un contre-exemple.

3.1 Système formel de la déduction naturelle

Nous présentons d'abord les règles du système de la déduction naturelle. Elles constituent les raisonnements élémentaires autorisés. Ensuite nous exposons la notion de preuve qui se construit par applications successives de ces règles.

Définition 3.1.1 (Règle) *Une* règle *est constituée de formules dites hypothèses $H_1, \ldots, H_n$ et d'une conclusion. Les hypothèses sont écrites au-dessus d'un trait et l'unique formule en dessous de ce trait est la conclusion de la règle. Le nom d'une règle est écrit au même niveau que le trait séparant hypothèses et conclusion.*

$$\frac{H_1 \ldots H_n}{C}\ R$$

La déduction naturelle que nous présentons comporte une dizaine de règles de déduction qui sont regroupées dans la table 3.1 page suivante. La règle fondamentale est la suivante : si nous pouvons déduire une formule B d'une hypothèse A, alors nous pouvons déduire $A \Rightarrow B$ en nous passant de cette hypothèse. Nous décrivons chacune des dix règles du système de déduction que nous considérons. L'application de ces règles et seulement de ces règles va nous permettre de prouver la correction des raisonnements. Il est donc important de bien comprendre le fonctionnement de chacune d'entre elles. Nous avons deux familles de règles pour chaque connecteur : les règles d'introduction et les règles d'élimination. L'intuition du rôle des règles d'introduction est de générer un symbole de la logique afin de bâtir progressivement la formule à prouver. Les règles d'élimination servent à simplifier un raisonnement ou à obtenir une nouvelle formule grâce aux raisonnements déjà réalisés et ainsi faire disparaître un connecteur logique. En plus nous avons deux règles spéciales, que nous décrirons en dernier, une pour éliminer deux occurrences successives de la négation et une autre pour déduire d'importe quoi à partir du faux.

3.1.1 Règles de conjonction

La règle d'introduction de la conjonction, notée $\wedge$I, signifie simplement que si nous avons une preuve de A et une preuve de B, alors nous pouvons déduire une preuve de la proposition $A \wedge B$. Nous pouvons aussi dire que la preuve de la proposition $A \wedge B$ peut se décomposer en une preuve de A et une preuve de B.

$$\frac{A \quad B}{A \wedge B}\ \wedge I$$

Les règles d'élimination de la conjonction, notées $\wedge E1$ et $\wedge E2$, permettent de déduire à partir de la conjonction de deux formules $A \wedge B$ soit la formule A soit la formule B. Ainsi nous avons éliminé le connecteur de la conjonction.

$$\frac{A \wedge B}{A}\ \wedge E1 \qquad \frac{A \wedge B}{B}\ \wedge E2$$

3.1.2 Règles de disjonction

Les règles d'introduction de la disjonction sont les duales des règles de l'élimination de la conjonction. Les deux règles d'introduction, notées $\vee$I1 et $\vee$I2, permettent de créer une disjonction de deux formules à partir d'une des deux formules. Ces règles signifient simplement que si A est vraie alors $A \vee B$ et $B \vee A$ sont aussi vraies quelle que soit la proposition B.

$$\frac{A}{A \vee B}\ \vee I1 \qquad \frac{A}{B \vee A}\ \vee I2$$

La règle d'élimination de la disjonction, notée $\vee E$, est plus complexe. Afin de déduire la formule C à partir de la disjonction $A \vee B$ il faut aussi prouver les prémisses suivantes : A implique C et B implique C. Cette règle formalise la notion de *preuve par cas* utilisée en mathématiques : supposons que dans un environnement donné, nous souhaitons prouver C alors que deux cas, A ou B, sont possibles ; nous nous plaçons alors dans le cas où A est vérifiée et nous prouvons C, puis nous nous plaçons dans le cas où B est vérifiée et nous prouvons C.

$$\frac{A \vee B \quad A \Rightarrow C \quad B \Rightarrow C}{C}\ \vee E$$

3.1.3 Règles de l'implication

La règle de l'élimination de l'implication $\Rightarrow E$ dit que si nous sommes capables d'obtenir une preuve de A et une preuve de $A \Rightarrow B$ alors nous avons une preuve de B. Cette règle correspond au *modus ponens*.

$$\frac{A \quad A \Rightarrow B}{B} \Rightarrow E$$

La règle fondamentale de notre système est la règle d'*introduction de l'implication* $\Rightarrow I$: si nous pouvons déduire une formule B d'une hypothèse A, alors nous pouvons déduire $A \Rightarrow B$ en nous passant de cette hypothèse. Cette règle est résumée par le schéma ci-dessous, où la notation $[A]$ indique que si A est une hypothèse de la preuve de B, cette hypothèse est *enlevée* de la preuve de $A \Rightarrow B$, autrement dit ne sert plus pour prouver $A \Rightarrow B$.

$$\frac{\begin{array}{c}[A]\\ \vdots\\ B\end{array}}{A \Rightarrow B} \Rightarrow I$$

3.1.4 Deux règles spéciales

Enfin nous avons deux règles spéciales :

— La première permet de déduire n'importe quoi à partir du faux : il s'agit de la « règle du faux », notée Efq, pour la formule latine « ex falso, quodlibet », indiquant que du faux, on peut déduire ce qu'on veut.

$$\frac{\perp}{A}\ Efq$$

— La seconde est la règle de réduction à l'absurde, notée RAA pour « reductio ad absurdum ». Elle élimine deux occurrences successives de la négation.

$$\frac{\neg\neg A}{A}\ RAA$$

Nous regroupons l'ensemble des règles de la déduction naturelle dans la table 3.1.

Introduction	Élimination
Règle du faux	
Règle de réduction à l'absurde	

TABLE 3.1 – Ensemble de règles de déduction naturelle.

3.1.5 Preuves en déduction naturelle

Intuitivement une preuve est l'application successive de règles de la déduction naturelle. Dans le système originel introduit par Gerhard Gentzen, une preuve est représentée par un arbre de formules (certaines formules pouvant être marquées comme enlevées). Nous avons choisi de représenter une preuve par une succession d'étapes de raisonnement basées sur les règles de la déduction naturelle, ceci afin d'être plus proche des preuves faites en mathématiques. Dans le chapitre précédent, une preuve par résolution était constituée d'une liste de clauses. En déduction naturelle, au cours d'une preuve, nous pouvons ajouter des hypothèses et les enlever, ce qui rend plus complexe la définition de ce qu'est une preuve. Nous introduisons donc plusieurs notions intermédiaires afin de définir clairement la notion de preuve en déduction naturelle. Tout d'abord nous introduisons les composants d'une preuve, c'est-à-dire, les « lignes d'une preuve », puis la notion de brouillon de preuve, ensuite les contextes de ces lignes de preuve afin de définir une preuve en déduction naturelle.

3.1.5.1 Brouillon de preuve

Une preuve est constituée d'une succession de lignes qui sont soit une formule soit une formule précédée par le mot clef « `Supposons` » ou « `Donc` ».

Définition 3.1.2 (Ligne de preuve) *Une* ligne *de preuve est de l'une des trois formes suivantes :*
- *`Supposons`* formule*,*
- formule*,*
- *`Donc`* formule*.*

Un brouillon de preuve est une preuve en construction, intuitivement elle a au moins autant de `Supposons` que de `Donc`.

Définition 3.1.3 (Brouillon de preuve) *Un* brouillon *de preuve est une suite de lignes telle que, dans tout préfixe de la suite, le nombre de lignes commençant par le mot `Supposons` est au moins égal à celui des lignes commençant par le mot `Donc`.*

Exemple 3.1.4 *Dans l'exemple ci-dessous, la suite des lignes de 1 à 3 est un brouillon de preuve. Par contre la suite des lignes de 1 à 5,* n'est pas un brouillon de preuve *car dans la suite des lignes 1 à 4, le nombre de lignes commençant par le mot `Donc` dépasse celui des lignes commençant par le mot `Supposons`.*

numéro	*ligne*
1	`Supposons` a
2	$a \vee b$
3	`Donc` $a \Rightarrow a \vee b$
4	`Donc` $\neg a$
5	`Supposons` b

3.1.5.2 Contexte des lignes d'un brouillon de preuve

Afin de pouvoir appliquer la règle d'introduction de l'implication nous avons besoin de connaître à chaque étape d'une preuve les formules qui sont des hypothèses utilisables. Pour ce faire, nous associons à chaque ligne d'un brouillon de preuve un *contexte*, qui est la suite des hypothèses introduites par les lignes `Supposons` et non enlevées par les lignes `Donc`. Puisque dans tout préfixe d'un brouillon de preuve, le nombre de lignes commençant par le mot `Supposons` est au moins égal à celui des lignes commençant par le mot `Donc`, le contexte de la ligne précédant une ligne `Donc` n'est pas vide et par suite le contexte de chaque ligne d'un brouillon de preuve est bien défini. Dans la suite, nous représentons les contextes de chaque ligne en remplaçant chaque hypothèse du contexte par le numéro de la ligne où cette hypothèse est introduite.

Définition 3.1.5 (Contexte) *Nous numérotons les lignes d'un brouillon de preuve de 1 à n. Pour i de 1 à n, la liste de formules Γ_i est le* contexte *de la ligne i. La liste Γ_0 est vide et les listes de formules Γ_i sont ainsi définies :*
- *Si la ligne i est « `Supposons` A », alors $\Gamma_i = \Gamma_{i-1}, i$.*
- *Si la ligne i est « A » alors $\Gamma_i = \Gamma_{i-1}$*
- *Si la ligne i est « `Donc` A » alors Γ_i est obtenue en enlevant la dernière formule de Γ_{i-1}*

La liste Γ_i *est le* contexte *de la ligne i.*

Nous présentons nos preuves sous forme de tableaux, ce qui nous permet de numéroter les lignes d'une preuve, les contextes utilisables, et les règles utilisées pour générer une nouvelle ligne.

Exemple 3.1.6 *Nous illustrons ici la gestion des contextes sur un brouillon de preuve simple.*

contexte	*numéro*	*ligne*	*règle*
1	*1*	`Supposons` a	
1,2	*2*	`Supposons` b	
1,2	*3*	$a \wedge b$	$\wedge I$ *1,2*
1	*4*	`Donc` $b \Rightarrow a \wedge b$	$\Rightarrow I$ *2,3*
1,5	*5*	`Supposons` e	

3.1.5.3 Preuves

Nous précisons la notion de conclusion et de formule utilisable afin de définir et manipuler des preuves en déduction naturelle.

Définition 3.1.7 (Conclusion, formule utilisable) *Nous définissons la notion de conclusion et de formule utilisable :*
- *La formule figurant sur une ligne d'un brouillon de preuve est la* conclusion *de la ligne.*
- *La conclusion d'une ligne est* utilisable *tant que son contexte (c'est-à-dire les hypothèses qui ont permis de la déduire) est présent.*

Autrement dit la conclusion de la ligne i est utilisable sur la ligne i et sur toutes les lignes suivantes dont le contexte a pour préfixe le contexte de la ligne i.

Exemple 3.1.8 *Dans l'exemple ci-dessous, la conclusion de la ligne 2 est utilisable sur la ligne 2, et pas au delà, car à la ligne 3, l'hypothèse qui a permis de la déduire est enlevée.*

contexte	*numéro*	*ligne*	*règle*
1	*1*	`Supposons` a	
1	*2*	$a \vee b$	$\vee I1$ *1*
	3	`Donc` $a \Rightarrow a \vee b$	$\Rightarrow I$ *1,2*
4	*4*	`Supposons` c	
4	*5*	$c \vee a$	$\vee I$ *4*
	6	`Donc` $c \Rightarrow c \vee a$	$\Rightarrow I$ *4,5*
	7	$(a \Rightarrow a \vee b) \wedge (c \Rightarrow c \vee a)$	$\wedge I$ *3,6*

Nous donnons la définition de preuve dans un environnement. Un environnement est un ensemble de formules supposées « vraies ». Intuitivement, réaliser une preuve dans un environnement consiste à pouvoir utiliser sans les prouver les formules de l'environnement.

Définition 3.1.9 (Preuve) *Soit* Γ *un ensemble de formules,* une preuve dans l'environnement Γ *est un brouillon de preuve ayant les propriétés suivantes :*

1. *Pour toute ligne «* `Donc` A *», la formule* A *est égale à* $B \Rightarrow C$*, où* B *est la dernière formule du contexte de la ligne précédente et où* C *est une formule utilisable sur la ligne précédente ou égale à un élément de l'environnement* Γ*.*
2. *Pour toute ligne «* A *», la formule* A *est la conclusion d'une règle (autre que la règle d'introduction de l'implication) dont les prémisses sont utilisables sur la ligne précédente ou sont éléments de l'environnement* Γ*.*

La ligne « `Donc` A » est une application de la règle d'introduction de l'implication. En effet, C est déduite de Γ ou d'hypothèses qui figurent dans la ligne précédente. La liste des hypothèses de la ligne précédente se terminant par B, nous pouvons en déduire $B \Rightarrow C$, en nous passant de l'hypothèse B.

Définition 3.1.10 (Preuve d'une formule) *Une preuve de la formule* A *dans l'environnement* Γ *est soit la preuve vide lorsque* A *est élément de* Γ*, soit une preuve dont la dernière ligne est de conclusion* A *et de contexte vide.*

Nous notons $\Gamma \vdash A$ le fait qu'il y a une preuve de A dans l'environnement Γ et $\Gamma \vdash P : A$ le fait que P est une preuve de A dans l'environnement Γ. Lorsque l'environnement est vide, nous l'omettons, autrement dit nous abrégeons $\emptyset \vdash A$ en $\vdash A$. Lorsque nous demandons une preuve d'une formule sans indiquer d'environnement, nous supposons que l'environnement est vide.

Exemple 3.1.11 *Prouvons* $(a \Rightarrow b) \Rightarrow (\neg b \Rightarrow \neg a)$.

contexte	*numéro*	*preuve*	*règle*
	1		
	2		
	3		
	4		
	5		
	6		
	7		
	8		

La preuve elle-même est ce qui figure dans la colonne preuve. Nous y avons ajouté la numérotation des lignes de preuves, les justifications qui indiquent les règles utilisées et les contextes de chaque ligne de la preuve.

Nous présentons la même preuve sous forme d'arbre, ce qui correspond à l'approche originelle de Gerhard Gentzen.

Nous remarquons que pour lever une hypothèse il est d'usage de barrer cette hypothèse.

Exemple 3.1.12 *Nous donnons la preuve de* $\neg A \vee B$ *dans l'environnement* $A \Rightarrow B$*. Nous suggérons de numéroter par i, ii, iii, etc... les formules de l'environnement pour distinguer ces numéros et les numéros des lignes de la preuve.*

environnement			
référence		*formule*	
i		$A \Rightarrow B$	
contexte	*numéro*	*preuve*	*règle*
	1		
	2		
	3		
	4		
	5		
	6		
	7		
	8		
	9		
	10		

Nous présentons aussi la preuve précédente sous forme d'arbre.

Comme il a été rappelé dans les remarques préliminaires, nous considérons comme égales deux formules qui sont identiques quand nous éliminons leurs abréviations. Ainsi dans l'exemple précédent, nous avons identifié $A \Rightarrow \perp$ et $\neg A$ ainsi que $(\neg A \vee B) \Rightarrow \perp$ et $\neg(\neg A \vee B)$.

3.2 Tactiques de preuve

Nous donnons des conseils pour construire des preuves. Ces conseils sont la base du logiciel présenté dans le paragraphe 3.5 page 66. Pour prouver la formule A dans un environnement Γ, nous appliquons *dans l'ordre* les règles suivantes, où l'ordre a été choisi de façon à reculer le plus possible l'usage de la règle RAA :

1. Si $A \in \Gamma$, alors la preuve obtenue est vide.
2. Si A est la conséquence d'une règle dont les premisses sont dans Γ, alors la preuve obtenue est « A ».
3. Si Γ comporte une contradiction, c'est-à-dire une formule B et une formule $\neg B$, alors la preuve obtenue est « $\perp$, A ».
4. Si $A = B \wedge C$, alors :
   — prouver B : Soit P la preuve obtenue pour B,
   — prouver C : Soit Q la preuve obtenue pour C.
   La preuve obtenue pour A est « P, Q, A ».

   Les preuves peuvent échouer (si l'on demande de prouver une formule improuvable dans l'environnement donné) : Si la preuve de B ou C échoue, il en est de même de la preuve de A. Pour simplifier la suite, nous ne signalons plus les cas d'échecs, sauf s'ils doivent être suivis d'une autre preuve.
5. Si $A = B \Rightarrow C$, alors prouver C sous l'hypothèse B (qui est ajoutée à l'environnement). Soit P, la preuve obtenue pour C, la preuve obtenue pour A est « `Supposons` B, P, `Donc` A ».
6. Si $A = B \vee C$, alors prouver B : Si P est la preuve obtenue pour B, alors « P, A » est la preuve obtenue pour A. Si la preuve de B échoue, alors prouver C : Si P est la preuve obtenue pour C, alors « P, A » est la preuve obtenue pour A. *Si la preuve de C échoue, essayer les règles suivantes.*
7. Si $B \wedge C$ est dans l'environnement, alors prouver A à partir des formules B, C, qui remplacent $B \wedge C$ dans l'environnement et soit P le résultat de cette preuve. Alors « B, C, P » est une preuve de A dans l'environnement initial.
8. Si $B \vee C$ est dans l'environnement, alors :
   — prouver A dans l'environnement où B remplace $B \vee C$: Soit P la preuve obtenue,
   — prouver A dans l'environnement où C remplace $B \vee C$: Soit Q la preuve obtenue.
   La preuve de A est alors « `Supposons` B, P, `Donc` $B \Rightarrow A$, `Supposons` C, Q, `Donc` $C \Rightarrow A$, A ».
9. Si $\neg(B \vee C)$ est dans l'environnement, alors nous déduisons $\neg B$ par la preuve $P4$ et $\neg C$ par la preuve $P5$ (preuves demandées à l'exercice 59 page 68). Soit P la preuve de A dans l'environnement où $\neg B$, $\neg C$ remplacent la formule $\neg(B \vee C)$. La preuve de A est « $P4$, $P5$, P ».

10. Si $A = B \vee C$, alors prouver C sous l'hypothèse $\neg B$: soit P la preuve obtenue. « `Supposons` $\neg B$, P, `Donc` $\neg B \Rightarrow C$ » est une preuve de la formule $\neg B \Rightarrow C$ qui est équivalente à A. Pour obtenir la preuve de A, il suffit d'ajouter la preuve $P1$, demandé à l'exercice 59 page 68, de A dans l'environnement $\neg B \Rightarrow C$. La preuve obtenue de A est donc « `Supposons` $\neg B$, P, `Donc` $\neg B \Rightarrow C$, $P1$ ».
11. Si $\neg(B \wedge C)$ est dans l'environnement, alors nous en déduisons $\neg B \vee \neg C$ par la preuve $P3$ demandée à l'exercice 59 page 68 puis nous raisonnons par cas comme ci-dessous :
    — prouver A dans l'environnement où $\neg B$ remplace $\neg(B \wedge C)$: Soit P la preuve obtenue,
    — prouver A dans l'environnement où $\neg C$ remplace $\neg(B \wedge C)$: Soit Q la preuve obtenue.
    La preuve de A est « $P3$, `Supposons` $\neg B$, P, `Donc` $\neg B \Rightarrow A$, `Supposons` $\neg C$, Q, `Donc` $\neg C \Rightarrow A$, A ».
12. Si $\neg(B \Rightarrow C)$ est dans l'environnement, alors nous déduisons B par la preuve $P6$, $\neg C$ par la preuve $P7$ (preuves demandées à l'exercice 59 page 68). Soit P la preuve de A dans l'environnement où B, $\neg C$ remplacent la formule $\neg(B \Rightarrow C)$. La preuve de A est « $P6$, $P7$, P ».
13. Si $B \Rightarrow C$ est dans l'environnement et si $C \neq \bot$, autrement dit si $B \Rightarrow C$ n'est pas égale à $\neg B$, alors, nous déduisons $\neg B \vee C$ dans l'environnement $B \Rightarrow C$ par la preuve $P2$ demandée à l'exercice 59 page 68 puis nous raisonnons par cas :
    — prouver A dans l'environnement où $\neg B$ remplace $B \Rightarrow C$: Soit P la preuve obtenue,
    — prouver A dans l'environnement où C remplace $B \Rightarrow C$: Soit Q la preuve obtenue.
    La preuve de A est « $P2$, `Supposons` $\neg B$, P, `Donc` $\neg B \Rightarrow A$, `Supposons` C, Q, `Donc` $C \Rightarrow A$, A ».

Exemple 3.2.1 *Nous appliquons ces tactiques à la preuve de la loi de Peirce*[1] *: $((p \Rightarrow q) \Rightarrow p) \Rightarrow p$. Cette formule n'est pas prouvable en logique intuitionniste, qui est, ainsi qu'il est dit au début du chapitre, la logique classique privée de la règle de réduction à l'absurde. Nous pouvons montrer, qu'en logique intuitionniste, cette formule est équivalente à la loi du tiers-exclu.*

Vu la forme de la formule, nous devons appliquer la tactique 5. La preuve de la formule de Peirce est donc de la forme suivante :

Preuve Q :
`Supposons` $(p \Rightarrow q) \Rightarrow p$
$\boxed{Q_1 \text{ preuve de } p \text{ dans l'environnement } (p \Rightarrow q) \Rightarrow p}$
`Donc` $((p \Rightarrow q) \Rightarrow p) \Rightarrow p$

La preuve Q_1 utilise nécessairement la tactique 13. Donc cette preuve s'écrit : Dans l'environnement $B \Rightarrow C$ où $B = p \Rightarrow q$, $C = p$.

Preuve Q_1 :
$\boxed{Q_{11} = P2 \text{ où } P2 \text{ est la preuve de } \neg B \vee C \text{ dans l'environnement } B \Rightarrow C\text{, voir exercice 59 page 68}}$
`Supposons` $\neg B$
$\boxed{Q_{12} \text{ preuve de } A = p \text{ dans l'environnement } \neg B}$
`Donc` $\neg B \Rightarrow A$
`Supposons` C
$\boxed{Q_{13} \text{ preuve de } A = p \text{ dans l'environnement } C}$
`Donc` $C \Rightarrow A$
A

Q_{13} est la preuve vide, car $A = C = p$.
Q_{12} est la preuve de $A = p$ dans l'environnement $\neg(p \Rightarrow q)$. Cette preuve est la preuve $P6$ demandée à l'exercice 59 page 68, où $B = p$ et $C = q$. En recollant les morceaux Q_1, Q_{11}, Q_{12}, Q_{13}, nous obtenons la preuve Q, que nous ne recopions pas ici, car elle peut être obtenue automatiquement par le logiciel développé par Michel Levy, disponible à l'adresse suivante : `http://teachinglogic.univ-grenoble-alpes.fr/DN/`

Nous montrons ci-dessous comme retrouver la preuve Q_{12} sans utiliser les tactiques. La seule règle, ne conduisant pas à une impasse, est la réduction à l'absurde. Donc cette preuve est de la forme :

1. Charles Sanders Peirce était un logicien et philosophe américain (1839 - 1914).

Preuve Q_{12} de p dans l'environnement $\neg(p \Rightarrow q)$
`Supposons` $\neg p$
$\boxed{Q_{121} \text{ preuve de } \perp \text{ dans l'environnement } \neg(p \Rightarrow q),\ \neg p}$
`Donc` $\neg\neg p$
p

Pour obtenir une contradiction, donc une preuve de $\perp$, il faut déduire $p \Rightarrow q$. Donc la preuve Q_{121} est :

`Supposons` p
$\perp$
q
`Donc` $p \Rightarrow q$
$\perp$

3.3 Cohérence de la déduction naturelle

La preuve de cohérence de la déduction naturelle consiste à montrer que chaque preuve construite à partir d'un ensemble de formules Γ donne une formule qui est déductible de Γ.

Théorème 3.3.1 (Cohérence de la déduction) *Si une formule A est déduite d'un environnement de formules Γ ($\Gamma \vdash A$), alors A est une conséquence de Γ ($\Gamma \models A$).*

Preuve : Soit Γ un ensemble de formules. Soit P une preuve de A dans cet environnement. Soient C_i la conclusion et H_i le contexte de la $i^{\text{ème}}$ ligne de la preuve P. Nous posons $H_0 = \emptyset$, la liste vide. Si la preuve P est vide, nous posons $C_0 = A$. Notons par $\Gamma,\ H_i$ l'ensemble des formules de l'ensemble Γ et de la liste H_i. Nous montrons par induction que pour tout k nous avons $\Gamma,\ H_k \models C_k$, ce qui implique que pour la dernière ligne n de cette preuve, comme par définition H_n est la liste vide et $C_n = A$, nous obtenons $\Gamma \models A$.

Cas de base : Supposons que A est déduite de Γ par la preuve vide. Alors A est élément de Γ, donc $\Gamma \models A$. Puisque $H_0 = \emptyset$, nous pouvons conclure : $\Gamma, H_0 \models A$, donc $\Gamma, H_0 \models C_0$.

Induction : Supposons que pour toute ligne $i < k$ de la preuve nous avons $\Gamma,\ H_i \models C_i$. Montrons que $\Gamma,\ H_k \models C_k$. Supposons que la ligne k est :

1. « `Supposons` C_k ». La formule C_k est la dernière formule de H_k, donc $\Gamma,\ H_k \models C_k$.
2. « `Donc` C_k ». La formule C_k est égale à la formule $B \Rightarrow D$, et B est la dernière formule de H_{k-1}. Nous distinguons deux cas, soit D est élément de Γ, soit D est utilisable sur la ligne précédente.
   (a) Dans le premier cas, puisque D est égale à une formule de Γ, D est conséquence de Γ donc de $\Gamma,\ H_k$. Puisque $B \Rightarrow D$ est conséquence de D, il en résulte que $\Gamma,\ H_k \models C_k$
   (b) Dans le deuxième cas, D est utilisable sur la ligne précédente. Donc il existe $i < k$ tel que $D = C_i$ et H_i est préfixe de H_{k-1}. Par *hypothèse de récurrence*, $\Gamma,\ H_i \models D$. Puisque H_i est préfixe de H_{k-1}, nous avons $\Gamma,\ H_{k-1} \models D$. Puisque B est la dernière formule de H_{k-1}, nous avons $H_{k-1} = H_k,\ B$ et donc $\Gamma,\ H_k,\ B \models D$, ce qui implique $\Gamma,\ H_k \models B \Rightarrow D$. Enfin puisque C_k est égale à $B \Rightarrow D$ et que deux formules égales sont équivalentes, nous avons $\Gamma,\ H_k \models C_k$.
3. « C_k ». Cette formule est la conclusion d'une règle de la table 3.1 page 59, appliquée à ses prémisses utilisables à la ligne précédente ou aux éléments de Γ. Considérons le seul cas de la règle $\wedge I$, les autres cas étant analogues. La formule C_k est égale à $(D \wedge E)$ et les prémisses de la règle sont D et E. Puisque D et E sont éléments de Γ ou utilisables à la ligne précédente, comme dans le cas précédent, en utilisant l'hypothèse de récurrence, nous avons : $\Gamma,\ H_{k-1} \models D$ et $\Gamma,\ H_{k-1} \models E$. Puisque la ligne k ne change pas les hypothèses, nous avons $H_{k-1} = H_k$, donc $\Gamma,\ H_k \models D$ et $\Gamma,\ H_k \models E$. Puisque C_k est égale à $(D \wedge E)$, nous avons : $D,\ E \models C_k$. Par suite $\Gamma,\ H_k \models C_k$.

□

3.4 Complétude de la déduction naturelle

Nous prouvons la complétude des règles uniquement pour les formules avec les symboles logiques $\perp$, $\wedge$, $\vee$, $\Rightarrow$. La complétude pour les formules obtenues en ajoutant les symboles logiques $\top$, $\neg$ et $\Leftrightarrow$ résulte immédiatement du fait que ces symboles peuvent être considérés comme des abréviations.

Vu l'absence de négation, nous définissons un littéral comme étant une variable ou une implication entre une variable et $\perp$. Soit x une variable, x et $x \Rightarrow \perp$ (qui peut être abrégé en $\neg x$) sont des littéraux complémentaires. Dans la suite nous distinguons une liste de formules Γ et $s(\Gamma)$ l'ensemble des formules de la liste. Pour simplifier les notations, nous utilisons la virgule pour ajouter un élément en début ou en fin de liste ainsi que pour concaténer deux listes, que ces listes soit des listes de formules ou des preuves.

Théorème 3.4.1 *Soient Γ un ensemble fini de formules et A une formule, si $\Gamma \models A$ alors $\Gamma \vdash A$.*

Remarque 3.4.2 *Nous admettons ce thèorème dans ce polycopié. On peut en donner une preuve constructive, c'est-à-dire qu'elle donne un ensemble complet de tactiques pour construire des preuves d'une formule dans un environnement. Cependant ces tactiques peuvent donner des preuves longues. En particulier si nous devons prouver une formule $(B \vee C)$, il vaut mieux en général suivre les tactiques données au paragraphe 3.2 page 63, c'est-à-dire essayer de prouver B, puis essayer de prouver C et* seulement en cas d'échec*, utiliser la tactique donnée dans la preuve de complétude, qui « réduit » cette preuve à une preuve de C en ajoutant l'hypothèse $\neg B$.*

3.5 Outils

Nous indiquons deux logiciels pour se familiariser avec la déduction naturelle. Le premier construit automatiquement des preuves comme nous les avons présentées, le deuxième illustre sur quelques exemples de manière interactive comment dessiner des preuves sous forme d'arbres.

3.5.1 Logiciel de construction automatique de preuves

Pour produire automatiquement des preuves, nous recommandons d'utiliser le logiciel :

`http://teachinglogic.univ-grenoble-alpes.fr/DN/`

Ce logiciel permet grâce à une syntaxe intuitive de saisir une formule et de générer automatiquement :
— si la formule est (syntaxiquement) incorrecte, un message d'erreur (en rouge) est produit au dessus de la formule
— si la formule est prouvable, sa preuve (sans annotation) est générée par le logiciel.
— si la formule est correcte mais n'est pas prouvable, un contre-modèle est proposé au dessus de la formule.
Si la formule est prouvable, il est aussi possible d'obtenir une version de la preuve annotée.

3.5.2 Dessiner des arbres de preuves

Ceux qui affectionnent les preuves sous forme d'arbres peuvent utiliser le logiciel suivant développé Laurent Théry :

`http://www-sop.inria.fr/marelle/Laurent.Thery/peanoware/Nd.html`

ou sa version Android :

`https://play.google.com/store/apps/details?id=org.inria.peanoware`

Ce logiciel se présente comme un jeu, il propose des formules à prouver et permet d'appliquer les règles de la déduction naturelle, de manière interactive, pour en construire les arbres de preuve. Quand vous réussissez à construire la preuve d'une formule, vous voyez la photo de Dag Prawitz, un des principaux logiciens qui a étudié les propriétés de la déduction naturelle.

3.6 Exercices

Exercice 52 (Brouillon de preuve) *La suite de lignes suivante n'est pas un brouillon de preuve. Indiquer le numéro i de la première ligne telle que les lignes 1 à $i-1$ soient un brouillon de preuve et que les lignes 1 à i ne soient pas un brouillon de preuve. Préciser le contexte des lignes 1 à $i-1$.*

contexte	*numéro*	*preuve*
	1	`Supposons` a
	2	`Supposons` b
	3	c
	4	`Donc` d
	5	`Supposons` e
	6	f
	7	`Donc` g
	8	h
	9	i
	10	`Donc` j
	11	`Donc` k
	12	l

□

Exercice 53 (Preuves de formules avec les règles spéciales) *Donner une preuve pour les formules suivantes :*

1. $a \Rightarrow \neg\neg a$,
2. $\neg\neg a \Rightarrow a$,
3. $a \Leftrightarrow \neg\neg a$,
4. $a \vee \neg a$.

□

Exercice 54 (Preuves simples de formules) *Donner une preuve pour les formules suivantes :*

1. $a \Rightarrow c$ *dans l'environnement* $a \Rightarrow b,\ b \Rightarrow c$.
2. $(a \Rightarrow b) \wedge (b \Rightarrow c) \Rightarrow (a \Rightarrow c)$.
3. $(a \Rightarrow b) \Rightarrow ((b \Rightarrow c) \Rightarrow (a \Rightarrow c))$.

□

Exercice 55 (Abréviations) *Soient A une formule et $dpl(A)$ la formule obtenue en remplaçant dans A, le vrai, les négations et les équivalences (en forme abréviées) par leur définition. $dpl(A)$ est la formule obtenue en « dépliant » A, d'où le nom dpl choisi pour cette fonction de dépliage.*

— *Définir par récurrence la fonction dpl.*
— *Montrer que les formules A et $dpl(A)$ sont équivalentes.*
— *En déduire que deux formules, égales aux abréviations près, sont équivalentes.*

□

Exercice 56 (Preuves de formules) *Donner une preuve pour les formules suivantes :*

1. $a \Rightarrow (b \Rightarrow a)$.
2. $a \wedge b \Rightarrow a$.
3. $\neg a \Rightarrow (a \Rightarrow b)$.
4. $(a \Rightarrow (b \Rightarrow c)) \Rightarrow ((a \Rightarrow b) \Rightarrow (a \Rightarrow c))$.
5. $a \wedge b \Rightarrow b \wedge a$.
6. $a \vee b \Rightarrow b \vee a$.
7. $(a \Rightarrow (b \Rightarrow c)) \Rightarrow (a \wedge b \Rightarrow c)$.
8. $(a \wedge b \Rightarrow c) \Rightarrow (a \Rightarrow (b \Rightarrow c))$.

9. $(a \Rightarrow b) \wedge (c \Rightarrow d) \Rightarrow (a \wedge c \Rightarrow b \wedge d)$.

$\square$

Exercice 57 (À propos de l'implication) *Donner une preuve pour les formules suivantes :*

1. *(**)* $a \Rightarrow b$ *dans l'environnement* $\neg a \vee b$.
2. $(\neg b \Rightarrow \neg a) \Rightarrow (a \Rightarrow b)$.

$\square$

Exercice 58 (Algèbre de Boole) *Donner une preuve pour les formules suivantes :*

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

$\square$

Exercice 59 (Preuves de formules avec environnement) *Donner une preuve des formules suivantes :*

1. *P1 :* $B \vee C$ *dans l'environnement* $\neg B \Rightarrow C$.
2. *P2 :* $\neg B \vee C$ *dans l'environnement* $B \Rightarrow C$.
3. *P3 :* $\neg B \vee \neg C$ *dans l'environnement* $\neg(B \wedge C)$.
4. *P4 :* $\neg B$ *dans l'environnement* $\neg(B \vee C)$.
5. *P5 :* $\neg C$ *dans le même environnement* $\neg(B \vee C)$.
6. *P6 :* B *dans l'environnement* $\neg(B \Rightarrow C)$.
7. *P7 :* $\neg C$ *dans le même environnement* $\neg(B \Rightarrow C)$.

$\square$

Exercice 60 (Preuves de formules) *Donner une preuve en déduction naturelle des formules suivantes :*

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

$\square$

Exercice 61 (Partiel 2011) *Prouver les formules suivantes en utilisant la déduction naturelle sous forme de tableau :*

1. $\neg p \wedge \neg(\neg p \wedge q) \Rightarrow \neg q$.
2. $((p \vee q) \vee (p \vee r)) \wedge \neg p \Rightarrow q \vee r$.
3. *(*)* $\neg p \vee \neg(p \wedge q) \Rightarrow \neg q \vee \neg p$.

$\square$

Exercice 62 (Partiel 2013) *Donner une preuve des formules suivantes en utilisant la déduction naturelle sous forme de tableau :*

1. $(p \vee q) \Rightarrow (\neg p \wedge \neg q) \Rightarrow r$.
2. $((p \Rightarrow q) \wedge (q \Rightarrow r)) \wedge \neg r \Rightarrow \neg p$.
3. $(p \Rightarrow q) \Rightarrow ((p \wedge q) \vee \neg p)$.

$\square$

Exercice 63 (Quelques questions posées en examen)

1. *Démontrer la formule suivante par déduction naturelle :* $(p \vee q) \wedge (p \Rightarrow r) \Rightarrow q \vee r$
2. *Soient les hypothèses :*
   - *(H1) : Si Pierre est grand, alors Jean n'est pas le fils de Pierre*
   - *(H2) : Si Pierre n'est pas grand, alors Jean est le fils de Pierre*
   - *(H3) : Si Jean est le fils de Pierre alors Marie est la soeur de Jean*

   Formaliser et démontrer qu'on peut en déduire la conclusion (C) : « Marie est la soeur de Jean ou Pierre est grand ou les deux à la fois » par déduction naturelle.

□

Deuxième partie

Logique du premier ordre

Chapitre 4

Logique du premier ordre

Sommaire

Ce célèbre syllogisme [1] ne peut pas se formaliser en logique propositionnelle :

Tous les hommes sont mortels ;
or Socrate est un homme ;
donc Socrate est mortel.

Pour formaliser un tel raisonnement nous avons besoin d'enrichir la logique propositionnelle avec de nouveaux connecteurs appelés *quantificateurs*, cette logique étendue s'appelle logique du premier ordre. Elle permet de parler de structures comportant un seul domaine non vide (contrairement à la logique propositionelle, ce domaine peut avoir plus de deux valeurs), des fonctions et relations sur ce domaine. Le langage de cette logique comporte plusieurs catégories : les termes qui représentent les éléments du domaine ou des fonctions sur ces éléments, des relations qui relient des termes entre eux et les formules qui décrivent les interactions entre les relations grâce aux connecteurs et aux quantificateurs. Par exemple, la relation $mortel(x)$ désigne x est mortel, la relation $homme(x)$ signifie que x est un homme. De plus, $Socrate$ est une constante du domaine (c'est-à-dire, elle a pour valeur un élément du domaine) et $homme(Socrate)$ signifie que Socrate est un homme. Enfin, la formule $\forall x\ (homme(x) \Rightarrow mortel(x))$ indique que tous les hommes sont mortels. À partir de ces deux hypothèses, il est possible de conclure que Socrate est mortel ($mortel(Socrate)$) par différentes méthodes

1. La notion de syllogisme fut introduite par Aristote et signifie littéralement « parole (qui va) avec (une autre) », aussi connu en latin par *modus ponendo ponens* = « manière d'affirmer, d'établir en affirmant » et plus brièvement *modus ponens*.

que nous détaillerons dans cette seconde partie. Ce même raisonnement s'applique également pour prouver le syllogisme suivant :

Un cheval bon marché est rare.
Tout ce qui est rare est cher.
Donc un cheval bon marché est cher.

Malgré la conclusion qui semble contradictoire, le raisonnement est correct. Afin d'obtenir une contradiction il faut ajouter l'hypothèse suivante : $\forall x($ bonmarché $(x) \Leftrightarrow \neg$ cher $(x))$. Ceci traduit la relation communément admise entre la notion de bon marché et de cher, *i.e.*, tout ce qui est cher n'est pas bon marché et réciproquement. Avec cette hypothèse supplémentaire nous pouvons montrer que ce raisonnement est contradictoire, sans cette hypothèse le raisonnement est correct. Ceci élucide le paradoxe de ce syllogisme.

Notre objectif dans ce chapitre est d'introduire les concepts et notions élémentaires de la logique du premier ordre afin de pouvoir proposer des méthodes de raisonnement dans les chapitres suivants.

Plan : Nous commençons par décrire la syntaxe de la logique du premier ordre. Puis, nous définissons les notions de *libre* et *liée* qui sont essentielles pour pouvoir interpréter le sens des formules. Ensuite nous définissons le sens des formules que nous pouvons construire à partir de la syntaxe introduite précédemment. Enfin, nous énonçons des propriétés remarquables de la logique du premier ordre, qui pourront être utilisées dans les raisonnements.

4.1 Syntaxe

Nous ajoutons à la logique propositionnelle deux symboles : le symbole *existentiel* ($\exists$) et le symbole *universel* ($\forall$). Le symbole existentiel signifie qu'il existe un élément ayant une certaine propriété alors que le symbole universel permet de parler de tous les éléments ayant une propriété. Ces changements impliquent qu'un élément de ce langage peut désormais dépendre de plusieurs variables. Il faut donc rajouter dans la syntaxe un délimiteur de variables. Nous avons choisi classiquement la virgule. Ces changements introduisent de nouveaux symboles en plus des variables, ces symboles peuvent par exemple être des nombres sur lesquels nous pouvons créer des fonctions (comme l'addition) ou encore définir des relations (comme l'égalité). Toutes ces modifications rendent la présentation de la syntaxe légèrement plus subtile que celle de la logique propositionnelle.

4.1.1 Formules strictes

Pour écrire les formules de la logique du premier ordre nous étendons la syntaxe de la logique propositionnelle, ainsi nous disposons du vocabulaire suivant :

- Deux constantes propositionnelles : $\perp$ et $\top$ représentant respectivement le faux et le vrai.
- Variables : suite de lettres et de chiffres commençant par une des minuscules u, v, w, x, y, z.
- Connecteurs : $\neg, \wedge, \vee, \Rightarrow, \Leftrightarrow$.
- Quantificateurs : $\forall$ le quantificateur universel et $\exists$ le quantificateur existentiel.
- Ponctuations : la virgule « , » et les parenthèses ouvrantes « (» et fermantes «) ».
- Symbole :
  - ordinaire : suite de lettres et de chiffres ne commençant pas par une des minuscules u, v, w, x, y, z.
  - spécial : $+, -, *, /, =, \neq, <, \leq, >, \geq$

L'ensemble des symboles spéciaux peut être augmenté de nouveaux éléments, suivant les domaines mathématiques étudiés, à condition de respecter la contrainte que ces nouveaux éléments ne soient pas dans les ensembles des constantes propositionnelles, des variables, des connecteurs, des quantificateurs, des ponctuations ou des symboles ordinaires.

Exemple 4.1.1 *Ci-dessous, nous illustrons ces notions :*

- $x, x1, x2, y$ *sont des variables,*
- *homme, parent, succ, 12, 24,* $f1$ *sont des symboles ordinaires, les symboles ordinaires représenteront des fonctions (constantes numériques ou fonctions à plusieurs arguments) ou des relations (variables propositionelles ou relations à plusieurs arguments).*
- $x = y$, $z > 3$ *sont des exemples d'application de symboles spéciaux.*

Nous définissons en toute généralité la notion de terme qui est essentielle en logique du premier ordre. Nous étendrons dans la suite cette notion en définissant ce qu'est un terme associé à un ensemble de symboles.

Définition 4.1.2 (Terme) *Un* terme *est défini de manière inductive par :*
- *un symbole ordinaire est un terme,*
- *une variable est un terme,*
- *si $t_1, \ldots, t_n$ sont des termes et si s est un symbole (ordinaire ou spécial) alors $s(t_1, \ldots, t_n)$ est un terme.*

Exemple 4.1.3

sont des termes, par contre

n'est pas un terme. Notons que $42(1, y, 3)$ *est aussi un terme, mais l'usage veut que les noms des fonctions et relations soient des symboles ordinaires commençant par des lettres.*

Définition 4.1.4 (Formule atomique) *Nous définissons une formule atomique de manière inductive par :*
- *$\top$ et $\bot$ sont des formules atomiques*
- *un symbole ordinaire est une formule atomique*
- *si $t_1, \ldots, t_n$ sont des termes et si s est un symbole (ordinaire ou spécial) alors $s(t_1, \ldots, t_n)$ est une formule atomique.*

Exemple 4.1.5

sont des formules atomiques, notons également que

ne sont pas des formules atomiques.

Notons que l'ensemble des termes et l'ensemble des formules atomiques ne sont pas disjoints. Par exemple, $p(x)$ est à la fois un terme et une formule atomique. Lorsque t est à la fois un terme et une formule atomique, nous distinguons $[\![t]\!]$ le sens de t vu comme un terme, de $[t]$ le sens de t vu comme une formule (voir le paragraphe 4.3.4 page 80).

Définition 4.1.6 (Formule) *Nous définissons une* formule *de manière inductive par :*
- *une formule atomique est une formule,*
- *si A est une formule alors $\neg A$ est une formule,*
- *si A et B sont des formules et si $\circ$ est une des opérations $\vee, \wedge, \Rightarrow, \Leftrightarrow$ alors $(A \circ B)$ est une formule ,*
- *si A est une formule et si x est une variable* quelconque *alors $\forall x\, A$ et $\exists x\, A$ sont des formules* [2].

Exemple 4.1.7

sont des formules atomiques, donc des formules. Par contre

est une formule qui n'est pas atomique.

La notion de *sous-formule* (définition 1.1.4) s'étend naturellement à la logique du premier ordre. La notion de *taille de formule stricte* nécessite qu'en à elle une nouvelle définition :

Définition 4.1.8 (Taille d'une formule) *La* taille d'une formule *A, notée $|A|$, est définie inductivement par :*
- $|\top| = 0$ *et* $|\bot| = 0$.
- *Si A est une formule atomique alors* $|A| = 0$.
- $|\neg A| = 1 + |A|$.
- $|Qx\, A| = 1 + |A|$ *si Q est un des quantificateurs $\forall$ ou $\exists$.*
- $|(A \circ B)| = |A| + |B| + 1$.

2. Attention, x peut ne pas exister dans A.

4.1.2 Formules à priorité

Nous reprenons les priorités de la logique propositionnelle pour les connecteurs, et nous ajoutons une priorité identique à celle de la négation pour les quantificateurs. Dans le tableau 4.1 nous donnons les priorités des symboles et des connecteurs par ordre décroissant du haut vers le bas. Nous indiquons seulement deux règles de formation des formules à priorité, qui les distinguent des formules complètement parenthésées et permettent d'omettre des parenthèses ou d'ajouter des nouvelles parenthèses.

Pour abréger l'écriture des termes, certains symboles de fonction $+,-,*,/$ et certains symboles de relations $=,\neq,<,\leq,>,\geq$ peuvent être écrits de manière infixe, c'est-à-dire, de façon usuelle.

Exemple 4.1.9 *Nous* abrégeons *le terme* $+(x,*(y,z))$ *en* $x+y*z$ *et* $\leq(*(3,x),+(y,5))$ *en* $3*x\leq y+5$. *La transformation inverse est définie en donnant aux symboles* $=,\neq,<,\leq,>,\geq$ *des priorités inférieures à celle des symboles* $+,-,*,/$.

OPÉRATIONS	
$-+$ unaire	
$*,/$	associatif gauche
$+,-$ binaire	associatif gauche
RELATIONS	
$=,\neq,<,\leq,>,\geq$	
NÉGATION, QUANTIFICATEURS	
$\neg,\forall,\exists$	
CONNECTEURS BINAIRES	
$\wedge$	associatif gauche
$\vee$	associatif gauche
$\Rightarrow$	associatif **droit**
$\Leftrightarrow$	associatif gauche

TABLE 4.1 – Priorités des connecteurs et symboles.

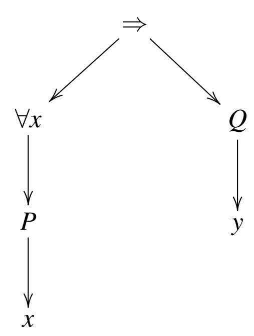
Définition 4.1.10 (Formule à priorité) *Une* formule à priorité *est :*

- *Une formule atomique.*
- *Si A est une formule à priorité alors $\neg A$ est une formule à priorité.*
- *Si A et B sont des formules à priorité alors $A\circ B$ est une formule à priorité.*
- *Si A est une formule à priorité et si x une variable* quelconque *alors $\forall x\, A$ et $\exists x\, A$ sont des formules à priorité.*
- *Si A est une formule à priorité alors (A) est une formule à priorité.*

Exemple 4.1.11 *La formule (à priorité)* $\forall xP(x)\wedge\forall xQ(x)\Leftrightarrow\forall x(P(x)\wedge Q(x))$ *peut être vue comme une* abréviation *de la formule* $((\forall xP(x)\wedge\forall xQ(x))\Leftrightarrow\forall x(P(x)\wedge Q(x)))$.

La formule (à priorité) $\forall x\forall y\forall z(x\leq y\wedge y\leq z\Rightarrow x\leq z)$ *peut être vue comme une* abréviation *de la formule* $\forall x\forall y\forall z((\leq(x,y)\wedge\leq(y,z))\Rightarrow\leq(x,z))$.

Exemple 4.1.12 *La priorité du $\forall$ est plus forte que celle de $\Rightarrow$, dans la formule* $\forall xP(x)\Rightarrow Q(y)$. *Donc, l'opérande gauche de l'implication est* $\forall xP(x)$. *La structure de la formule sera ainsi représentée par l'arbre suivant :*



De manière analogue à la logique propositionnelle, la taille d'une formule à priorité sera égale à la taille de la formule stricte dont elle est l'abréviation. De même, pour les sous-formules, nous considérerons toujours la formule stricte dont la formule à priorité est l'abréviation.

4.2 Être libre ou lié

Le sens de la formule $x+2=4$ dépend de x : la formule n'est vraie (en arithmétique) que si $x=2$. La variable x est libre dans cette formule.

Par contre, toujours en arithmétique $\forall x(x+2=4)$ est une formule fausse et $\exists x(x+2=4)$ est une formule vraie car $2+2=4$. Pour chacune de ces deux formules, il n'y a pas à choisir de valeur pour x afin de déterminer leur valeur respective : ces deux formules n'ont pas de variables libres.

4.2.1 Occurrences libres et liées

Sur une représentation en arbre, où nous dessinons les structures des formules en faisant apparaître $\forall x$ et $\exists x$ comme des sommets de l'arbre, une *occurrence liée* de la variable x est une occurrence en dessous d'un sommet $\exists x$ ou $\forall x$. Une occurrence de x qui n'est pas sous un tel sommet est *libre*.

Définition 4.2.1 (Portée de liaison, occurrence libre, liée) *Soient x une variable et A une formule. Dans une formule $\forall x\, A$ ou $\exists x\, A$, la* portée de la liaison *pour x est A. Une occurrence de x dans une formule est* libre *si elle n'est pas dans la portée d'une liaison pour x, sinon elle est dite* liée.

Exemple 4.2.2 *Pour voir les occurrences des variables, nous dessinons la structure de la formule $\forall x P(x,y) \wedge \exists z R(x,z)$, puisque la priorité de $\forall$ est supérieure à celle du $\wedge$, nous obtenons :*

L'occurrence de x présente dans $P(x,y)$ est liée, l'occurrence de x présente dans $R(x,z)$ est libre. L'occurrence de z est liée.

4.2.2 Variables libres et liées

Définition 4.2.3 (Variable libre, liée, formule fermée) *La variable x est une* variable libre *d'une formule si et seulement s'il y a une occurrence libre de x dans la formule. Une variable x est une* variable liée *d'une formule si et seulement s'il y a une occurrence liée de x dans la formule. Une formule sans variable libre est aussi appelée une* formule fermée.

Remarque 4.2.4 *Une variable peut-être à la fois libre et liée. Par exemple, dans la formule $\forall x P(x) \vee Q(x)$, x est à la fois libre et liée.*

Remarque 4.2.5 *Par définition, une variable qui n'apparaît pas dans une formule (0 occurrence) est une variable NON libre de la formule.*

Exemple 4.2.6 *Les variables libres de la formule de l'exemple 4.2.2 sont x et y.*

4.3 Sens des formules

Dans la suite, nous définissons en logique du premier ordre, le sens des connecteurs, des quantificateurs ainsi que de *l'égalité*. Aussi, pour définir le sens d'une formule, il suffit d'indiquer la valeur des variables libres et le sens de ses symboles utilisés. Dans un premier temps nous expliquons comment nous déclarons un symbole. Ensuite nous introduisons la notion de signature qui permet de définir les termes et formules associés. Une fois toutes ces notions établies nous définissons les interprétations afin de donner du sens aux formules de la logique du premier ordre.

4.3.1 Déclaration de symbole

Afin de donner un sens aux formules de la logique du premier ordre il nous faut déclarer les symboles utilisés. Nous distinguons en plus des variables, les symboles de fonctions, par exemple $g(x,y)$, et les symboles de relation comme $parent(x,y)$.

Définition 4.3.1 (Déclaration de symbole) *Une* déclaration de symbole *est un triplet noté s^{gn} où s est un symbole, g est soit f (pour fonction) ou r (pour relation), et n est un entier naturel dénotant le nombre d'arguments de ce symbole, n est aussi appelé l'*arité *de s.*

Exemple 4.3.2 *Les fonctions d'arité* 0 *constituent les constantes, par exemple* 1 *et* 2 *sont des constantes d'arité nulle, mais $parent^{r2}$ est une notation signifiant que parent est employé comme relation d'arité* 2*. De même $*^{f2}$ est une déclaration annonçant que $*$ est une fonction* 2*-aire. Le symbole A^{r0} est d'arité* 0 *et signifie que A est une variable propositionnelle. Le symbole $homme^{r1}$ est d'arité* 1 *ou* unaire *et le symbole g^{f2} est d'arité* 2 *ou* 2*-aire.*

Remarque 4.3.3 *Lorsque le contexte ou les conventions usuelles comportent une déclaration implicite d'un symbole, nous omettons l'exposant. Par exemple, le symbole égal étant toujours employé comme relation à deux arguments, nous abrégeons la déclaration de symbole $=^{r2}$ en $=$.*

4.3.2 Signature

En logique du premier ordre nous pouvons choisir le nom des variables utilisées mais nous avons aussi la possibilité de construire nos propres constantes (fonctions sans arguments), fonctions, variables propositionnelles (relations d'arité nulle) et relations. Une signature est donc l'ensemble des déclarations de symboles autorisés pour construire des formules. Elle permet de définir les symboles dont le sens n'est pas fixé *a priori*, contrairement par exemple aux constantes $\top$ et $\bot$ dont le sens est toujours fixé respectivement à 1 et 0.

Définition 4.3.4 (Signature, constante, symbole de fonction, variable propositionnelle, et relation) *Une* signature Σ *est un ensemble de déclarations de symboles de la forme s^{gn}. Soient n un entier strictement positif et Σ une signature, le symbole s est :*

1. *une* constante *de la signature si et seulement si $s^{f0} \in \Sigma$*
2. *un* symbole de fonction à *n* arguments *de la signature, si et seulement si $s^{fn} \in \Sigma$*
3. *une* variable propositionnelle *de la signature si et seulement si $s^{r0} \in \Sigma$*
4. *un* symbole de relation à *n* arguments *de la signature, si et seulement si $s^{rn} \in \Sigma$*

Au lieu de dire $0^{f0}, 1^{f0}, +^{f2}, *^{f2}, =^{r2}$ est une signature pour l'arithmétique, nous disons plus simplement qu'une signature possible pour l'arithmétique comporte 0, 1, $+$ (à deux arguments), $*$ et $=$. Sur cet exemple, nous devons préciser que le symbole *plus* est utilisé avec deux arguments (car nous pouvons aussi rencontrer le symbole *plus* avec un seul argument).

Exemple 4.3.5 *Une signature possible pour la théorie des ensembles est $\in, =$. Notons que toutes les autres opérations sur les ensembles peuvent être définies à partir de ces deux symboles.*

Nous définissons maintenant les termes associés à une signature. Nous considérons un ensemble dénombrable de variables disjoint de l'ensemble de symboles de la signature.

Définition 4.3.8 (Terme sur une signature) *Soit Σ une signature, un* terme *sur Σ est :*

— *soit une variable,*
— *soit une constante s où $s^{f0} \in \Sigma$,*

— *soit un terme de la forme* $s(t_1,\ldots,t_n)$*, où* $n \geq 1$*,* $s^{fn} \in \Sigma$ *et* $t_1,\ldots,t_n$ *sont des termes sur* Σ*.*

L'ensemble des termes sur la signature Σ est noté T_Σ.

Définition 4.3.9 (Formule atomique sur une signature) *Soit* Σ *une signature, une* formule atomique *sur* Σ *est :*
— *soit une des constantes* $\top, \bot$,
— *soit une variable propositionnelle s où* $s^{r0} \in \Sigma$,
— *soit de la forme* $s(t_1,\ldots,t_n)$ *où* $n \geq 1$*,* $s^{rn} \in \Sigma$ *et où* $t_1,\ldots,t_n$ *sont des termes sur* Σ*.*

Notons que les variables ne sont pas des formules atomiques.

Définition 4.3.10 (Formule sur une signature) *Une* formule *sur une signature* Σ *est une formule, dont les sous-formules atomiques sont des formules atomiques sur* Σ *(au sens de la définition 4.3.9).*

Nous dénotons l'ensemble des formules sur la signature Σ par F_Σ .

Exemple 4.3.11 $\forall x\ (p(x) \Rightarrow \exists y\ q(x,y))$ *est une formule sur la signature* $\Sigma = \{p^{r1}, q^{r2}, h^{f1}, c^{f0}\}$*. Mais c'est aussi une formule sur la signature* $\Sigma' = \{p^{r1}, q^{r2}\}$*, puisque les symboles h et c ne figurent pas dans la formule.*

Définition 4.3.12 (Signature associée à une formule) *La* signature associée *à une formule est la plus petite signature* Σ *telle que la formule est élément de* F_Σ*, c'est la plus petite signature permettant d'écrire la formule.*

Exemple 4.3.13 *La signature* Σ *associée à la formule* $\forall x\ (p(x) \Rightarrow \exists y\ q(x,y))$ *est* $\Sigma = \{p^{r1}, q^{r2}\}$*.*

Définition 4.3.14 (Signature associée à un ensemble de formules) *La* signature associée *à un ensemble de formules est l'union des signatures associées à chaque formule de l'ensemble.*

Exemple 4.3.15 *La signature* Σ *associée à l'ensemble constitué des deux formules* $\forall x (p(x) \Rightarrow \exists y\ q(x,y)), \forall u\ \forall v\ (u + s(v) = s(u) + v)$ *est* $\Sigma = \{p^{r1}, q^{r2}, +^{f2}, s^{f1}, =^{r2}\}$*.*

4.3.3 Interprétation

En logique propositionnelle, le sens des formules est uniquement fixé par les valeurs des variables, en logique du premier ordre le sens des formules dépend aussi du sens des fonctions et des relations. Le sens des fonctions et des relations est fixé par une interprétation.

Définition 4.3.16 (Interprétation) *Une* interprétation *I sur une signature* Σ *est définie par un domaine D non vide et une application qui à chaque déclaration de symbole* $s^{gn} \in \Sigma$ *associe sa valeur* s_I^{gn} *comme suit :*

1. s_I^{f0} *est un élément de D.*
2. s_I^{fn} *où* $n \geq 1$ *est une fonction de* D^n *dans D, autrement dit une fonction à n arguments.*
3. s_I^{r0} *vaut* 0 *ou* 1.
4. s_I^{rn} *où* $n \geq 1$ *est un sous-ensemble de* D^n*, autrement dit une relation à n arguments.*

Exemple 4.3.17 *Soit l'interprétation I de domaine* $D = \{1,2,3\}$ *où la relation binaire* ami *est vraie pour les couples* $(1,2)$*,* $(1,3)$ *et* $(2,3)$*, c'est-à-dire,* $ami_I^{r2} = \{(1,2),(1,3),(2,3)\}$*.* $ami(2,3)$ *est vraie dans l'interprétation I. En revanche,* $ami(2,1)$ *est fausse dans l'interprétation I.*

Remarque 4.3.18 *Dans* toute *interprétation I, la valeur du symbole* $=$ *est l'ensemble* $\{(d,d) \mid d \in D\}$*, autrement dit dans toute interprétation le sens de l'égalité est* l'identité *sur le domaine de l'interprétation.*

Exemple 4.3.19 *Considérons une signature suivante.*
— $Anne^{f0}$*,* $Bernard^{f0}$ *et* $Claude^{f0}$ *: les prénoms Anne, Bernard, et Claude dénotent des constantes,*
— a^{r2} *: la lettre a dénote une relation à deux arguments (nous lisons* $a(x,y)$ *comme x aime y) et*
— c^{f1} *: la lettre c dénote une fonction à un argument (nous lisons* $c(x)$ *comme le copain ou la copine de x).*
Une interprétation possible sur cette signature est l'interprétation I de domaine $D = \{0,1,2\}$ *où :*
— $Anne_I^{f0} = 0$*,* $Bernard_I^{f0} = 1$*, et* $Claude_I^{f0} = 2$*.*
— $a_I^{r2} = \{(0,1),(1,0),(2,0)\}$*.*

— $c_I^{f1}(0) = 1, c_I^{f1}(1) = 0, c_I^{f1}(2) = 2$. *Notons que la fonction* c_I^{f1} *a comme domaine D, ce qui oblige à définir artificiellement* $c_I^{f1}(2)$ *: Claude, dénoté par* 2*, n'a ni copain, ni copine.*

Définition 4.3.20 (Interprétation d'un ensemble de formules) L'interprétation d'un ensemble de formules *est une interprétation qui définit seulement le sens de la signature associée à l'ensemble des formules.*

Définition 4.3.21 (État) *Un* état *e d'une interprétation est une application de l'ensemble des variables dans le domaine de l'interprétation.*

Définition 4.3.22 (Assignation) *Une* assignation *est un couple* (I,e) *composé d'une interprétation I et d'un état e.*

Exemple 4.3.23 *Soient le domaine* $D = \{1,2,3\}$ *et l'interprétation I où la relation binaire* ami *est vraie uniquement pour les couples* $(1,2)$*,* $(1,3)$ *et* $(2,3)$*, c'est-à-dire,* $ami_I^{r2} = \{(1,2),(1,3),(2,3)\}$*. Soit e l'état qui associe* 2 *à x et* 1 *à y. L'assignation* (I,e) *rend la relation* $ami(x,y)$ *fausse.*

4.3.4 Sens des formules

Nous expliquons maintenant comment évaluer une formule à partir d'une assignation, c'est-à-dire, une interprétation et un état. Il faut noter que dans certains cas, l'état de l'assignation est inutile pour fixer le sens d'une formule.

Remarque 4.3.24 *La valeur d'une formule ne dépend que de ses variables libres et de ses symboles, aussi pour évaluer une formule sans variable libre, l'état des variables est inutile. Nous avons alors deux possibilités :*

— *Pour une formule* sans variables libres*, il suffit de donner une interprétation I des symboles de la formule. Dans ce cas, les assignations* (I,e) *et* (I,e') *donnerons la même valeur à la formule pour tous états e et* e'*. Ainsi pour tout état e, nous identifierons* (I,e) *et I. Selon le contexte, I sera considéré comme soit une interprétation soit une assignation dont l'état est quelconque.*

— *Pour une formule* avec des variables libres*, nous avons donc besoin d'une assignation.*

Soient Σ une signature, I une interprétation sur Σ de domaine D et e un état de cette interprétation. Nous souhaitons connaître la valeur (0 ou 1) de toute formule A sur Σ dans l'assignation (I,e). Cette valeur sera notée $[A]_{(I,e)}$. Pour cela, nous avons tout d'abord besoin de définir le sens de chaque terme t sur Σ dans l'assignation (I,e), noté $[\![t]\!]_{(I,e)}$. Ensuite nous devons fixer le sens de chaque formule atomique B sur Σ dans la même assignation, noté $[B]_{(I,e)}$.

4.3.4.1 Sens des termes sur une signature

Chacun sait intuitivement comment évaluer un terme : nous remplaçons les variables par leurs valeurs, les symboles de fonctions par les fonctions qui leur sont associées et nous appliquons les fonctions. Mais pour raisonner sur le sens des termes ou écrire un programme d'évaluation des termes, nous devons formaliser cette évaluation.

Définition 4.3.25 (Évaluation) *Nous donnons la définition inductive de l'évaluation d'un terme t :*

1. *si t est une variable, alors* $[\![t]\!]_{(I,e)} = e(t)$*,*
2. *si t est une constante alors* $[\![t]\!]_{(I,e)} = t_I^{f0}$*,*
3. *si* $t = s(t_1,\dots,t_n)$ *où s est un symbole et* $t_1,\dots,t_n$ *sont des termes, alors* $[\![t]\!]_{(I,e)} = s_I^{fn}([\![t_1]\!]_{(I,e)},\dots,[\![t_n]\!]_{(I,e)})$.

Dans les exemples, nous remplaçons les variables par leurs valeurs, nous confondons les symboles et leur sens.

Exemple 4.3.26 *Soit la signature* a^{f0}, f^{f2}, g^{f2}.
Soit I l'interprétation de domaine $\mathbb{N}$ *où :*
— $a_I = 1$ *;*
— f_I^{f2} *est le produit ;*
— g_I^{f2} *est la somme.*
Soit e l'état tel que $x = 2, y = 3$*. Calculons* $[\![f(x,g(y,a))]\!]_{(I,e)}$.

4.3.4.2 Sens des formules atomiques sur une signature

Définition 4.3.27 (Sens des formules atomiques) *Le sens des formules atomiques est donné par les règles inductives suivantes :*

1. $[\top]_{(I,e)} = 1, [\bot]_{(I,e)} = 0$. *Dans les exemples, nous autorisons à remplacer* $\top$ *par sa valeur* 1 *et* $\bot$ *par sa valeur* 0.
2. *Soit s une variable propositionnelle,* $[s]_{(I,e)} = s_I^{r0}$.
3. *Soit* $A = s(t_1, \dots, t_n)$ *où s est un symbole et* $t_1, \dots, t_n$ *sont des termes. Si* $([\![t_1]\!]_{(I,e)}, \dots, [\![t_n]\!]_{(I,e)}) \in s_I^{rn}$ *alors* $[A]_{(I,e)} = 1$ *sinon* $[A]_{(I,e)} = 0$.

Remarque 4.3.28 *Nous devons distinguer entre évaluer un terme* $[\![\ ]\!]$ *et évaluer une formule atomique* $[\ ]$*, car dans notre présentation, ces deux catégories syntaxiques ne sont pas disjointes, car* $s(t_1, \dots, t_n)$ *peut être l'application d'une fonction (*$[\![]\!]$*) ou d'une relation (*$[\ ]$*). Mais quand le contexte est sans ambiguïté, nous pouvons omettre cette distinction.*

Exemple 4.3.29 *En reprenant l'exemple 4.3.19 page 79, nous obtenons :*

— $[a(Anne, Bernard)]_I =$

— $[a(Anne, Claude)]_I =$

Soit e l'état $x = 0, y = 2$. *Nous avons :*

— $[a(x, c(x))]_{(I,e)} =$

— $[a(y, c(y))]_{(I,e)} =$

Attention à distinguer (suivant le contexte), les éléments du domaine 0, 1 *et les valeurs de vérité* 0, 1. *Les exemples suivants mettent en évidence l'interprétation de l'égalité comme une identité. Dans l'interprétation I, nous avons :*

— $[(Anne = Bernard)]_I =$

— $[(c(Anne) = Anne)]_I =$

— $[(c(c(Anne)) = Anne)]_I =$

4.3.4.3 Sens des formules sur une signature

Nous pouvons maintenant ajouter le sens des quantificateurs et aussi celui des opérateurs qui reste identique à celui donné pour la logique propositionnelle (*cf.* sous-section 1.2.1 page 14).

Définition 4.3.30 (Sens des formules) *Le sens des formules est donné par :*

1. *Les connecteurs propositionnels ont le même sens qu'en logique propositionnelle. Soient B et C des formules, rappelons uniquement le sens de l'implication : si* $[B]_{(I,e)} = 0$ *alors* $[(B \Rightarrow C)]_{(I,e)} = 1$ *sinon* $[(B \Rightarrow C)]_{(I,e)} = [C]_{(I,e)}$.

2. *Soient x une variable et B une formule.* $[\forall xB]_{(I,e)} = 1$ *si et seulement si* $[B]_{(I,f)} = 1$ pour tout *état f identique à e, sauf pour x. Soit* $d \in D$*. Notons* $e[x = d]$ *l'état identique à l'état e, sauf pour la variable x, auquel l'état* $e[x = d]$ *associe la valeur d. La définition ci-dessus peut être mise sous la forme suivante :*

$$[\forall xB]_{(I,e)} = min_{d \in D}[B]_{(I,e[x=d])} = \prod_{d \in D}[B]_{(I,e[x=d])},$$

où le produit est le produit booléen.
Cette définition permet de calculer la valeur de $\forall xB$ *dans le cas où D est un domaine fini. Mais quand D n'est pas un domaine fini, c'est seulement une traduction du quantificateur universel d'un formalisme dans un autre.*

3. $[\exists xB]_{(I,e)} = 1$ *si et seulement s'*il y a *un état f identique à e, sauf pour x, tel que* $[B]_{(I,f)} = 1$*. La définition ci-dessus peut être mise sous la forme suivante :*

$$[\exists xB]_{(I,e)} = max_{d \in D}[B]_{(I,e[x=d])} = \sum_{d \in D}[B]_{(I,e[x=d])},$$

où la somme est la somme booléenne.

Exemple 4.3.31 *Soit l'interprétation I de domaine* $D = \{1,2,3\}$ *où la relation binaire* ami *est vraie pour les couples* $(1,2)$*,* $(1,3)$ *et* $(2,3)$*, c'est-à-dire,* $ami_I^{r2} = \{(1,2),(1,3),(2,3)\}$*. La formule* $ami(1,2) \wedge ami(2,3) \Rightarrow ami(1,3)$ *est vraie dans l'interprétation I,* i.e., $[ami(1,2) \wedge ami(2,3) \Rightarrow ami(1,3)]_I = 1$.

Exemple 4.3.32 *Utilisons l'interprétation I donnée dans l'exemple 4.3.19 page 79. D'après le sens du quantificateur existentiel, nous devons évaluer* $a(x,x)$ *pour* $x = 0$*,* $x = 1$*, et* $x = 2$*. Nous simplifions ce calcul et les calculs suivants en remplaçant immédiatement les variables par leurs valeurs : cela nous évite d'utiliser les états.*

— $[\exists x\ a(x,x)]_I =$

— $[\forall x \exists y\ a(x,y)]_I = \min\{\max\{[a(0,0)]_I, [a(0,1)]_I, [a(0,2)]_I\}, \max\{[a(1,0)]_I, [a(1,1)]_I, [a(1,2)]_I\}, \max\{[a(2,0)]_I, [a(2,1)]_I, [a(2,2)]_I\}\} = \min\{\max\{0,1,0\}, \max\{1,0,0\}, \max\{1,0,0\}\} = \min\{1,1,1\} = 1$.
D'après la définition, nous avons : $[\forall x \exists y\ a(x,y)]_I = ([a(0,0)]_I + [a(0,1)]_I + [a(0,2)]_I).\ ([a(1,0)]_I + [a(1,1)]_I + [a(1,2)]_I).\ ([a(2,0)]_I + [a(2,1)]_I + [a(2,2)]_I) = (0+1+0).(1+0+0).(1+0+0) = 1.1.1 = 1$.

— $[\exists y \forall x\ a(x,y)]_I =$

Remarque 4.3.33 *Dans l'interprétation ci-dessus, les formules* $\forall x \exists y\ a(x,y)$ *et* $\exists y \forall x\ a(x,y)$ *n'ont pas la même valeur. Donc en intervertissant un quantificateur existentiel et un quantificateur universel, nous ne préservons pas le sens des formules.*

4.3.5 Modèle, validité, conséquence, équivalence

Ces notions sont définies comme en logique propositionnelle. Mais alors qu'en logique propositionnelle, une assignation est une application des variables propositionnelles dans $0,1$, en logique du premier ordre une assignation est un couple constitué d'une interprétation des symboles d'une part et de l'état des variables d'autre part.

4.3.6 Instanciation

Définition 4.3.34 (Instanciation) *Soit x une variable, t un terme et A une formule.*

1. $A{<}x := t{>}$ *est la formule obtenue en remplaçant dans la formule A toute occurrence libre de x par le terme t.*

2. *Le terme t est libre pour x dans A si les variables de t ne sont pas liées dans les occurrences libres de x dans A.*

Exemple 4.3.35 *Le terme z est libre pour x dans la formule $\exists yp(x,y)$. Par contre le terme y, comme tout terme comportant la variable y, n'est pas libre pour x dans cette formule. Soit A la formule $(\forall xP(x) \vee Q(\mathbf{x}))$, la formule $A{<}x := b{>}$ vaut*

Théorème 4.3.36 *Soient A une formule et t un terme libre pour la variable x dans A. Soient I une interprétation et e un état de l'interprétation. Nous avons $[A{<}x := t{>}]_{(I,e)} = [A]_{(I,e[x=d])}$, où $d = [\![t]\!]_{(I,e)}$.*

Autrement dit, la valeur de $A{<}x := t{>}$ dans une assignation est la même que celle de A dans une assignation identique, sauf qu'elle donne à x la valeur du terme t. Ce théorème, dont le résultat est évident, peut être prouvé par une récurrence (que nous ne ferons pas) sur la taille des formules. Nous montrons seulement que la condition sur t est indispensable en observant un exemple où cette condition n'est pas respectée.

Exemple 4.3.37 *Soient I l'interprétation de domaine $\{0,1\}$ avec $p_I = \{(0,1)\}$ et e, un état où $y = 0$. Soient A la formule $\exists yp(x,y)$ et t le terme y.* Ce terme n'est pas libre pour x dans A. *Nous avons :*

— $A{<}x := y{>} =$

et $[A{<}x := y{>}]_{(I,e)} =$

— *Soit $d = [\![t]\!]_{(I,e)} = [\![y]\!]_{(I,e)} = 0$. Dans l'assignation $(I, e[x = d])$, nous avons $x = 0$. Donc, nous obtenons :* $[A]_{(I,e[x=d])} =$

Ainsi, $[A{<}x := t{>}]_{(I,e)} \neq [A]_{(I,e[x=d])}$, pour $d = [\![t]\!]_{(I,e)}$.

Corollaire 4.3.38 *Soient A une formule et t un terme libre pour x dans A. Les formules $\forall xA \Rightarrow A{<}x := t{>}$ et $A{<}x := t{>} \Rightarrow \exists xA$ sont valides.*

Ce corollaire est une conséquence immédiate du théorème précédent.

4.3.7 Interprétation finie

Nous montrons comment rechercher des modèles finis de formules fermées (c'est-à-dire sans variables libres). Un modèle fini d'une formule fermée est une interprétation de la formule de domaine fini, qui rend vraie la formule. Il est clair que le nom des éléments du domaine est sans importance, aussi quand nous cherchons un modèle avec un domaine de n éléments, le domaine que nous utiliserons, sera celui des entiers (sous-entendu naturels) inférieurs à n. Pour savoir si une formule fermée a un modèle de domaine $\{0,\ldots,n-1\}$, il suffit d'énumérer toutes les interprétations possibles de la signature associée à la formule et d'évaluer la formule pour ces interprétations. Mais cette méthode est inutilisable en pratique, car le nombre de ces interprétations est énorme. Soit une signature avec une constante, un symbole de fonction à un argument et un symbole de relation à deux arguments (plus éventuellement l'égalité de sens fixé), sur un domaine à 5 éléments, cette signature à $5 \times 5^5 \times 2^{25} = 524288000000$ interprétations. Aussi nous montrons d'abord, comment dans le cas *où une formule n'a aucun symbole de fonction et aucune constante, sauf des représentations d'entiers inférieurs à n*, nous pouvons rechercher ses modèles à n éléments par réduction au cas propositionnel. En présence de symboles de fonctions et de constantes, nous pouvons en énumérer leurs valeurs, puis appliquer les méthodes ci-dessous. Pour trouver des énumérations intelligentes, nous conseillons de lire le manuel de Prover9 [22].

4.3.7.1 Les entiers et leurs représentations

Nous distinguons dans la suite un entier n et sa représentation $\underline{n}$. Par exemple l'entier 3 peut être représenté en base 10 par 3, en base 2 par 11, en chiffres romains par III, en chiffre grec par γ. Par habitude, nous choisissons la représentation en base 10. Dans la suite, de façon implicite, toutes les interprétations considérées donnent à la représentation d'un entier, la valeur de l'entier représenté.

4.3.7.2 Expansion d'une formule

Dans certains cas il est suffisant de regarder les expansions de taille finie afin de trouver un modèle ou un contre-modèle pour une formule donnée.

Définition 4.3.39 (n-expansion) *Soient A une formule et n un entier. La n-*expansion de *A est la formule qui consiste à remplacer toute sous-formule de A de la forme $\forall xB$ par la conjonction $(\prod_{i<n} B<x := \underline{i}>)$ et toute sous-formule de A de la forme $\exists xB$ par la disjonction $(\sum_{i<n} B<x := \underline{i}>)$ où $\underline{i}$ est la représentation décimale de l'entier i.*

Exemple 4.3.40 *La 2-expansion de la formule $\exists xP(x) \Rightarrow \forall xP(x)$ est la formule*

Théorème 4.3.41 *Soient n un entier et A une formule ne comportant que des représentations d'entiers de valeur inférieure à n. Soit A' la n-expansion de A. Toute interprétation de domaine $\{0,\dots,n-1\}$ attribue la même valeur à A et à A'.*

La condition sur A est nécessaire car si A comporte une représentation d'un entier au moins égal à n, la valeur de cette représentation ne sera pas dans le domaine de l'interprétation. La preuve du théorème est une récurrence sur la taille des formules, que nous ne ferons pas. Nous nous contentons de montrer que l'élimination d'un quantificateur universel de la formule $\forall xB$ conserve la valeur de cette formule.

Soient (I,e) une interprétation et un état de domaine $\{0,\dots,n-1\}$ donnant à la représentation d'un entier, la valeur de l'entier représenté. Par définition du sens du quantificateur universel : $[\forall xB]_{(I,e)} = \prod_{i<n}[B]_{(I,e[x=i])}$. D'après le théorème 4.3.36 page précédente et le fait que la valeur de la représentation de l'entier i est i, nous avons : $[B]_{(I,e[x=i])} = [B< x := \underline{i}>]_{(I,e)}$. Par suite : $[\forall xB]_{(I,e)} = \prod_{i<n}[B<x := \underline{i}>]_{(I,e)} = [\prod_{i<n} B<x := \underline{i}>]_{(I,e)}$.

4.3.7.3 Interprétation et assignation propositionnelle

Soient n un entier et A une formule fermée, sans quantificateur, sans égalité, sans symbole de fonction, sans constante sauf des représentations d'entiers inférieurs à n. Soit P l'ensemble des formules atomiques de A (sauf $\top$ et $\bot$ dont le sens est fixé).

De l'assignation à l'interprétation.

Théorème 4.3.42 *Soit v une assignation propositionnelle de P dans $\{0,1\}$ alors il existe une interprétation I de A telle que $[A]_I = [A]_v$.*

Preuve : Soit v une assignation propositionnelle de P dans $\{0,1\}$. Soit I l'interprétation suivante de A :

1. Domaine : $\{0,\dots,n-1\}$
2. $s_I^{r0} = v(s)$ si et seulement si $s \in P$
3. Soit $p \geq 1$ et s^{rp} un symbole de relation de A . Alors $s_I^{rp} = \{(k_1,\dots,k_p) | s(\underline{k_1},\dots,\underline{k_p}) \in P$ et $v(s(\underline{k_1},\dots,\underline{k_p})) = 1\}$

Par définition de I, l'assignation v et l'interprétation I donnent la même valeur aux sous-formules atomiques de A, donc (par récurrence sur les formules) la même valeur à la formule A. □

Exemple 4.3.43 *L'assignation v, définie par $p(0) = 1$ et $p(1) = 0$, donne la valeur 0 à la formule suivante :*

$$(p(0)+p(1)) \Rightarrow (p(0).p(1)).$$

Donc I définie par $p_I = \{0\}$ donne aussi la valeur 0 à cette même formule. Cet exemple montre que v et I sont deux façons analogues de présenter une interprétation, la deuxième étant souvent plus concise.

De l'interprétation à l'assignation.

Théorème 4.3.44 *Soit I une interprétation de A alors il existe une assignation v de P telle que*

$$[A]_I = [A]_v.$$

Preuve : Soient I une interprétation de A et v l'assignation propositionnelle suivante de P dans $\{0,1\}$: pour tout $B \in P$, $v(B) = [B]_I$. Par définition de v, l'assignation v et l'interprétation I donnent la même valeur aux sous-formules atomiques de A, donc la même valeur à la formule A. □

4.3.7.4 Recherche d'un modèle fini d'une formule fermée

Formule fermée sans symbole de fonction : Soit A une formule fermée sans symbole de fonction ni constantes, sauf des représentations d'entiers de valeur inférieure à n. Pour trouver une interprétation I modèle de A de domaine $\{0,\dots,n-1\}$ donnant à la représentation d'un entier, la valeur de l'entier représenté, nous procédons ainsi :

1. Nous remplaçons A par sa n-expansion B.
2. Dans la formule B, nous remplaçons les égalités par leur valeur, c'est-à-dire que $\underline{i} = \underline{j}$ est remplacée par 0 si $i \neq j$ et par 1 si $i = j$. De plus, nous recommandons d'éliminer ces valeurs de vérité par les identités $x+0=x$, $x+1=1$, $x*0=0$, $x*1=x$. Soit C la formule obtenue après ces remplacements et simplifications.
3. Nous cherchons une assignation propositionnelle v des formules atomiques de C, qui soit modèle de C : si une telle assignation n'existe pas, A n'a pas de modèle, sinon l'interprétation I déduite de v comme il est indiqué dans le théorème 4.3.42 page précédente est modèle de A.

Prouvons la correction de cette méthode :

1. Supposons qu'il n'y ait pas d'assignation propositionnelle modèle de C, mais que A ait un modèle I. D'après le théorème 4.3.41 page ci-contre, I est modèle de B, donc de C, et d'après le théorème 4.3.44 page précédente, il y a une assignation propositionnelle modèle de C. De cette contradiction, nous déduisons que A n'a pas de modèle à n éléments.
2. Supposons que l'assignation propositionnelle v soit modèle de C. Donc, l'interprétation I construite comme il est indiqué dans le théorème 4.3.42 page ci-contre est modèle de C, donc elle est modèle de B, donc aussi d'après le théorème 4.3.41 page précédente, elle est modèle de A.

Exemple 4.3.45 *Soit A la formule $\exists x P(x) \land \exists x \neg P(x) \land \forall x \forall y (P(x) \land P(y) \Rightarrow x = y)$. Il est clair que cette formule n'a pas de modèle à un élément car cet élément devrait vérifier à la fois la propriété P et sa négation. La 2-expansion de A est :*

Formule fermée avec symbole de fonction : Soit A une formule fermée pouvant comporter des représentations d'entiers de valeur inférieure à n. Comme dans le cas précédent, nous remplaçons A par son expansion, nous supprimons les égalités. Puis, nous énumérons les choix des valeurs des symboles comme dans *l'algorithme* `DPLL`, en propageant le plus possible chacun des choix effectués.

Exemple 4.3.46 *Soit A la formule $\exists y\ P(y) \Rightarrow P(a)$, dont nous cherchons un contre-modèle à 2 éléments (cela revient au même que de trouver un modèle de la négation de A).*

Exemple 4.3.47 *Nous cherchons un modèle à 2 éléments des formules $P(a)$, $\forall x(P(x) \Rightarrow P(f(x)))$, $\neg P(f(b))$.*

4.3.8 Substitution et remplacement

La propriété de substitution et la propriété de remplacement déjà énoncées en logique propositionnelle (voir 1.3.1 page 19 et 1.3.10 page 21) s'étendent à la logique du premier ordre [3]. Plus précisément l'application d'une substitution à une formule *propositionnellement* valide donne une formule valide. Par exemple soit σ la substitution propositionnelle telle que $\sigma(p) = \forall x\ q(x)$. Puisque la formule $p \vee \neg p$ est valide, il en est de même de la formule $\sigma(p \vee \neg p) = \forall x\ q(x) \vee \neg \forall x\ q(x)$. Le principe de remplacement s'étend aussi avec le même énoncé de la logique propositionnelle à la logique du premier ordre car il est déduit des propriétés élémentaires suivantes. Pour toutes formules A et B et toute variable x :

— $(A \Leftrightarrow B) \models (\forall xA \Leftrightarrow \forall xB)$.
— $(A \Leftrightarrow B) \models (\exists xA \Leftrightarrow \exists xB)$.

4.4 Équivalences remarquables

Dans cette section, nous donnons les règles permettant de simplifier les formules de la logique du premier ordre. Remarquons que les règles de simplifications de la logique propositionnelle sont bien entendu encore appliquables au premier ordre.

4.4.1 Relation entre $\forall$ et $\exists$

Lemme 4.4.1 *Soient A une formule et x une variable.*

1. $\neg \forall xA \equiv \exists x \neg A$.

3. Notons que dans le livre de Stephen Cole Kleene [19], cette notion de substitution sur les variables propositionnelles est étendue à une substitution plus générale applicable à tous les symboles de relation.

2. $\forall xA \equiv \neg\exists x\neg A$.
3. $\neg\exists xA \equiv \forall x\neg A$.
4. $\exists xA \equiv \neg\forall x\neg A$.

Prouvons les deux premières identités, les deux autres sont données en exercice 78 page 92 :

1. Soient I une interprétation de domaine D et e un état.

$$
\begin{array}{lll}
[\neg\forall xA]_{(I,e)} & \equiv & 1-[\forall xA]_{(I,e)} \\
 & \equiv & 1-\prod_{d\in D}[A]_{(I,e[x=d])} \quad \text{d'après le sens de } \forall \\
 & \equiv & \sum_{d\in D}\left(1-[A]_{(I,e[x=d])}\right) \quad \text{par les lois de De Morgan généralisées} \\
 & \equiv & \sum_{d\in D}[\neg A]_{(I,e[x=d])} \quad \text{par définition du sens de } \neg \\
 & \equiv & [\exists x\neg A]_{(I,e)} \quad \text{par définition du sens de } \exists
\end{array}
$$

En examinant de façon critique cette preuve, nous voyons qu'elle utilise la loi de De Morgan généralisée, qui est l'expression dans un autre formalisme d'une loi analogue. Il faut donc formaliser l'usage des quantificateurs pour échapper à ces preuves qui ne sont en fait que des changements de notations. Nous le ferons en ajoutant à la déduction naturelle des règles qui gouvernent les quantificateurs.

2. Les autres équivalences sont prouvables à partir de la première par le principe de remplacement. Nous ne prouvons que la deuxième identité :

$$
\begin{array}{lll}
\forall xA & \equiv & \neg\neg\forall xA \quad \text{identité de la double négation} \\
 & \equiv & \neg\exists x\neg A \quad \text{par l'identité 1 page précédente}
\end{array}
$$

4.4.2 Déplacement des quantificateurs

Nous donnons, sans preuve, des équivalences qui s'appliquent à toutes les formules (avec ou sans variable libre) et qui permettent de retrouver les lois usuelles de déplacement des quantificateurs. Nous supposons ci-dessous que x, y sont des variables et que A, B sont des formules.

1. $\forall x\forall yA \equiv \forall y\forall xA$.
2. $\exists x\exists yA \equiv \exists y\exists xA$.
3. $\forall x(A\wedge B) \equiv (\forall xA \wedge \forall xB)$.
4. $\exists x(A\vee B) \equiv (\exists xA \vee \exists xB)$.
5. Soient Q un des quantificateurs $\forall, \exists$, et $\circ$ une des opérations $\wedge, \vee, \Rightarrow$. Supposons que x ne soit pas une variable libre de A.
   (a) $QxA \equiv A$,
   (b) $Qx(A\circ B) \equiv (A\circ QxB)$.

Exemple 4.4.2 *Nous éliminons de ces deux formules les quantificateurs inutiles :*
— $\forall x\exists xP(x) \equiv$

— $\forall x(\exists xP(x)\vee Q(x)) \equiv$

4.4.3 Changement de variables liées

Théorème 4.4.3 *Soit Q un des quantificateurs $\forall, \exists$. Supposons que y soit une variable qui ne figure pas dans QxA alors : $QxA \equiv QyA$<$x := y$>.*

Exemple 4.4.4 *La formule $\forall x\ p(x,z)$ est équivalente à la formule $\forall y\ p(y,z)$ (d'après le théorème ci-dessus) mais elle n'est pas équivalente à la formule $\forall z\ p(z,z)$, où le changement de variable ne respecte pas les conditions du théorème.*

Définition 4.4.5 (Formules égales à un changement près de variables liées) *Deux formules sont* égales à un changement près de variables liées *si nous pouvons obtenir l'une à partir de l'autre par des remplacements de sous-formules de la forme QxA par $QyA<x := y>$ où Q est un quantificateur et y est une variable qui ne figure pas dans QxA. Deux formules égales à un changement près de variables liées, sont dites aussi copies l'une de l'autre ou encore* α-équivalentes*.*

Théorème 4.4.6 *Si deux formules sont égales à un changement près de variables liées alors elles sont équivalentes.*

Ce résultat est clairement une conséquence du théorème 4.4.3 page précédente. Nous n'en ferons pas la preuve, qui est longue et fastidieuse. Nous nous contentons de suggérer cette preuve à l'aide d'un exemple.

Exemple 4.4.7 *Nous montrons que les formules $\forall x \exists y P(x,y)$ et $\forall y \exists x P(y,x)$ sont égales par changement des variables liées, au sens de la définition 4.4.3 page précédente donc sont équivalentes, en effet :*

Notre définition de l'égalité entre deux formules à un changement près de variables liées n'est pas pratique, car la définition ne donne pas un test. Or il est simple de voir si deux formules sont égales à un changement près de variables liées : nous traçons des traits entre chaque quantificateur et les variables qu'il lie et nous effaçons les noms des variables liées. Si après cette transformation, les deux formules deviennent identiques, c'est qu'elles sont égales à un changement près des variables liées.

Exemple 4.4.8 *Les deux formules $\forall x \exists y P(y,x)$ et $\forall y \exists x P(x,y)$ sont transformées en* $\forall|\exists|P(|,|)$.

4.5 Exercices

Exercice 64 (Structure et variables libres)
Pour chaque formule ci-dessous, indiquer sa structure et ses variables libres.

1. $\forall x(P(x) \Rightarrow \exists y Q(x,y))$.
2. $\forall a \forall b(b \neq 0 \Rightarrow \exists q \exists r(a = b * q + r \wedge r < b))$ [4].
3. $Pair(x) \Leftrightarrow \exists y(x = 2 * y)$.
4. x *Divise* $y \Leftrightarrow \exists z(y = z * x)$.
5. $Premier(x) \Leftrightarrow \forall y(y$ *Divise* $x \Rightarrow y = 1 \vee y = x)$.

□

Exercice 65 (Formalisation, symbole de fonction et de relation) *Nous considérons* $\Sigma = \{f^{r2}, o^{r2}, c^{r2}, j^{r2}, r^{f0}, p^{f1}\}$ *la signature ayant la sémantique donnée ci-dessous.*

— $f(x,y)$: *x est frère de y.*
— $o(x,y)$: *x est l'oncle de y.*
— $c(x,y)$: *x est le cousin de y.*
— $j(x,y)$: *x est plus jeune que y.*
— *r est le diminutif de Robert.*
— $p(x)$ *est le père de x.*

Exprimer en logique du premier ordre et en utilisant la signature Σ *les phrases suivantes :*

1. *Tout frère du père de Robert est un oncle de Robert.*
2. *Si les pères de deux enfants sont des frères alors ces deux enfants sont des cousins.*
3. *Robert a un cousin plus jeune qu'un des frères de Robert.*

Exprimer en français les propositions logiques suivantes :

1. $\nexists x\ j(p(x), x)$
2. $\forall x \forall y(p(p(x)) = p(p(y)) \Rightarrow c(x,y))$
3. $\forall x \exists y(f(x,y) \wedge j(x,y))$
4. $\exists x \exists y(f(x,y) \wedge \neg(p(x) = p(y)))$

□

Exercice 66 (Formalisation) *Considérons la signature* $\Sigma = \{a^{f0}, f^{f0}, J^{r2}, G^{r2}\}$, *où les symboles ont le sens donné ci-dessous.*

— a : *l'équipe d'Allemagne.*
— f : *l'équipe de France.*
— $J(x,y)$: *x a joué un match contre y.*
— $G(x,y)$: *x a gagné contre y.*

Exprimer en logique du premier ordre en utilisant la signature Σ *les assertions suivantes :*

1. *L'équipe de France a gagné un match et en a perdu un.*
2. *L'équipe de France et l'équipe d'Allemagne ont fait match nul.*
3. *Une équipe a gagné tous ses matchs.*
4. *Aucune équipe n'a perdu tous ses matchs.*
5. *Considérons l'assertion suivante : « Tous ceux qui ont joué contre une équipe qui a gagné tous ses matchs, ont gagné au moins un match ». Parmi les formules suivantes, lesquelles expriment la phrase ci-dessus, et lesquelles sont équivalentes ?*
   (a) $\forall x \exists y (J(x,y) \wedge \forall z(J(y,z) \Rightarrow G(y,z)) \Rightarrow \exists v G(x,v))$.
   (b) $\forall x(\exists y (J(x,y) \wedge \forall z(J(y,z) \Rightarrow G(y,z))) \Rightarrow \exists v G(x,v))$.
   (c) $\exists x(\forall y (J(x,y) \Rightarrow G(x,y)) \Rightarrow \forall z(J(x,z) \Rightarrow \exists v G(x,v)))$.
   (d) $\forall x \forall y (J(x,y) \wedge \forall z(J(y,z) \Rightarrow G(y,z)) \Rightarrow \exists v G(x,v))$.
   (e) $\forall x(\forall y (J(x,y) \wedge \forall z(J(y,z) \Rightarrow G(y,z))) \Rightarrow \exists v G(x,v))$.

4. Afin de respecter la notation usuelle pour la division euclidienne nous prenons exceptionnellement *a*, *b*, *q* et *r* comme nom de variable.

□

Exercice 67 (Formaliser,)** *Soient les constantes s pour Serge, t pour Tobby et les symboles de relation $A(x,y)$, x aime y, $C(x)$, x est un chien, $D(x)$, x est un animal domestique, $E(x)$, x est un enfant, $O(x)$, x est un oiseau et $P(x,y)$, x a peur de y, donner la signature Σ associée à ces symboles. Formaliser en logique du premier ordre en utilisant Σ les énoncés suivants :*

1. *Les chiens et les oiseaux sont des animaux domestiques.*
2. *Tobby est un chien qui aime les enfants.*
3. *Les oiseaux n'aiment pas les chiens.*
4. *Serge aime tous les animaux domestiques sauf les chiens.*
5. *Tous les enfants n'ont pas peur des chiens.*
6. *Certains chiens aiment les enfants.*
7. *Certains chiens aiment les enfants et réciproquement.*
8. *Les enfants aiment certains chiens.*

□

Exercice 68 (Évaluer des prédicats unaires) *Soit I l'interprétation de domaine $D = \{0,1\}$ telle que $P_I = \{0\}, Q_I = \{1\}$.*

1. *Évaluer dans cette interprétation les formules $\forall x P(x)$ et $\forall x(P(x) \vee Q(x))$.*
2. *Les formules $\forall x P(x) \vee \forall x Q(x)$ et $\forall x(P(x) \vee Q(x))$ sont-elles équivalentes ?*
3. *Évaluer dans cette interprétation les formules $\exists x P(x)$ et $\exists x(P(x) \wedge Q(x))$.*
4. *Les formules $\exists x P(x) \wedge \exists x Q(x)$ et $\exists x(P(x) \wedge Q(x))$ sont-elles équivalentes ?*
5. *Évaluer dans cette interprétation les formules $\forall x(P(x) \Rightarrow Q(x))$ et $\forall x P(x) \Rightarrow \forall x Q(x)$.*
6. *Les deux formules $\forall x(P(x) \Rightarrow Q(x))$ et $\forall x P(x) \Rightarrow \forall x Q(x)$ sont-elles équivalentes ?*

□

Exercice 69 (Interprétation,*) *Nous donnons les formules suivantes :*

1. $\forall x \exists y (y = x+1)$.
2. $\exists y \forall x (y = x+1)$.
3. $\forall x \exists y (y = x+1) \Rightarrow \exists y \forall x (y = x+1)$.
4. $\forall x \exists y (x = y+1)$.
5. $\exists x \forall y (y = x+y)$.
6. $\exists x (x \neq 0 \wedge x+x = x)$.

Nous donnons les interprétations suivantes :

1. *I1 est l'algèbre de Boole sur $\{0,1\}$.*
2. *I2 est l'arithmétique usuelle sur les entiers naturels.*
3. *I3 est l'arithmétique usuelle sur les entiers relatifs.*
4. *I4 est l'algèbre de Boole de domaine $\mathcal{P}(X)$ où les constantes 0 et 1 dénotent les ensembles $\emptyset$ et X et où l'addition est l'union d'ensembles.*

Indiquer si ces interprétations sont des modèles ou des contre-modèles des formules ci-dessus. □

Exercice 70 (Prédicat unaire et égalité) *Soit I l'interprétation de domaine $D = \{0,1,2\}$ telle que : $P_I = \{0,1\}, Q_I = \{1,2\}, R_I = \{\}$. Évaluer dans cette interprétation les formules suivantes :*

1. $\exists x R(x)$.
2. $\forall x (P(x) \vee Q(x))$.
3. $\forall x (P(x) \Rightarrow Q(x))$.
4. $\forall x (R(x) \Rightarrow Q(x))$.
5. $\exists x (P(x) \wedge \forall y (P(y) \Rightarrow x = y))$ [5].

5. Cette formule signifie qu'il y a un et un seul élément vérifiant *P*.

6. $\exists x(P(x) \wedge Q(x) \wedge \forall y(P(y) \wedge Q(y) \Rightarrow x = y))$.

□

Exercice 71 (Évaluation, égalité) *Nous utilisons le symbole de fonction unaire f et la constante a. Nous abrégeons $\neg(x = y)$ en $x \neq y$. Soient I_1, I_2 les interprétations suivantes de domaine $\{0, 1, 2\}$: $a_{I_1} = a_{I_2} = 0$.*

x	$f_{I_1}(x)$	$f_{I_2}(x)$
0	0	1
1	0	2
2	2	0

Dans l'interprétation I_1 puis dans I_2, évaluer les formules suivantes :

1. $f(a) = a$.
2. $f(f(a)) = a$.
3. $f(f(f(a)) = a$.
4. $\exists x(f(x) = x)$, *f a un point-fixe.*
5. $\forall x(f(f(f(x))) = x)$.
6. $\forall y \exists x(f(x) = y)$, *$f$ est surjective.*
7. $\forall x \forall y(f(x) = f(y) \Rightarrow x = y)$, *$f$ est injective.*
8. $\neg(\exists x \exists y(f(x) = f(y) \wedge x \neq y))$.

□

Exercice 72 (Formalisation et évaluation) *Nous adoptons les notations suivantes :*

— *Anatoli, Boris, Catarina et Denka sont des constantes du domaine,*
— *$P(x)$ signifie que x a réussi son examen,*
— *$Q(x, y)$ signifie que x a téléphoné à y.*

Donner la signature associée à ces notations et traduire en formules les énoncés suivants :

1. *Quelqu'un a raté l'examen et n'a été appelé par personne.*
2. *Tous ceux qui ont réussi à l'examen ont été appelés.*
3. *Personne n'a appelé tous ceux qui ont réussi à l'examen.*
4. *Tous ceux qui ont appelé quelqu'un, ont appelé quelqu'un qui a réussi l'examen.*

Soit l'interprétation ayant pour domaine $D = \{0, 1, 2, 3\}$. Anatoli, Boris, Catarina et Denka valent respectivement 0, 1, 2 et 3 dans l'interprétation. Anatoli et Boris sont des garçons et Catarina et Denka sont des filles. Dans cette interprétation seuls Boris et Catarina ont réussi l'examen, les garçons ont appelé les filles, Denka a appelé Boris, Catarina a appelé Denka et ce sont les seuls appels.

Nous demandons de définir formellement l'interprétation donnée ci-dessus, puis de donner la valeur des énoncés précédents dans cette interprétation.

Indication *: pour faciliter le calcul de la valeur, nous suggérons de dessiner l'interprétation en entourant les prénoms des personnes qui ont réussi leurs examens, et en mettant une flèche de x vers y si x a téléphoné à y.* □

Exercice 73 (Expansion et contre-modèle) *Trouver, par la méthode des expansions, des contre-modèles des formules suivantes :*

1. $\exists x P(x) \Rightarrow \exists x(P(x) \wedge Q(x))$.
2. $\forall x(P(x) \Rightarrow Q(x)) \Rightarrow \exists x Q(x)$.
3. $\forall x(P(x) \Rightarrow Q(x)) \Rightarrow (\exists x P(x) \Rightarrow \forall x Q(x))$.
4. $(\exists x F(x) \Rightarrow \exists x G(x)) \Rightarrow \forall x(F(x) \Rightarrow G(x))$.
5. $\forall x \exists y R(x, y) \Rightarrow \exists x R(x, x)$.
6. $\forall x \forall y(R(x, y) \Rightarrow R(y, x)) \Rightarrow \forall x R(x, x)$.

Indication *: il suffit de construire des 1 ou 2 expansions.* □

Exercice 74 (Raisonnement incorrect) *Considérons les hypothèses suivantes :*

1. $\exists x P(x)$.
2. $\exists x Q(x)$.
3. $\forall x(P(x) \wedge Q(x) \Rightarrow R(x))$.

Montrer, en utilisant la méthode des expansions, qu'il est incorrect de déduire à partir de ces trois hypothèses la conclusion suivante : $\exists x R(x)$. □

Exercice 75 (Contre-modèles avec relation) *Construire des contre-modèles des formules suivantes, où F est une relation :*

1. $\forall x \exists y(x = y) \Rightarrow \exists y \forall x(y = x)$.
2. $F(a) \wedge (a \neq b) \Rightarrow \neg F(b)$.
3. $\exists x \exists y(F(x) \wedge F(y) \wedge x \neq y) \Rightarrow \forall x F(x)$.
4. $\forall x \forall y(F(x,y) \Rightarrow x = y) \Rightarrow \exists x F(x,x)$.

□

Exercice 76 (Contre-modèles avec fonction) *Construire, en utilisant la méthode des expansions, des contre-modèles des formules suivantes, où f est une fonction et P une relation :*

1. $\forall y \exists x(f(x) = y)$.
2. $\forall x \forall y(f(x) = f(y) \Rightarrow x = y)$.
3. $\exists x \forall y(f(y) = x)$.
4. $\forall x(P(x) \Rightarrow P(f(x)))$.

□

Exercice 77 (Équivalences) *Prouver que :*

1. $\neg \forall x \exists y P(x,y) \equiv \exists y \forall x \neg P(y,x)$.
2. $\exists x(P(x) \Rightarrow Q(x)) \equiv \forall x P(x) \Rightarrow \exists x Q(x)$.
3. *La phrase « aucun malade n'aime les charlatans » a été traduite en logique du premier ordre par deux étudiants par les 2 formules suivantes :*
— $\forall x \forall y((M(x) \wedge A(x,y)) \Rightarrow \neg C(y))$.
— $\neg(\exists x(M(x) \wedge (\exists y(A(x,y) \wedge C(y)))))$.
Montrer que ces étudiants disent la même chose, c'est-à-dire les deux formules associées aux traductions sont équivalentes.

□

Exercice 78 (, Preuve,*) *Prouver les deux équivalences suivantes du lemme 4.4.1 page 86 :*
— $\neg \exists x A \equiv \forall x \neg A$.
— $\exists x A \equiv \neg \forall x \neg A$.

Conseil *: utiliser les propriétés 1 page 86 et 2 page 87 du lemme 4.4.1 page 86.* □

Exercice 79 (Preuve) *Nous savons que $(\forall x(A \wedge B)) \equiv (A \wedge (\forall x B))$ à la condition que x ne soit pas une variable libre de A comme il est dit au paragraphe 4.4.2 page 87. Montrer que cette condition est nécessaire en donnant une assignation qui donne des valeurs différentes aux deux formules $\forall x(P(x) \wedge Q(x))$ et $P(x) \wedge (\forall x Q(x))$.*

□

Chapitre 5

Base de la démonstration automatique

Sommaire

IL est important de préciser que la logique du premier ordre est *indécidable*, cela signifie qu'il n'existe pas d'algorithme pour *déterminer* si une formule est valide ou non valide. Ce résultat fut établi par Alonzo Church et Alan Türing en 1936 et 1937 [4, 24]. Ils ont aussi montré que cette logique est *semi-décidable*, c'est-à-dire que nous pouvons écrire un programme, qui prend en entrée une formule et qui a le comportement suivant :

1. S'il termine alors il *décide correctement* si la formule est valide ou non. Lorsque la formule est valide, la décision est généralement accompagnée d'une preuve.
2. Si la formule est valide, alors il termine. Cependant, l'exécution peut être longue !

Notons que *si la formule n'est pas valide, la terminaison de ce programme n'est pas garantie*. En effet si le programme terminait pour toute formule, alors, d'après le premier point, ce serait un algorithme pour déterminer si une formule est valide ou non. Or, ainsi qu'il est dit ci-dessus, Church et Türing ont prouvé qu'un tel algorithme n'existait pas.

Plan : Dans ce chapitre nous commençons par présenter la méthode, introduite par Jacques Herbrand, permettant d'énumérer un ensemble fini de formules afin de trouver un modèle à une formule. Ensuite nous présentons la transformation due à Thoralf Albert Skolem qui transforme (skolémise) une formule en une « forme de Skolem » où l'existence d'un modèle est préservée. En transformant ensuite cette forme de Skolem en une « forme clausale » équivalente, nous pouvons appliquer la résolution. Nous reprenons alors le concept de résolution introduit dans le chapitre 2 page 37 et nous l'étudions en logique du premier ordre. En 1929, Kurt Gödel (1906 - 1978) prouva que la logique du premier ordre est complète et cohérente. Nous présentons ce résultat pour la résolution, c'est-à-dire qu'il existe une preuve par résolution en logique du premier ordre d'une formule F à partir d'un ensemble de formules Γ si et seulement si F est conséquence de Γ.

5.1 Méthode de Herbrand

Jacques Herbrand (1908-1931) est un mathématicien et logicien français. Il a établi en 1930 un lien entre calcul des prédicats et calcul des propositions.

5.1.1 Domaine et base de Herbrand

Nous introduisons d'abord la notion de fermeture universelle qui permet de transformer toute formule en une formule fermée où la propriété de satisfiabilité est maintenue.

Définition 5.1.1 (Fermeture universelle) *Soit C une formule ayant pour variables libres $x_1,\ldots,x_n$. La* fermeture universelle *de C, notée $\forall(C)$, est la formule $\forall x_1 \ldots \forall x_n C$. Cette notion est définie à l'ordre près des variables libres de C. Soit Γ un ensemble de formules, $\forall(\Gamma) = \{\forall(A) \mid A \in \Gamma\}$.*

Exemple 5.1.2 *La fermeture universelle de $P(x) \wedge R(x,y)$ est*

Nous généralisons la définition 1.3.1 page 19 aux termes de la logique du premier ordre.

Définition 5.1.3 (Substitution) *Une substitution est une application des variables dans les termes. Soient A une formule et σ une substitution. $A\sigma$ est la formule obtenue en remplaçant toute occurrence libre d'une variable par son image dans l'application. La formule $A\sigma$ est une* instance *de A.*

Dans la suite de ce chapitre, nous ne considérons que des formules qui ne contiennent ni le symbole égal, ni les constantes propositionnelles $\top$ et $\bot$, *car leur sens est fixé dans toute interprétation* : $\top$ vaut 1, $\bot$ vaut 0, et le symbole $=$ est l'identité sur le domaine de l'interprétation (voir Remarque 4.3.18 page 79). De plus nous considérons que toute signature comporte au moins une constante. Ainsi la signature d'un ensemble de formules ne comportant pas de constante, sera arbitrairement complétée par la constante a.

Définition 5.1.4 (Domaine et base de Herbrand)

1. *Le* domaine de Herbrand *pour Σ est l'ensemble des termes fermés (*i.e.*, sans variable) de cette signature, dénoté par D_Σ.*
2. *La* base de Herbrand *pour Σ est l'ensemble des formules atomiques fermées de cette signature, nous la notons B_Σ.*

Notons que le domaine de Herbrand d'une signature sans constante est vide. Puisque toutes les signatures considérées dans ce chapitre doivent comporter au moins une constante, ce domaine n'est pas vide.

Exemple 5.1.5 *Calculs de base de Herbrand :*

1. *Soit Σ la signature comportant uniquement les 2 constantes a,b et les 2 symboles de relations unaires P,Q. Nous avons $D_\Sigma = \{a,b\}$ et $B_\Sigma =$*
2. *Soit Σ la signature comportant uniquement la constante a, le symbole de fonction unaire f et le symbole de relation unaire P. Nous avons $D_\Sigma = \{f^n(a) \mid n \in \mathbb{N}\}$ et $B_\Sigma =$*

5.1.2 Interprétation de Herbrand

À partir de la base et du domaine de Herbrand d'une formule, nous sommes capables de construire une interprétation dite de « Herbrand ».

Définition 5.1.6 (Interprétation de Herbrand) *Soient Σ une signature et $E \subseteq B_\Sigma$. L'*interprétation de Herbrand *$H_{\Sigma,E}$ a pour domaine D_Σ et donne aux symboles le sens suivant :*

1. *Si le symbole s est une constante de la signature, il vaut lui-même dans cette interprétation.*
2. *Si s est un symbole de fonction à* $n \geq 1$ *arguments de la signature et si* $t_1, \ldots, t_n \in D_\Sigma$ *alors* $s^{fn}_{H_{\Sigma,E}}(t_1, \ldots, t_n) = s(t_1, \ldots, t_n)$.
3. *Si le symbole s est une variable propositionnelle, il vaut* 1*, autrement dit il est vrai, si et seulement si* $s \in E$.
4. *Si s est un symbole de relation de la signature à* $n \geq 1$ *arguments et si* $t_1, \ldots, t_n \in D_\Sigma$ *alors* $s^{rn}_{H_{\Sigma,E}} = \{(t_1, \ldots, t_n) \mid t_1, \ldots, t_n \in D_\Sigma \wedge s(t_1, \ldots, t_n) \in E\}$.

Le résultat suivant montre qu'une interprétation de Herbrand peut être identifiée à l'ensemble des formules atomiques fermées dont elle est modèle.

Propriété 5.1.7 (Propriété d'une interprétation de Herbrand) *Soient* Σ *une signature et* $E \subseteq B_\Sigma$*. Dans l'interprétation de Herbrand* $H_{\Sigma,E}$ *:*

1. *La valeur d'un terme sans variable est lui-même.*
2. *L'interprétation est modèle d'une formule atomique fermée si et seulement si elle est élément de E.*

La preuve est une conséquence directe de la définition d'une interprétation de Herbrand.
Notons ici, avec un exemple, pourquoi on a supposé que les formules ne contiennent pas les symboles de relation $\top, \bot, =$, dont le sens est fixé dans toutes les interprétations. Supposons au contraire que $\top$ soit élément de la base mais non élément de E. D'après le point 2, l'interprétation $H_{\Sigma,E}$ donnerait $\top$ la valeur 0, alors que $\top$ vaut 1 dans toute interprétation.

Exemple 5.1.8 *Soit* Σ *la signature comportant les* 2 *constantes* a, b *et les* 2 *symboles de relations unaires* P, Q*. L'ensemble* $E = \{P(b), Q(a)\}$ *définit l'interprétation de Herbrand* $H_{\Sigma,E}$ *de domaine* $D_\Sigma = \{a, b\}$

Théorème 5.1.16 (Fermeture universelle et modèle de Herbrand) *Soit* Γ *un ensemble de formules sans quantificateur sur la signature* Σ*. La fermeture universelle* $\forall(\Gamma)$ *a un modèle si et seulement si* $\forall(\Gamma)$ *a un modèle qui est une interprétation de Herbrand de* Σ.

Ce théorème est appliqué dans la preuve du théorème 5.2.17 page 100.

5.1.3 Théorème de Herbrand

Le théorème de Herbrand établit un lien entre la logique du premier ordre et la logique propositionnelle.

Théorème 5.1.17 (Théorème de Herbrand) *Soit* Γ *un ensemble de formules sans quantificateur de signature* Σ*.* $\forall(\Gamma)$ *a un modèle si et seulement si tout ensemble fini d'instances fermées sur la signature* Σ *des formules de* Γ *a un modèle propositionnel application de la base de Herbrand* B_Σ *dans l'ensemble* $\{0, 1\}$.

Corollaire 5.1.18 *Soit* Γ *un ensemble de formules sans quantificateur de signature* Σ*. La fermeture universelle* $\forall(\Gamma)$ *est insatisfaisable si et seulement s'il existe un ensemble fini insatisfaisable d'instances fermées sur la signature* Σ *des formules de* Γ.

Preuve : Le corollaire est obtenu en remplaçant chaque côté de l'équivalence du théorème de Herbrand par sa négation.
□

Dans le cas où Γ est un ensemble *fini* de formules, ce corollaire fonde une procédure de *semi-décision* pour savoir si $\forall(\Gamma)$ est ou n'est pas insatisfaisable : nous énumérons l'ensemble des instances fermées des formules de Γ sur la signature Σ et nous arrêtons cette énumération dès que nous obtenons un ensemble insatisfaisable, ou dès que cette énumération est terminée sans contradiction ou dès que nous sommes « fatigués » :

1. Dans le premier cas, la procédure répond (en appliquant le corollaire) que $\forall(\Gamma)$ est insatisfaisable.
2. Le deuxième cas ne peut se produire que si le domaine de Herbrand ne comprend que des constantes, la procédure répond (en appliquant le corollaire) que $\forall(\Gamma)$ est satisfaisable et en donne un modèle.
3. Dans le troisième cas, nous ne pouvons évidemment pas conclure : le corollaire nous dit que si $\forall(\Gamma)$ est insatisfaisable, et si nous avions été plus courageux, nous aurions obtenu une contradiction.

Exemple 5.1.19

1. *Soit* $\Gamma = \{P(x), Q(x), \neg P(a) \vee \neg Q(b)\}$. *La signature* Σ *comporte 2 constantes* a, b *et 2 symboles de relation unaire* P, Q.

2. *Soit* $\Gamma = \{P(x) \vee Q(x), \neg P(a), \neg Q(b)\}$. *Nous avons la même signature que précédemment.*

3. *Soit* $\Gamma = \{P(x), \neg P(f(x))\}$. *La signature* Σ *comporte une constante* a*, un symbole de fonction unaire* f *et un symbole de relation unaire* P. *La constante* a *est ajoutée à la signature de* Γ *pour que le domaine de Herbrand ne soit pas vide.*

4. *Soit* $\Gamma = \{P(x) \vee \neg P(f(x)), \neg P(a), P(f(f(a)))\}$. *Nous avons la même signature que précédemment.*

5. *Soit* $\Gamma = \{R(x, s(x)), R(x, y) \wedge R(y, z) \Rightarrow R(x, z), \neg R(x, x)\}$. *La signature* Σ *comporte une constante* a*, un symbole de fonction unaire* s *et un symbole de relation binaire* R.

5.2 Skolémisation

Le théorème de Herbrand est applicable à la fermeture universelle d'un ensemble de formules sans quantificateur. Nous étudions une transformation, la *skolémisation*, qui change un ensemble de formules fermées en la fermeture universelle d'un ensemble de formules sans quantificateur et *qui préserve l'existence d'un modèle.* Cette transformation est due à Thoralf Albert Skolem (1887 - 1963), mathématicien et logicien norvégien. Elle est nécessaire pour pouvoir appliquer la résolution sur un ensemble de formules de la logique du premier ordre. Nous commençons par donner l'intuition de la skolémisation avant de regarder le résultat de cette transformation sur des exemples simples afin d'observer les propriétés de la skolémisation. La skolémisation sert à éliminer les quantificateurs existentiels et change une formule fermée A en une formule B telle que :

— B a pour conséquence A.

— tout modèle de A donne un modèle de B.

Par suite A a un modèle si et seulement si B a un modèle : la skolémisation préserve l'existence d'un modèle, nous disons aussi qu'elle préserve la satisfaisabilité. Regardons ce qu'il en est sur deux exemples simples.

Exemple 5.2.1 *La formule $\exists x P(x)$ est skolémisée en $P(a)$. Nous observons les relations entre ces deux formules :*

1. *$P(a)$ a pour conséquence $\exists x P(x)$.*
2. *$\exists x P(x)$ n'a pas pour conséquence $P(a)$ mais un modèle de $\exists x P(x)$ « donne » un modèle de $P(a)$. En effet soit I un modèle de $\exists x P(x)$. Donc il existe $d \in P_I$. Soit J l'interprétation telle que $P_J = P_I$ et $a_J = d$. J est modèle de $P(a)$.*

Exemple 5.2.2 *La formule $\forall x \exists y Q(x,y)$ est skolémisée en $\forall x Q(x, f(x))$. Nous observons les relations entre ces deux formules :*

1. *$\forall x Q(x, f(x))$ a pour conséquence $\forall x \exists y Q(x,y)$.*
2. *$\forall x \exists y Q(x,y)$ n'a pas pour conséquence $\forall x Q(x, f(x))$ mais un modèle de $\forall x \exists y Q(x,y)$ « donne » un modèle de $\forall x Q(x, f(x))$. En effet soient I un modèle de $\forall x \exists y Q(x,y)$ et D le domaine de I. Pour tout $d \in D$, l'ensemble $\{e \in D \mid (d,e) \in Q_I\}$ n'est pas vide, donc il existe $g : D \to D$ une fonction [1] telle que pour tout $d \in D$, $g(d) \in \{e \in D \mid (d,e) \in Q_I\}$. Soit J l'interprétation telle que $Q_J = Q_I$ et $f_J = g$: J est modèle de $\forall x Q(x, f(x))$.*

5.2.1 Algorithme de skolémisation d'une formule

Nous définissons d'abord les éléments nécessaires pour présenter la forme de Skolem, ensuite nous donnons un algorithme de skolémisation.

Définition 5.2.3 (Formule propre) *Une formule fermée est dite* propre*, s'il elle ne comporte pas de variable liée par deux quantificateurs distincts.*

Exemple 5.2.4 *Les formules $\forall x P(x) \vee \forall x Q(x)$ et $\forall x (P(x) \Rightarrow \exists x Q(x) \wedge \exists y R(x,y))$ ne sont pas propres.*
Les formules $\forall x P(x) \vee \forall y Q(y)$ et $\forall x (P(x) \Rightarrow \exists y R(x,y))$ sont propres.

Nous étendons la notion de forme normale de la logique des propositions à la logique du premier ordre en disant qu'une formule est en forme normale si elle est sans équivalence, ni implication, et dont les négations portent uniquement sur les formules atomiques.

Définition 5.2.5 (Forme de Skolem) *Soient A une formule fermée et B la formule en forme normale sans quantificateur obtenue par la transformation ci-après : B est la* forme de Skolem *de A.*

Il y a plusieurs façons de skolémiser une formule fermée. Nous proposons un compromis entre l'efficacité et la simplicité de la transformation. Notre algorithme est consistué de la succession des quatre étapes suivantes :

1. **Transformation en formule normale** : Transformation de la formule fermée en une autre formule fermée en forme normale équivalente.
2. **Transformation en formule propre** : Transformation de la formule fermée en forme normale en une formule fermée, en forme normale, propre et équivalente.
3. **Élimination des quantificateurs existentiels** : Cette transformation préserve *seulement l'existence de modèle*, comme nous l'avons vu sur les exemples.
4. **Transformation en forme de Skolem** : Transformation de la formule fermée, en forme normale, propre et sans quantificateur existentiel en une formule en forme normale sans quantificateur.

Nous détaillons chacune de ces quatre étapes.

1. En affirmant l'existence de g, nous utilisons implicitement l'axiome du choix.

5.2.1.1 Transformation en formule normale

Comme dans le cas propositionnel, nous enlevons les équivalences et les implications et déplaçons les négations vers les formules atomiques au moyen des équivalences suivantes utilisées de gauche à droite :

— $A \Leftrightarrow B \equiv (A \Rightarrow B) \wedge (B \Rightarrow A)$.
— $A \Rightarrow B \equiv \neg A \vee B$.
— $\neg\neg A \equiv A$.
— $\neg(A \wedge B) \equiv \neg A \vee \neg B$.
— $\neg(A \vee B) \equiv \neg A \wedge \neg B$.
— $\neg\forall x A \equiv \exists x \neg A$.
— $\neg\exists x A \equiv \forall x \neg A$.

Remarque 5.2.6 *En combinant déplacement de la négation et élimination de l'implication, c'est-à-dire en remplaçant $\neg(A \Rightarrow B)$ par $A \wedge \neg B$, nous pouvons effectuer plus rapidement la transformation en formule normale équivalente, qu'en se limitant aux équivalences ci-dessus.*

Exemple 5.2.7 *La formule $\forall y(\forall x P(x,y) \Leftrightarrow Q(y))$ est transformée par élimination de l'équivalence et de l'implication en :*

5.2.1.2 Transformation en formule propre

En utilisant les résultats sur les changements de variables présentés dans la section 4.4.3 page 87, nous changeons le nom des variables liées à deux quantificateurs, en choisissant par exemple de nouvelles variables à chaque changement de nom.

Exemple 5.2.8 *La formule $\forall x P(x) \vee \forall x Q(x)$ est changée en*

La formule $\forall x(P(x) \Rightarrow \exists x Q(x) \wedge \exists y R(x,y))$ est changée en

5.2.1.3 Élimination des quantificateurs existentiels

Soit A une formule fermée normale et propre ayant une occurrence de la sous-formule $\exists y B$. Soient $x_1,\ldots,x_n$ l'ensemble des variables libres de $\exists y B$, où $n \geq 0$. Soit f un symbole *ne figurant pas dans A*. Nous remplaçons cette occurrence de $\exists y B$ par $B{<}y := f(x_1,\ldots,x_n){>}$ [2].

Théorème 5.2.9 *Soit A' la formule obtenue en appliquant l'élimination d'un quantificateur existentiel sur la formule A. La formule A' est une formule fermée en forme normale et propre vérifiant :*

1. *A' a pour conséquence A ($A' \models A$).*
2. *Si A a un modèle alors A' a un modèle identique à celui de A sauf pour le sens de f.*

2. Si $n = 0$, f est une constante.

Remarque 5.2.10 *Dans le théorème 5.2.9 page précédente, il faut constater que la formule A' obtenue à partir de la formule A par élimination d'un quantificateur reste fermée, en forme normale et propre. Donc, en « appliquant » plusieurs fois le théorème, ce qui implique de choisir un* nouveau *symbole à chaque quantificateur éliminé, nous transformons une formule A fermée, en forme normale et propre en une formule B fermée, en forme normale, propre et* sans quantificateur existentiel *telle que :*

— *la formule A est conséquence de la formule B,*
— *si A a un modèle, alors B a un modèle identique sauf pour le sens des nouveaux symboles.*

Exemple 5.2.11 *En éliminant les quantificateurs existentiels de la formule $\exists x \forall y P(x,y) \wedge \exists z \forall u \neg P(z,u)$ nous obtenons*

Donc, il faut impérativement utiliser un nouveau symbole lors de chaque élimination d'un quantificateur existentiel.

Exemple 5.2.12 *En éliminant les quantificateurs existentiels de la formule $\exists x \forall y \exists z P(x,y,z)$ nous obtenons deux solutions possibles :*

— $\forall y P(a,y,f(a,y))$
— $\forall y P(a,y,f(y))$

Ceci dépend de l'ordre dans lequel nous appliquons l'elimination des quantificateurs existentiels. Or le théorème 5.2.9 montre que une des ces transformations a un modèle si et seulement si l'autre a également un modèle.

5.2.1.4 Transformation en fermeture universelle

Nous pouvons enfin enlever les quantificateurs universels afin d'obtenir une forme de Skolem.

Théorème 5.2.13 *Soit A une formule fermée, en forme normale, propre et sans quantificateur existentiel. Soit B la formule obtenue en enlevant de A tous les quantificateurs universels. La formule A est équivalente à la fermeture universelle de B.*

Preuve : Avec les conditions posées sur A, la transformation de A en $\forall(B)$ revient à effectuer tous les remplacements possibles des sous-formules de la forme suivante :

— $(\forall x C) \wedge D$ par $\forall x(C \wedge D)$, où x non libre dans D,
— $(\forall x C) \vee D$ par $\forall x(C \vee D)$, où x non libre dans D,
— $D \wedge (\forall x C)$ par $\forall x(D \wedge C)$, où x non libre dans D,
— $D \vee (\forall x C)$ par $\forall x(D \vee C)$, où x non libre dans D.

Puisque chacun de ces remplacements change une formule en une autre équivalente, les formules A et $\forall(B)$ sont équivalentes. □

5.2.2 Propriétés de la forme de Skolem

Propriété 5.2.14 *Soit A une formule fermée et B la forme de Skolem de A.*

— *La formule $\forall(B)$ a pour conséquence la formule A,*
— *si A a un modèle alors $\forall(B)$ a un modèle.*

Donc A a un modèle si et seulement si $\forall(B)$ a un modèle.

Preuve : Soit C la formule fermée en forme normale et propre, obtenue au terme des deux premières étapes de la skolémisation de A. Soit D le résultat de l'élimination des quantificateurs existentiels appliquée à C. D'après la remarque 5.2.10 nous avons :

— la formule D a pour conséquence la formule C,
— si C a un modèle alors D a un modèle.

Puisque les deux premières étapes changent des formules en des formules équivalentes, A et C sont équivalentes. D'après le théorème 5.2.13, D est équivalent à $\forall(B)$. Donc nous pouvons remplacer ci-dessus D par $\forall(B)$ et C par A, ce qui prouve le théorème. □

Exemple 5.2.15 *Soit $A = \forall x(P(x) \Rightarrow Q(x)) \Rightarrow (\forall x P(x) \Rightarrow \forall x Q(x))$. Nous skolémisons $\neg A$.*

1. *$\neg A$ est transformée en la formule en forme normale :*

2. *La formule en forme normale est transformée en la formule propre :*

3. *Le quantificateur existentiel est « remplacé » par une constante :*

4. *Les quantificateurs universels sont enlevés :*

   Cette formule est la forme de Skolem de $\neg A$ suivant la définition 5.2.5 page 97.

Instancions la forme de Skolem de $\neg A$ en remplaçant x et y par a. Nous obtenons la formule $(\neg P(a) \vee Q(a)) \wedge P(a) \wedge \neg Q(a)$ qui est insatisfaisable. Donc $\forall((\neg P(x) \vee Q(x)) \wedge P(y) \wedge \neg Q(a))$ est insatisfaisable. Puisque, d'après la propriété 5.2.14 page précédente, la skolémisation préserve l'existence d'un modèle, $\neg A$ est insatisfaisable, donc A est valide.

Pour skolémiser un ensemble de formules, nous skolémisons chaque formule de l'ensemble comme il a été indiqué ci-dessus en choisissant impérativement un nouveau symbole pour chaque quantificateur existentiel éliminé durant la troisième étape de la skolémisation. Les raisons pour le choix d'un symbole nouveau apparaissent clairement dans la preuve du théorème 5.2.9 page 98 et dans l'exemple 5.2.11 page précédente.

Corollaire 5.2.16 *En skolémisant un ensemble de clauses fermées Γ, nous obtenons un ensemble Δ de formules sans quantificateurs tel que :*
- *Tout modèle de $\forall(\Delta)$ est modèle de Γ.*
- *Si Γ a un modèle alors $\forall(\Delta)$ en a un modèle qui ne diffère de celui de Γ que par le sens des nouveaux symboles.*

La skolémisation a des conséquences pratiques : elle facilite la preuve d'insatisfiabilité d'une formule. Elle a aussi des conséquences théoriques, que nous examinons.

Théorème 5.2.17 *Soit Γ un ensemble dénombrable de formules fermées. Si Γ a un modèle alors Γ a un modèle dénombrable.*

Preuve : Nous skolémisons Γ en un ensemble Δ de formules sans quantificateurs. Supposons que Γ a un modèle. Alors $\forall(\Delta)$ a un modèle d'après le corollaire 5.2.16. D'après le théorème 5.1.16 page 95, $\forall(\Delta)$ a un modèle de Herbrand sur la signature de Δ. Puisque l'ensemble Γ est dénombrable, sa signature l'est aussi, donc aussi celle de Δ et par suite le domaine de ce modèle de Herbrand est dénombrable. □

Remarque 5.2.18 *Appelons* théorie*, un ensemble de formules. D'après le théorème, toute théorie dénombrable qui a un modèle, a un modèle dénombrable. Autrement dit avec une théorie du premier ordre, nous pouvons parler des propriétés des nombres réels ou des ensembles, mais nous ne pouvons pas obliger notre théorie à n'avoir que des modèles non dénombrables, bien que l'ensemble des réels ou la classe de tous les ensembles ne soient pas dénombrables.*

5.2.3 Forme clausale

Après la skolémisation des formules et avant de pouvoir appliquer une généralisation au premier ordre de la méthode de résolution il faut transformer les formes de Skolem en clauses comme nous l'avons fait en logique propositionnelle. Nous étendons donc les notions essentielles à cette transformation avant de donner les règles permettant cette transformation.

Définition 5.2.19 (Littéral positif, négatif et clause) *Un* littéral positif *est une formule atomique. Un* littéral négatif *est la négation d'une formule atomique.*

Nous remarquons que tout littéral est positif ou négatif, et nous rappelons qu'une clause est une somme de littéraux.

Définition 5.2.20 (Forme clausale d'une formule) *Soit A une formule fermée, la* forme clausale *de A est un ensemble de clauses obtenu en deux étapes :*

1. *Skolémisation de A, autrement dire construire B la forme de Skolem de A.*
2. *Remplacement de B par un ensemble Γ équivalent de clauses obtenu par distributivité de la somme sur le produit.*

Propriété 5.2.21 *La fermeture universelle de la forme clausale d'une formule fermée a un modèle si et seulement la formule a un modèle. Plus précisément :*
- *la fermeture universelle de la forme clausale d'une formule a pour conséquence la formule,*
- *si la formule a un modèle alors la fermeture universelle de sa forme clausale en a un.*

Preuve : Soient A une formule fermée, B sa forme de Skolem et Γ sa forme clausale. D'après les propriétés de la skolémisation :
- $\forall(B)$ a pour conséquence A.
- Si A a un modèle alors $\forall(B)$ a un modèle.

Puisque Γ est obtenu par distributivité, B et Γ sont équivalents donc $\forall(B)$ et $\forall(\Gamma)$ sont aussi équivalents. Par suite dans les deux propriétés ci-dessus, nous pouvons remplacer $\forall(B)$ par $\forall(\Gamma)$. □

Définition 5.2.22 (Forme clausale d'un ensemble de formules) *Soit Γ un ensemble de formules fermées. Nous définissons la* forme clausale *de Γ comme l'union des formes clausales de chacune des formules de Γ, en prenant soin au cours de la skolémisation d'éliminer chaque occurrence d'un quantificateur existentiel à l'aide d'un nouveau symbole.*

Corollaire 5.2.23 *Soient Γ un ensemble de formules fermées et Δ la forme clausale de Γ. Nous avons :*
- *$\forall(\Delta)$ a pour conséquence Γ, et*
- *si Γ a un modèle alors $\forall(\Delta)$ a un modèle.*

Théorème 5.2.24 (Adaptation du théorème de Herbrand aux formes clausales) *Soient Γ un ensemble de formules fermées et Δ la forme clausale de Γ. L'ensemble Γ est insatisfaisable si et seulement s'il existe un sous-ensemble fini insatisfaisable d'instances des clauses de Δ sur la signature de Δ.*

Preuve : D'après l5.2.23, la skolémisation préserve la satisfaisablité, donc : Γ est insatisfaisable si et seulement si $\forall(\Delta)$ est insatisfaisable. D'après le corollaire du théorème de Herbrand 5.1.18 page 95, $\forall(\Delta)$ est insatisfaisable si et seulement s'il existe un sous-ensemble fini insatisfaisable d'instances des clauses de Δ sur la signature de Δ. □

Exemple 5.2.25 *Soit $A = \exists y \forall z(P(z,y) \Leftrightarrow \neg\exists x(P(z,x) \wedge P(x,z)))$. Calculons la forme clausale de A.*

1. *Mettre A sous forme en forme normale :*

2. *Rendre propre le résultat :*

3. *Éliminer les quantificateurs existentiels :*

4. *Supprimer les quantificateurs universels, nous obtenons la forme de Skolem de A :*

5. *Transformer en produit de sommes de littéraux, nous obtenons la forme clausale de A, qui est l'ensemble suivant de clauses :*

D'après la propriété ci-dessus, A n'a pas de modèle si et seulement s'il y a un ensemble fini insatisfaisable d'instances de C_1, C_2, C_3 *sur la signature de ces clauses. Nous recherchons ces instances :*

5.3 Unification

Après la mise en forme clausale d'un ensemble de formules en logique du premier ordre et avant de généraliser la résolution pour ces clauses, nous avons besoin d'un algorithme résolvant l'égalité entre deux termes. Étant donné deux termes, trouver une substitution qui appliquée à ces deux termes les rendent égaux est appelé *problème d'unification* de ces deux termes. Ce problème a intéressé de grands noms de l'informatique comme Alan Robinson [23], Gérard Huet [15, 16], Alberto Martelli et Ugo Montanari [21], ou encore plus récemment Franz Baader et Ayne Snyder [3].

5.3.1 Unificateur

Définition 5.3.1 (Expression, solution) *Appelons* expression*, un terme ou un littéral. Une substitution* σ *(voir définition 5.1.3 page 94) est* solution de l'équation $e_1 = e_2$ *entre deux expressions, si les deux expressions* $e_1\sigma$ *et* $e_2\sigma$ *sont* identiques*. Une substitution est solution d'un ensemble d'équations si elle est solution de chaque équation de l'ensemble.*

Définition 5.3.2 (Unificateur) *Soient* σ *une substitution et E un ensemble d'expressions.* $E\sigma = \{t\sigma \mid t \in E\}$*. La substitution* σ *est un unificateur de E si et seulement si l'ensemble* $E\sigma$ *n'a qu'un élément. Soit* $\{e_i | 1 \leq i \leq n\}$ *un ensemble fini d'expressions. La substitution* σ *est un unificateur de cet ensemble si et seulement si elle est solution du système d'équations* $\{e_i = e_{i+1} | 1 \leq i < n\}$*.*

Définition 5.3.3 (Support d'une substitution) *Le* support *d'une substitution* σ *est l'ensemble des variables x telles que* $x\sigma \neq x$*. Dans la suite, nous nous intéressons seulement aux substitutions à support fini, c'est à dire qui ne changent qu'un nombre fini de variables. Une substitution* σ *à support fini est notée* $< x_1 := t_1, \ldots, x_n := t_n >$ *ou plus simplement* $x_1 := t_1, \ldots, x_n := t_n$ *quand il n'y a pas de risque d'ambiguïté. Les variables* $x_1, \ldots, x_n$ *sont distinctes et la substitution vérifie :*

— *Pour i de 1 à n, $x_i\sigma = t_i$.*
— *Pour toute variable y telle que $y \notin \{x_1, \ldots, x_n\}$, nous avons : $y\sigma = y$.*

Exemple 5.3.4 *L'équation $P(x, f(y)) = P(g(z), z)$ a pour solution :*

Le système d'équations $x = g(z), f(y) = z$ a pour solution :

Définition 5.3.5 (Composition de substitutions) *Soient σ et τ deux substitutions, nous notons $\sigma\tau$ la substitution telle que pour toute variable x, $x\sigma\tau = (x\sigma)\tau$. La substitution $\sigma\tau$ est une instance de σ. Deux substitutions sont équivalentes si chacune d'elles est une instance de l'autre.*

Exemple 5.3.6 *Considérons les substitutions suivantes :*
— $\sigma_1 =< x := g(z), y := z >$.
— $\sigma_2 =< x := g(y), z := y >$.
— $\sigma_3 =< x := g(a), y := a, z := a >$.
Nous avons les relations suivantes entre ces substitutions :

Définition 5.3.7 (Solution la plus générale) *Une solution d'un système d'équations est appelée* la plus générale *(en français nous disons aussi* principale*) si toute autre solution en est une instance. Notons que deux solutions « les plus générales » sont équivalentes.*

Exemple 5.3.8 *L'équation $f(x, g(z)) = f(g(y), x)$ a pour solutions (entre autres) :*

Définition 5.3.9 (Unificateur le plus général) *Soit E un ensemble d'expressions. Nous rappelons qu'une expression est un terme ou un littéral. Un unificateur de E (voir définition 5.3.2 page précédente) est appelé* le plus général *(ou encore principal), si tout autre unificateur en est une instance.*

Remarque 5.3.10 (Unificateur le plus général et solution la plus générale) *Soit $E = \{e_i \mid 1 \leq i \leq n\}$ un ensemble d'expressions. Dans la définition d'un unificateur, nous avons indiqué que σ est un unificateur de E si et seulement si σ est solution du système $S = \{e_i = e_{i+1} \mid 1 \leq i < n\}$.*
Donc l'unificateur le plus général de E est la solution la plus générale de S.

5.3.2 Algorithme d'unification

L'algorithme calculant la solution d'un système d'équations est appelé *algorithme d'unification* car la recherche d'un unificateur d'un ensemble d'expressions se réduit à la recherche d'une solution d'un système d'équations. L'algorithme sépare les équations en équations à résoudre, notées par une égalité et équations résolues, notées par le signe $:=$. Initialement, il n'y a pas d'équations résolues. L'algorithme applique les règles énoncées ci-dessous. Il s'arrête quand il n'y a plus d'équations à résoudre ou quand il a déclaré que le système à résoudre n'a pas de solution. Quand il s'arrête sans avoir déclaré l'absence de solution, la liste des équations résolues est la solution la plus générale du système initial d'équations.

5.3.2.1 Les règles de l'algorithme

Notre algorithme d'unification consiste à appliquer les six règles suivantes :

1. **Supprimer.** Si les deux membres d'une équation sont identiques, l'équation est supprimée [3].
2. **Décomposer.** Si les 2 membres d'une équation sont distincts et commencent par le même symbole alors :
   — Une équation de la forme $\neg A = \neg B$ est remplacée par $A = B$.
   — Une équation de la forme $f(s_1, \ldots, s_n) = f(t_1, \ldots, t_n)$ est remplacée par les équations $s_1 = t_1, \ldots, s_n = t_n$. Pour $n = 0$, cette décomposition supprime l'équation.
3. **Échec de la décomposition.** Si une équation à résoudre est de la forme $f(s_1, \ldots, s_n) = g(t_1, \ldots, t_p)$ avec $f \neq g$ ou $n \neq p$ alors l'algorithme déclare qu'il n'y a pas de solution. En particulier il y a évidemment un échec si nous cherchons à résoudre une équation entre un littéral positif et un littéral négatif ou bien si nous avons une constante et une fonction.
4. **Orienter.** Si une équation est de la forme $t = x$ où t est un terme qui n'est pas une variable et x une variable, alors nous remplaçons l'équation par $x = t$.
5. **Élimination d'une variable** : Si une équation à résoudre est de la forme $x = t$ où x est une variable et t un terme *ne contenant pas* x alors :
   (a) Enlever des équations à résoudre $x = t$.
   (b) Remplacer x par t dans toutes les équations (non résolues *et résolues*) [4].
   (c) Ajouter $x := t$ à la partie résolue.
6. **Échec de l'élimination** : Si une équation à résoudre est de la forme $x = t$ où x est une variable et t un terme distinct de x et *contenant* x alors l'algorithme déclare qu'il n'y a pas de solution.

Exemple 5.3.11 *Nous appliquons cet algorithme afin de résoudre les équations suivantes :*

1. $f(x, g(z)) = f(g(y), x)$.

2. $f(x, x, a) = f(g(y), g(a), y)$.

3. $f(x, x, x) = f(g(y), g(a), y)$.

3. Nous pouvons nous limiter à la suppression des équations de la forme $x = x$ où x est une variable.
4. Un algorithme efficace évite ce remplacement systématique.

Nous montrons d'abord que notre algorithme est correct, c'est-à-dire qu'il calcule bien un unificateur. Ensuite nous démontrons qu'il termine toujours.

5.3.2.2 Correction de l'algorithme

Nous observons facilement les trois faits suivants :

— (1) : les règles de suppression, de décomposition, d'orientation et d'élimination des variables préservent l'ensemble des solutions.

— (2) : initialement et après toute application de règle, l'ensemble des équations résolues forme une substitution $x_1 := t_1, \ldots, x_n := t_n$ où $x_1, \ldots, x_n$ sont des variables distinctes ne figurant pas dans les termes $t_1, \ldots, t_n$. Soit σ cette substitution. Il est clair que $t_i\sigma = t_i$ ainsi σ est solution des équations résolues.

— (3) : Les règles d'échec déclarent correctement l'absence de solution.

De (1) et (2), il résulte que si l'algorithme termine sans échec, la substitution σ trouvée est la solution la plus générale du *système initial*. En effet :

— σ est une solution, puisque, d'après (2), elle est solution du *système terminal* et que, d'après (1), l'ensemble des solutions est préservé par les règles.

— σ est une solution la plus générale, puisque, d'après (1), toute autre solution est une solution du système terminal, donc une instance de la substitution définie par ce système.

De (1) et de (3), il résulte que si l'algorithme déclare un échec, alors le système initial n'a pas de solution. En effet, dans ce cas :

— si le système initial avait une solution, d'après (1), elle serait solution du système terminal,

— d'après (3), le système terminal n'a pas de solution.

5.3.2.3 Terminaison de l'algorithme

— Les règles de suppression et de décomposition font décroître strictement la somme des longueurs des termes qui sont de part et d'autre des équations.

— La règle d'orientation fait décroître strictement la somme des longueurs des termes qui sont à gauche des équations.

— La règle d'élimination fait décroître strictement le nombre de variables des équations non résolues.

Il résulte que l'algorithme d'unification termine.

5.4 Résolution au premier ordre

Nous avons maintenant tous les ingrédients pour présenter la résolution au premier ordre. Bien entendu nous transformons un ensemble de formules en forme clausale. Ensuite nous pouvons appliquer trois règles permettant de « factoriser » des termes, de « copier » une clause et enfin de faire une « résolution binaire ». Ces différentes règles utilisent l'algorithme d'unification présenté précédemment. Par la suite, nous montrons la cohérence et la complétude de ce système formel. La complétude est basée sur le théorème de Herbrand dans la version donnée dans le corollaire 5.2.24 page 101.

5.4.1 Trois règles pour la résolution

Soit Γ un ensemble de clauses, nous supposons que $\forall(\Gamma)$ n'a pas de modèle, nous proposons alors un système formel permettant de déduire $\perp$ de Γ avec les trois règles suivantes :

1. La *factorisation* qui de la prémisse $P(x, f(y)) \vee P(g(z), z) \vee Q(z, x)$ déduit $P(g(f(y)), f(y)) \vee Q(f(y), g(f(y)))$. La clause déduite est obtenue en calculant la solution la plus générale $x := g(f(y)), z := f(y)$ de $P(x, f(y)) = P(g(z), z)$.
2. La règle de *copie* qui permet de renommer les variables d'une clause.
3. La *résolution binaire* (RB) qui des deux prémisses sans variable commune $P(x, a) \vee Q(x)$ et $\neg P(b, y) \vee R(f(y))$ déduit le résolvant $Q(b) \vee R(f(a))$, en calculant la solution plus générale $x := b, y := a$ de $P(x, a) = P(b, y)$.

Pour simplifier la présentation des règles, nous identifierons une clause, qui est une somme de littéraux, avec l'ensemble de ses littéraux. Nous formalisons ces trois règles.

5.4.1.1 Factorisation

Propriété 5.4.1 *Soient A une formule sans quantificateur et B une instance de A. Nous avons :* $\forall(A) \models \forall(B)$.

Définition 5.4.2 *La clause C' est un* facteur *de la clause C si $C' = C$ ou s'il existe un sous-ensemble E de C tel que E a au moins deux éléments, E est unifiable et $C' = C\sigma$ où σ est l'unificateur le plus général de E.*

Exemple 5.4.3 *La clause* $\underline{P(x)} \vee Q(g(x, y)) \vee \underline{P(f(a))}$ *a deux facteurs :*

Nous obtenons immédiatement la propriété suivante.

Propriété 5.4.4 (Cohérence de la règle de factorisation) *Soit C' un facteur de la clause C. Nous avons :* $\forall(C) \models \forall(C')$.

Preuve : Puisque C' est une instance de C, la propriété est une conséquence immédiate de la propriété 5.4.1. □

5.4.1.2 Copie d'une clause

Définition 5.4.5 *Soient C une clause et σ une substitution, qui ne change que les variables de C et dont la* restriction *aux variables de C est une* bijection *entre ces variables et celles de la clause $C\sigma$. La clause $C\sigma$ est une copie de la clause C. La substitution σ est aussi appelée un* renommage *de C.*

Définition 5.4.6 *Soient C une clause et σ un renommage de C. Soit f la restriction de σ aux variables de C et f^{-1} l'application réciproque de f. Soit σ_C^{-1} la substitution ainsi définie pour toute variable x :*

— *Si x est une variable de $C\sigma$ alors $x\sigma_C^{-1} = xf^{-1}$ (par soucis de clarté, nous avons préféré utiliser la notation postfixée xf^{-1} plutôt que la notation préfixée $f^{-1}(x)$, plus usuelle).*

— *Sinon $x\sigma_C^{-1} = x$.*

*Cette substitution est appelée l'*inverse du renommage *σ de C.*

Exemple 5.4.7 *Soit $\sigma = <x := u, y := v>$. σ est un renommage du littéral $P(x, y)$. Le littéral $P(u, v)$, où $P(u, v) = P(x, y)\sigma$, est une copie de $P(x, y)$. Soit $\tau = <u := x, v := y>$. τ est l'inverse du renommage σ de $P(x, y)$. Notons que $P(u, v)\tau = P(x, y)$: le littéral $P(x, y)$ est une copie de $P(u, v)$ par le renommage τ.*

Propriété 5.4.8 *Soient C une clause et σ un renommage de C.*

1. *σ_C^{-1} est un renommage de $C\sigma$.*
2. *Pour toute expression ou clause E, dont les variables sont celles de C, $E\sigma\sigma_C^{-1} = E$.*

Donc $C\sigma\sigma_C^{-1} = C$ et par suite C est une copie de $C\sigma$.

Preuve : Soit f la restriction de σ aux variables de C. Par définition du renommage, f est une bijection entre les variables de C et celles de $C\sigma$.

1. Par définition de σ_C^{-1}, cette substitution ne change que les variables de $C\sigma$ et sa restriction aux variables de $C\sigma$ est la bijection f^{-1}. Donc, σ_C^{-1} est un renommage de $C\sigma$.

2. Soit x une variable de C. Par définition de f, $x\sigma\sigma_C^{-1} = xff^{-1} = x$. Donc, par une récurrence omise et sans intérêt sur les termes, littéraux et clauses, pour toute expression ou clause E, dont les variables sont celles de C, nous avons $E\sigma\sigma_C^{-1} = E$.

□

Propriété 5.4.9 *Soient deux clauses copies l'une de l'autre, leurs fermetures universelles sont équivalentes.*

Preuve : Soit C' une copie de C. Par définition, C' est une instance de C et par la propriété précédente, C est une copie de C', donc une instance de C'. Donc par la propriété 5.4.1 page ci-contre, la fermeture universelle de C est conséquence de celle de C' et inversement. Par suite, ces deux fermetures universelles sont équivalentes. □

5.4.1.3 Résolution binaire

Définition 5.4.10 *Soient C et D deux clauses* n'ayant pas de variable commune. *La clause E est un* résolvant binaire *de C et D s'il y a un littéral $L \in C$ et un littéral $M \in D$ tels que L et M^c (le littéral opposé à M) sont unifiables et si $E = ((C - \{L\}) \cup (D - \{M\}))\sigma$ où σ est la solution la plus générale de l'équation $L = M^c$.*

Exemple 5.4.11 *Soient $C = P(x,y) \vee P(y,k(z))$ et $D = \neg P(a, f(a,y_1))$.*

Propriété 5.4.12 (Cohérence de la règle de résolution binaire) *Soit E un résolvant binaire des clauses C et D, nous avons : $\forall(C), \forall(D) \models \forall(E)$.*

Preuve : Soit I une interprétation. Puisque les trois formules $\forall(C), \forall(D), \forall(E)$ sont sans variable libre, il suffit de montrer que si I est modèle de $\forall(C)$ et de $\forall(D)$ alors I est modèle de $\forall(E)$. Supposons que I est modèle de $\forall(C), \forall(D)$ et montrons que c'est un modèle de $\forall(E)$. Soit e un état *quelconque* de cette interprétation. (I,e) est modèle de $C\sigma$ et $D\sigma$, pour toute substitution σ. Par définition du résolvant binaire, il y a un littéral $L \in C$ et un littéral $M \in D$ tels que L et M^c (le littéral opposé à M) sont unifiables. Ainsi, $L\sigma$ et $M\sigma$ sont deux littéraux opposés, donc (I,e) est modèle soit de l'un, soit de l'autre de ces deux littéraux.

1. Supposons (I,e) modèle de $L\sigma$. Puisque (I,e) est contre-modèle de $M\sigma$ et modèle de $D\sigma$, c'est un modèle de $(D - \{M\})\sigma$ donc de E car $(D - \{M\})\sigma \subset E$.
2. Supposons (I,e) contre-modèle de $L\sigma$. Puisque c'est un modèle de $C\sigma$, c'est un modèle de $(C - \{L\})\sigma$ donc de E car $(C - \{L\})\sigma \subset E$.

Donc (I,e) est modèle de E. Puisque l'état e est quelconque, I est modèle de $\forall(E)$. □

5.4.1.4 Preuve par factorisation, copie et résolution binaire

Définition 5.4.13 *Soient Γ un ensemble de clauses et C une clause. Une* preuve *de C à partir de Γ est une suite de clauses se terminant par C, toute clause de la preuve étant un élément de Γ, un facteur d'une clause la précédant dans la preuve, une copie d'une clause la précédant dans la preuve ou un résolvant binaire de deux clauses la précédant dans la preuve.*

C est déduite de Γ au premier ordre par les 3 règles de factorisation, copie et résolution binaire est dénoté par $\Gamma \vdash_{1fcb} C$. Cela signifie qu'il y a une preuve de C à partir de Γ. Quand il n'y a pas d'ambiguïté sur le système formel utilisé, nous remplaçons $\vdash_{1fcb}$ par $\vdash$.

5.4.2 Cohérence de la résolution

Propriété 5.4.14 (Cohérence) *Soient Γ un ensemble de clauses et C une clause. Si $\Gamma \vdash_{1fcb} C$ alors $\forall(\Gamma) \models \forall(C)$.*

Preuve : Cette propriété est une conséquence immédiate de la cohérence de la factorisation, de la copie et de la résolution binaire. Cette preuve est demandée dans l'exercice 97 page 114. □

Exemple 5.4.15 *Soient les deux clauses :*

1. $C_1 = P(x,y) \vee P(y,x)$.
2. $C_2 = \neg P(u,z) \vee \neg P(z,u)$.

Montrons par résolution que $\forall(C_1,C_2)$ *n'a pas de modèle.*

Cet exemple montre, a contrario, que la résolution binaire seule est incomplète, sans la factorisation, nous ne pouvons pas déduire la clause vide.

Exemple 5.4.16 *Soient les trois clauses :*

1. $C_1 = \neg P(z,a) \vee \neg P(z,x) \vee \neg P(x,z)$.
2. $C_2 = P(z,f(z)) \vee P(z,a)$.
3. $C_3 = P(f(z),z) \vee P(z,a)$.

Nous donnons une preuve par résolution que $\forall(C_1,C_2,C_3)$ *n'a pas de modèle. Dans cette preuve RB 1(3), 3(1) signifie par résolution binaire sur le 3º littéral de la clause 1 et le 1º littéral de la clause 3.*

5.4.3 Complétude de la résolution

Nous définissons une *nouvelle* règle, la résolution au 1º ordre, qui est une combinaison des trois règles de factorisation, copie et résolution binaire. Ceci nous permettra de prouver la complétude de la résolution.

Définition 5.4.17 *La clause E est un résolvant au 1º ordre des clauses C et D si E est un résolvant binaire de C' et D' où C' est un facteur de C et D' est une copie sans variable commune avec C' d'un facteur de D. La règle qui de C et D déduit E est appelée la résolution de 1º ordre.*

Exemple 5.4.18 *Soient $C = \neg P(z,a) \vee \neg P(z,x) \vee \neg P(x,z)$ et $D = P(z, f(z)) \vee P(z,a)$.*

$C' = \neg P(a,a)$ est un facteur de C. La clause $P(a, f(a))$ est un résolvant binaire de C' et de D (qui est facteur de lui-même) donc

Soient Γ un ensemble de clauses et C une clause, nous avons défini jusqu'à présent trois notions de preuve par résolution que nous distinguons en notant par :

1. $\Gamma \vdash_p C$ le fait qu'il y existe une preuve de C à partir de Γ obtenue par résolution propositionnelle, autrement dit sans substitution.
2. $\Gamma \vdash_{1fcb} C$ le fait qu'il y existe une preuve de C à partir de Γ par factorisation, copie et résolution binaire.
3. $\Gamma \vdash_{1r} C$ le fait qu'il y existe une preuve de C à partir de Γ obtenue par résolution de 1º ordre.

Puisque la résolution du premier ordre est une combinaison des règles factorisation, copie et résolution binaire, nous avons immédiatement $\Gamma \vdash_{1r} C$ implique $\Gamma \vdash_{1fcb} C$.

Théorème 5.4.19 (Théorème du relèvement) *Soient C et D deux clauses. Soient C' une instance de C et D' une instance de D. Soit E' un résolvant* propositionnel *de C' et D', il existe E un résolvant au* premier ordre *de C et D qui a pour instance E'.*

Exemple 5.4.20 *Soient $C = P(x) \vee P(y) \vee R(y)$ et $D = \neg Q(x) \vee P(x) \vee \neg R(x) \vee P(y)$.*

— *Les clauses $C' = P(a) \vee R(a)$ et $D' = \neg Q(a) \vee P(a) \vee \neg R(a)$ sont des instances respectivement de C et D.*
— *La clause $E' = P(a) \vee \neg Q(a)$ est un résolvant propositionnel de C' et D'.*
— *La clause $E = P(x) \vee \neg Q(x)$ est un résolvant au 1º ordre de C et D qui a pour instance E'.*

Théorème 5.4.21 *Soient Γ un ensemble de clauses et Δ un ensemble d'instances des clauses de Γ, et $C_1, \ldots, C_n$ une preuve par résolution propositionnelle à partir de Δ, il existe une preuve $D_1, \ldots, D_n$ par résolution au 1º ordre à partir de Γ telle que pour i de 1 à n, la clause C_i est une instance de D_i.*

Preuve : Nous effectuons une récurrence sur n. Soit $C_1, \ldots, C_n, C_{n+1}$ une preuve par résolution propositionnelle à partir de Δ. Par récurrence, il existe une preuve $D_1, \ldots, D_n$ par résolution 1º ordre à partir Γ telle que pour i de 1 à n, la clause C_i est une instance de D_i. Nous distinguons deux cas :

1. Supposons $C_{n+1} \in \Delta$. Il existe $E \in \Gamma$ dont C_{n+1} est une instance donc nous prenons $D_{n+1} = E$.
2. Supposons que C_{n+1} soit un résolvant propositionnel de C_j et C_k où $j, k \leq n$. D'après le résultat précédent, il existe E résolvant au 1º ordre de D_j et D_k : nous prenons $D_{n+1} = E$.

□

Nous en déduisons immédiatement le corollaire suivant.

Corollaire 5.4.22 *Soient Γ un ensemble de clauses et Δ un ensemble d'instances des clauses de Γ. Supposons que $\Delta \vdash_p C$. Il existe D telle que $\Gamma \vdash_{1r} D$ et C est une instance de D.*

Exemple 5.4.23 *Soit l'ensemble de clauses $P(f(x)) \vee P(u), \neg P(x) \vee Q(z), \neg Q(x) \vee \neg Q(y)$. La fermeture universelle de cet ensemble de clauses est insatisfaisable et nous le montrons de trois manières.*

1. *Par instanciation sur le domaine de Herbrand $\{f^n(a), n \in \mathbb{N}\}$:*
   — *$P(f(x)) \vee P(u)$ est instanciée par $x := a, u := f(a)$ en $P(f(a))$*
   — *$\neg P(x) \vee Q(z)$ est instanciée par $x := f(a), z := a$ en $\neg P(f(a)) \vee Q(a)$*
   — *$\neg Q(x) \vee \neg Q(y)$ est instanciée par $x := a, y := a$ en $\neg Q(a)$*

   L'ensemble de ces 3 instanciations est insatisfaisable, comme le montre la preuve par résolution propositionnelle ci-dessous :

$$\dfrac{\dfrac{P(f(a)) \quad \neg P(f(a)) \vee Q(a)}{Q(a)} \qquad \neg Q(a)}{\perp}$$

2. *Cette preuve par résolution propositionnelle est relevée en une preuve par la règle de résolution au premier ordre :*

$$\dfrac{\dfrac{P(f(x)) \vee P(u) \quad \neg P(x) \vee Q(z)}{Q(z)} \qquad \neg Q(x) \vee \neg Q(y)}{\perp}$$

3. *Chaque règle de résolution au premier ordre est décomposée en factorisation, copie et résolution binaire :*

$$\dfrac{\dfrac{\dfrac{P(f(x)) \vee P(u)}{P(f(x))}\,Fact \quad \dfrac{\neg P(x) \vee Q(z)}{\neg P(y) \vee Q(z)}\,Copie}{Q(z)}\,RB \qquad \dfrac{\neg Q(x) \vee \neg Q(y)}{\neg Q(x)}\,Fact}{\perp}\,RB$$

Théorème 5.4.24 (Complétude réfutationnelle de la résolution au 1^o ordre) *Soit* Γ *un ensemble de clauses. Les propositions suivantes sont équivalentes :*

1. $\Gamma \vdash_{1r} \perp$.
2. $\Gamma \vdash_{1fcb} \perp$.
3. $\forall(\Gamma) \models \perp$.

Preuve :
- Nous savons que (1) implique (2), car la résolution au 1^o ordre est une combinaison de factorisation, copie et résolution binaire.
- Nous savons que (2) implique (3), car la factorisation, la copie et la résolution binaire sont cohérentes.
- Il nous reste à montrer que (3) implique (1). Supposons que $\forall(\Gamma) \models \perp$, autrement dit $\forall(\Gamma)$ est insatisfaisable. D'après le théorème de Herbrand, il y a Δ un ensemble fini d'instances sans variable de clauses de Γ qui n'a pas de modèle propositionnel. Par complétude de la résolution propositionnelle, nous avons : $\Delta \vdash_p \perp$. D'après le corollaire au relèvement 5.4.22 page précédente, il existe D telle que $\Gamma \vdash_{1r} D$ et $\perp$ est instance de D. Dans ce cas, nous avons $D = \perp$.

□

5.5 Outil logiciel

Nous indiquons ici un logiciel pour expérimenter le système de déduction décrit dans ce chapitre :

`http://teachinglogic.univ-grenoble-alpes.fr/ResBinSc/`

Cet outil, de façon similaire à celui proposé pour la déduction naturelle propositionnelle au paragraphe 3.5 page 66, permet de saisir une liste de clauses du premier ordre. Il tente alors automatiquement d'en déduire la clause vide :
- si cette liste de clauses est insatisfaisable et qu'aucune limite de temps ni de taille n'a été fixée, alors la clause vide sera déduite ;
- il est possible de restreindre l'espace de recherche, en imposant une limite de temps ou de taille sur les clauses produites (mais alors on perd la complétude du prouveur).

Si la liste de clauses est insatisfaisable, on obtient une preuve utilisant les règles de copie, factorisation et résolution binaire, annotée par les unificateurs utilisés.

5.6 Exercices

Exercice 80 (Domaine de Herbrand) *Soient Σ la signature composée de la constante a et des symboles de fonctions f et g respectivement à un et deux arguments.*

— *Donner 5 éléments différents du domaine de Herbrand de cette signature.*
— *Définir inductivement ce domaine.*

□

Exercice 81 (Signature, domaine et base de Herbrand) *Pour chacun de ces ensembles de formules :*

— $\Gamma_1 = \{P(x) \vee Q(x) \vee R(x), \neg P(a), \neg Q(b), \neg R(c)\}$.
— $\Gamma_2 = \{P(x), \neg Q(x), \neg P(f(x)) \vee Q(f(x))\}$.
— $\Gamma_3 = \{P(x), Q(f(x)), \neg R(f(f(x))), \neg P(f(f(x))) \vee \neg Q(x) \vee R(f(x))\}$.

1. *Donner la signature, le domaine de Herbrand ainsi que la base de Herbrand.*
2. *Prouver si leur fermeture universelle a un modèle ou pas.*

□

Exercice 82 (Méthode de Herbrand) *Utiliser la méthode de Herbrand pour démontrer que l'ensemble de formules suivant est insatisfaisable :*

1. $\forall x R(x, f(x))$
2. $\forall x \forall y (\neg S(x,y) \vee R(x,y) \vee R(y,x))$
3. $\forall x \forall y (S(x,y) \vee \neg R(x,y))$
4. $\forall x \forall y (S(x,y) \vee \neg R(y,x))$
5. $\forall y \neg S(y,a)$

□

Exercice 83 (Méthode de Herbrand,*) *Utiliser la méthode de Herbrand pour démontrer que l'ensemble de formules suivant est insatisfaisable :*

1. $\forall x (Q(x) \vee \neg P(f(x)))$
2. $\forall y (Q(y) \Rightarrow R(y))$
3. $\forall z (\neg P(z) \Rightarrow Q(z) \vee R(z))$
4. $\forall u \neg R(u)$

En particulier, vous devez transformer votre ensemble d'instances insatisfaisable en un ensemble de clauses équivalent. Puis, vous devez démontrer la contradiction via une preuve par résolution propositionnelle à partir de ce dernier ensemble. □

Exercice 84 (Méthode de Herbrand,*) *Soit Γ l'ensemble de formules suivant :*

1. $\neg S(x,y) \vee \neg M(z,x) \vee M(z,y)$
2. $S(x,y) \vee M(f(x,y),x)$
3. $S(x,y) \vee \neg M(f(x,y),y)$
4. $S(c,a)$
5. $S(a,b)$
6. $\neg S(c,b)$

Déterminer un ensemble fini insatisfaisable d'instances fermées de ces formules.
On peut en déduire quelque chose à propos d'un ensemble de formules du 1erordre : lequel et que peut-on conclure ? □

Exercice 85 (Modèle de Herbrand,*)** *Soit Δ l'ensemble de formules suivant :*

1. $x < y \wedge y < z \Rightarrow x < z$.
2. $\neg(x < x)$.
3. $x < y \Rightarrow x < f(x,y) \wedge f(x,y) < y$.
4. $a < b$.

— *Indiquer un modèle de $\forall(\Delta)$.*

— $\forall(\Delta)$ *a-t-il un modèle sur le domaine de Herbrand engendré par* a, b, f *?*

□

Exercice 86 (Skolémisation et forme clausale) *Skolémiser les formules suivantes (attention aux négations!) puis les mettre en forme clausale.*

1. $\neg(\exists x P(x) \vee \exists x Q(x) \Rightarrow \exists x(P(x) \vee Q(x)))$.
2. $\neg(\forall x \forall y \forall z(e(x,y) \wedge e(y,z) \Rightarrow \neg e(x,z)) \Rightarrow \neg\exists x \forall y\, e(x,y))$.
3. $\neg(\neg\forall x P(x) \vee \neg\forall x Q(x) \Rightarrow \neg(\forall x P(x) \wedge \forall x Q(x)))$.
4. $\forall x((\exists y P(x,y) \Rightarrow \exists x Q(x)) \wedge \exists y P(x,y) \wedge \neg\exists x Q(x))$.
5. $\neg(\exists x \forall y \forall z((P(y) \Rightarrow Q(z)) \Rightarrow (P(x) \Rightarrow Q(x))))$.

□

Exercice 87 (Unification) *Les termes suivants sont-ils unifiables ? Si la réponse est oui, donner leur unificateur le plus général, sinon justifier la réponse négative.*

— $h(g(x), f(a,y), z)$ *et* $h(y, z, f(u,x))$.

— $h(g(x), f(a,y), z)$ *et* $h(y, z, f(u, g(x)))$.

□

Exercice 88 (Unification) *Donner les unificateurs les plus généraux des termes suivants s'ils existent :*

1. $R(a, f(x)) = R(y, f(g(y,b)))$,
2. $R(y, f(x)) = R(x, f(g(y,b)))$,
3. $Q(y, f(a), f(x)) = Q(g(a,b), x, f(y))$, *et*
4. $Q(y, f(x), g(y,x)) = Q(x, f(y), g(f(a), y))$.

□

Exercice 89 (Unification) *Donner les unificateurs les plus généraux des termes suivants s'ils existent :*

1. $pair(a, crypt(z,b))$ *et* $pair(x,y)$.
2. $pair(crypt(x,b), crypt(y,b))$ *et* $pair(crypt(a,b), z)$.
3. $crypt(pair(z,a), x)$ *et* $crypt(pair(y, crypt(x,b)), b)$.
4. $crypt(pair(a,z), x)$ *et* $crypt(pair(y, crypt(x,b)), b)$.
5. $f(x, y, g(a,a))$ *et* $f(g(y,y), z, z)$
6. $f(x, y, a)$ *et* $f(y, g(z,z), x)$

□

Exercice 90 (Unification avec plusieurs solutions) *L'équation* $f(g(y), y) = f(u, z)$ *a deux solutions « les plus générales »(rappel : elles sont donc équivalentes). Indiquer ces deux solutions.*

□

Exercice 91 (Forme clausale et instanciation,*) *Montrer que la formule suivante est valide en mettant* sa négation *sous la forme clausale et en trouvant un ensemble contradictoire d'instances des clauses obtenues :*

$$\forall x(P(x) \Rightarrow Q(x)) \Rightarrow (\forall x P(x) \Rightarrow \forall x Q(x)).$$

□

Exercice 92 (Forme clausale et preuve par résolution,)** *Considérons les formules suivantes :*

1. $H_1 = \exists x P(x) \Rightarrow \forall x P(x)$.
2. $H_2 = \forall x(P(x) \vee Q(x))$.
3. $C = \exists x \neg Q(x) \Rightarrow \forall x P(x)$.

Nous voulons montrer que C *est conséquence de* H_1 *et* H_2 *par instanciation et par résolution.*

1. *Mettre en forme clausale l'ensemble des trois formules* H_1, H_2, $\neg C$.
2. *Trouver des instances contradictoires des clauses obtenues et montrer par résolution propositionnelle que ces instances sont contradictoires.*

3. *Donner une preuve directe de cette contradiction par factorisation, copie et résolution binaire. Il est possible que seule la dernière règle soit utilisée.*

□

Exercice 93 (Preuve par résolution,)**

— *En utilisant une preuve par factorisation, copie et résolvant binaire prouver que la fermeture universelle de l'ensemble de clauses suivant est insatisfaisable :*
  1. $P(f(x)) \vee \neg Q(y,a)$.
  2. $Q(a,a) \vee R(x,x,b) \vee S(a,b)$.
  3. $S(a,z) \vee \neg R(x,x,b)$.
  4. $\neg P(f(c)) \vee R(x,a,b)$.
  5. $\neg S(y,z) \vee \neg S(a,b)$.

— *En utilisant une preuve par factorisation, copie et résolvant binaire prouver que la fermeture universelle de l'ensemble de clauses suivant est insatisfaisable :*
  1. $P(x)$.
  2. $\neg P(y) \vee Q(y,x)$.
  3. $\neg Q(x,a) \vee \neg Q(b,y) \vee \neg Q(b,a) \vee \neg P(f(y))$.

— *En utilisant une preuve par factorisation, copie et résolvant binaire prouver que la fermeture universelle de l'ensemble de clauses suivant est insatisfaisable :*
  1. $R(x,a)$.
  2. $R(y,b)$.
  3. $R(z,a+b)$.
  4. $\neg(b < x+b) \vee (a+b < a)$.
  5. $b < z$.
  6. $\neg(a+b < x) \vee \neg R(y,b)$.

  Rappel : *le symbole + est plus prioritaire que le symbole <, qui est lui-même plus prioritaire que les connecteurs logiques binaires.*

□

Exercice 94 (Unification, résolution) *Soient* $\Gamma_1 = \{P(x, f(x,b), u), \neg P(g(a), z, h(z))\}$ *l'ensemble des* deux *clauses unitaires et* Γ_2 *l'ensemble des* deux *clauses unitaires* $\{P(x, f(x,b), u), \neg P(g(z), z, h(z))\}$. *Les ensembles* $\forall(\Gamma_1)$ *et* $\forall(\Gamma_2)$ *sont-ils satisfaisables ou insatisfaisables ?* □

Exercice 95 (Skolémisation et preuve par résolution) *Le but de cet exercice est de démontrer le syllogisme suivant :*

$$\forall x(homme(x) \Rightarrow mortel(x)) \wedge homme(Socrate) \Rightarrow mortel(Socrate)$$

— *Skolémiser la* **négation** *du syllogisme.*
— *Transformer la forme de Skolem obtenue en forme clausale.*
— *Démontrer par instanciation que la* **négation** *du syllogisme est une contradiction.*
— *Démontrer en utilisant une preuve par factorisation, copie et résolution binaire que la négation du syllogisme est une contradiction.*

□

Exercice 96 (Forme clausale et preuve par résolution,)** *Considérons les formules suivantes :*

1. $A_1 = \exists u \forall v(P(u) \wedge (R(v) \Rightarrow Q(u,v)))$.
2. $A_2 = \forall u \forall v(\neg P(u) \vee \neg S(v) \vee \neg Q(u,v))$.
3. $A_3 = \exists v(R(v) \wedge S(v))$.

Montrer que la liste de ces trois formules est contradictoire par résolution :

1. *Mettre en forme clausale les trois formules* A_1, A_2, A_3 .

2. *Trouver des instances contradictoires des clauses obtenues et montrer cette contradiction par résolution propositionnelle.*
3. *Faire une preuve directe de $\perp$ par factorisation, copie et résolution binaire.*

□

Exercice 97 (☞Cohérence de la résolution au premier ordre) *Prouver la cohérence de la résolution au premier ordre, c'est-à-dire :*

Soit Γ un ensemble de clauses et C une clause. Si $\Gamma \vdash_{1fcb} C$ alors $\forall(\Gamma) \models \forall(C)$.

□

Exercice 98 (Forme clausale et preuve par résolution,)** *Considérons les formules suivantes :*

1. $E_1 = \forall x(x = x)$.
2. $E_2 = \forall x \forall y(x = y \Rightarrow y = x)$.
3. $E_3 = \forall x \forall y \forall z \forall t(x = y \wedge z = t \Rightarrow (x \in z \Rightarrow y \in t))$.
4. $C = \forall y \forall z(y = z \Rightarrow \forall x(x \in y \Leftrightarrow x \in z))$.

E_1, E_2, E_3 sont des axiomes de l'égalité : E_1 exprime que l'égalité est réflexive, E_2 qu'elle est symétrique, E_3 que c'est une congruence relativement à l'appartenance. La transitivité de l'égalité ne sert pas dans cet exercice. C exprime que deux ensembles égaux ont les mêmes éléments. Nous notons que dans cet exercice, le symbole égal est traité comme un symbole de prédicat « ordinaire » dont le sens n'est pas fixé comme étant l'identité, mais que nous « définissons » par ses propriétés. Autrement dit l'interdiction de l'égalité dans les formules, c'est en réalité l'interdiction de l'égalité comme symbole de sens fixé. Montrer que C est conséquence des axiomes de l'égalité.

1. *Mettre en forme clausale les trois formules E_1, E_2, E_3, $\neg C$.*
2. *Trouver des instances contradictoires des clauses obtenues.*
3. *Faire une preuve directe de $\perp$ par factorisation, copie et résolution binaire.*

□

Exercice 99 (Forme clausale et preuve par résolution,)** *Considérons les formules suivantes :*

1. $H_1 = \forall u(\exists v R(u,v) \Rightarrow R(u, f(u)))$.
2. $H_2 = \forall u \exists v R(u,v)$.
3. $H_3 = \exists u R(f(f(u)), u)$.
4. $C = \exists u \exists v \exists w(R(u,v) \wedge R(v,w) \wedge R(w,u))$.

Montrer que C est conséquence de H_1, H_2, H_3.

1. *Mettre en forme clausale les trois formules H_1, H_2, H_3, $\neg C$.*
2. *Trouver des instances contradictoires des clauses obtenues (montrez la contradiction par résolution propositionnelle).*
3. *Faire une preuve directe de cette contradiction par factorisation, copie et résolution binaire.*

□

Exercice 100 (Forme clausale et preuve par résolution,)** *Soit $P(x,y)$ une formalisation de « x est père de y ». Soit $A(x,y)$ une formalisation de « x est aïeul de y ». Soient les trois formules :*

1. $H_1 = \forall x \exists y P(x,y)$.
2. $H_2 = \forall x \forall y \forall z(P(x,y) \wedge P(y,z) \Rightarrow A(x,z))$.
3. $C = \forall x \exists y A(x,y)$.

Montrons que C est conséquence de H_1, H_2, de la façon suivante, transformer $H_1, H_2, \neg C$ en leur forme clausale, puis trouver des instances contradictoires des clauses obtenues (montrez la contradiction par résolution propositionnelle). Enfin, dériver la clause vide par factorisation, copie, résolution binaire. □

Exercice 101 (Formule de Bernays-Schonfinkel,*)** *Une formule BS (de Bernays-Schonfinkel) est une formule fermée,* sans *symbole de fonction, de la forme $\exists x_1 \ldots \exists x_n \forall y_1 \ldots \forall y_p$ B où B est sans quantificateur. Expliquer pourquoi nous pouvons décider la satisfaisabilité d'une telle formule.*

□

Chapitre 6

Déduction naturelle au premier ordre : quantificateurs, copie et égalité

Sommaire

En plus des règles du chapitre 3, nous ajoutons les règles concernant les quantificateurs, la copie et l'égalité. Notre système comporte toujours une seule règle pour enlever les hypothèses (la règle d'introduction de l'implication). Les définitions de brouillon de preuve, environnement, contexte, formule utilisable restent inchangées. Nous montrons la cohérence des règles de notre système, mais nous admettons sans preuve, que ce système est complet : des preuves de complétude pour des systèmes de règles similaires peuvent être trouvées dans les livres [2, 11]. Contrairement au cas propositionnel, *il n'y a aucun algorithme* pour décider si une formule est valide ou non valide (preuve donnée par Alonzo Church et Alan Türing en 1936 et 1937 [4, 24]). Autrement dit, en admettant l'équivalence entre prouvable (sans environnement) et valide, il n'y a pas d'algorithme qui, étant donné une formule, puisse nous en construire la preuve, ou nous avertir que cette formule n'a pas de preuve.

Plan : Nous étendons d'abord les règles de la déduction naturelle proposées pour la logique propositionnelle. Nous présentons ensuite deux nouvelles tactiques pour aider à la rédaction de preuves. Enfin, nous prouvons la cohérence de notre système.

6.1 Règles pour la logique du premier ordre

Notre système de règles se divise en plusieurs familles de règles :
— les règles « propositionnelles »,
— les règles d'introduction et d'élimination de quantificateur,
— les règles régissant l'égalité,
— et une règle de copie.

L'ensemble des règles définies dans la section 3 page 57 peuvent être utilisées dans notre système de déduction. Nous renvoyons le lecteur à la table 3.1 page 59, qui récapitule l'ensemble de ces règles. Dans la suite de cette section, nous présentons uniquement les règles de déduction naturelle dédiées à la logique du premier ordre.

6.1.1 Règles des quantificateurs

L'ensemble des règles d'introduction et d'élimination de quantificateur est présenté dans la table 6.1. Dans cette table, A et B sont des formules, x est une variable, t est un terme. Notez que contrairement aux règles « propositionnelles », l'usage de ces nouvelles règles est contraint par des conditions d'emploi, qui utilisent les notions de *variable libre* (Définition 4.2.3 page 77) et de *terme libre pour une variable* (Définition 4.3.34 page 82).

Dans la suite, nous détaillons ces règles et nous illustrons leur usage sur des exemples et des contre-exemples montrant les erreurs occasionnées par le non respect des conditions d'emploi.

Règles	**Conditions d'emploi**
$\dfrac{\forall xA}{A{<}x:=t{>}}\ \forall E$	
$\dfrac{A}{\forall xA}\ \forall I$	
$\dfrac{\exists xA \quad (A \Rightarrow B)}{B}\ \exists E$	
$\dfrac{A{<}x:=t{>}}{\exists xA}\ \exists I$	

TABLE 6.1 – Règles d'introduction et d'élimination des quantificateurs.

6.1.1.1 Règles pour le quantificateur universel

La règle d'élimination du quantificateur universel, notée $\forall E$, signifie que si la formule A est vraie pour toute valeur de x alors toute instance de A où x est remplacé par **un terme t libre pour** x est vraie. Notez que cette règle est une simple application du corollaire 4.3.38, page 83.

$$\frac{\forall xA}{A{<}x:=t{>}}\ \forall E$$

Nous justifions ci-dessous les conditions d'emploi de cette règle par un exemple montrant qu'un usage incorrect de la règle peut conduire à prouver une formule non-valide.

Exemple 6.1.1 *Nous montrons qu'un usage incorrect de la* **règle** $\forall E$ *conduit à « prouver » une formule non valide.*

contexte	*numéro*	*ligne*	*règle*
1	*1*	Supposons $\forall x\exists yP(x,y)$	
1	*2*	$\exists yP(y,y)$	$\forall E$ *1, y ERREUR*
	3	Donc $\forall x\exists yP(x,y) \Rightarrow \exists yP(y,y)$	$\Rightarrow I$ *1,2*

À la ligne 2, nous n'avons pas respecté les conditions d'applications de la règle $\forall E$ car le terme y n'est pas libre pour x dans la formule $\exists yP(x,y)$. Nous donnons une interprétation qui est un contre-modèle de la « conclusion » : soit I l'interprétation de domaine $\{0,1\}$ avec $P_I =$

La règle d'introduction du quantificateur universel, notée $\forall I$, signifie que si nous avons pu déduire la formule A indépendamment de la valeur de x, nous pouvons généraliser en déduisant $\forall xA$. Ainsi, il est nécessaire que x **ne soit libre ni dans l'environnement de la preuve, ni dans le contexte de la ligne où nous avons déduit** A.

$$\frac{A}{\forall xA}\ \forall I$$

Nous illustrons maintenant l'emploi de la règle d'introduction du quantificateur universel avec un exemple d'usage correct (exemple 6.1.2) et un exemple d'usage incorrect (exemple 6.1.3) de la règle.

Exemple 6.1.2 *Nous prouvons* $\forall y P(y) \wedge \forall y Q(y) \Rightarrow \forall x(P(x) \wedge Q(x))$.

contexte	*numéro*	*ligne*	*règle*
1	*1*	Supposons $\forall y P(y) \wedge \forall y Q(y)$	
1	*2*		
1	*3*		
1	*4*		
1	*5*		
1	*6*		
1	*7*		
	8	Donc $\forall x P(x) \wedge \forall x Q(x) \Rightarrow \forall x(P(x) \wedge Q(x))$	$\Rightarrow$ *I 1,7*

Exemple 6.1.3 *Nous montrons qu'un usage incorrect de la* **règle** $\forall I$ *conduit à « prouver » une formule non valide.*

contexte	*numéro*	*ligne*	*règle*
1	*1*	Supposons $P(x)$	
1	*2*	$\forall x P(x)$	$\forall I$ *1 ERREUR*
	3	Donc $P(x) \Rightarrow \forall x P(x)$	$\Rightarrow$ *I 1,2*

À la ligne 2, nous n'avons pas respecté les conditions d'applications de la règle $\forall I$ *car la prémisse* $P(x)$ *est établie dans le contexte* $P(x)$*, ce qui interdit de généraliser sur x. Nous donnons un contre-modèle de la « conclusion » :*

6.1.1.2 Règles pour le quantificateur existentiel

La règle d'élimination du quantificateur existentiel, notée $\exists E$, signifie que si $\exists x A$ est vraie et que nous pouvons déduire $A \Rightarrow B$ indépendemment de la valeur de x, alors nous pouvons déduire B, si la formule B ne dépend, elle non plus, de la valeur de x. Ainsi, il est nécessaire que x **ne soit libre ni dans l'environnement de la preuve, ni dans B, et ni dans le contexte de la ligne $A \Rightarrow B$.**

$$\frac{\exists x A \quad (A \Rightarrow B)}{B} \; \exists E$$

La condition d'emploi de cette règle étant complexe, nous illustrons l'emploi de la règle avec deux exemples d'usage incorrect.

Exemple 6.1.4 *Nous montrons qu'un usage incorrect de la* **règle** $\exists E$ *conduit à « prouver » une formule non valide.*

contexte	*numéro*	*ligne*	*règle*
1	*1*	Supposons $\exists x P(x) \wedge (P(x) \Rightarrow \forall y Q(y))$	
1	*2*	$\exists x P(x)$	$\wedge$*E1 1*
1	*3*	$P(x) \Rightarrow \forall y Q(y)$	$\wedge$*E2 1*
1	*4*	$\forall y Q(y)$	$\exists E$ *2,3 ERREUR*
	5	Donc $\exists x P(x) \wedge (P(x) \Rightarrow \forall y Q(y)) \Rightarrow \forall y Q(y)$	$\Rightarrow$ *I 1,4*

Nous n'avons pas respecté la condition que le contexte de la prémisse $P(x) \Rightarrow \forall y Q(y)$ ne doit pas dépendre de x. Il est clair que la conclusion obtenue n'est pas valide. Nous donnons une assignation (I,e) qui est un contre-modèle de cette « conclusion » :

Exemple 6.1.5 *Nous montrons qu'un usage incorrect de la **règle** $\exists E$ conduit à « prouver » une formule non valide.*

contexte	numéro	ligne	règle
1	1	`Supposons` $\exists x P(x)$	
1,2	2	`Supposons` $P(x)$	
1	3	`Donc` $P(x) \Rightarrow P(x)$	$\Rightarrow$ I 2,2
1	4	$P(x)$	$\exists$E 1,3 ERREUR
1	5	$\forall x P(x)$	$\forall$I 4
	6	`Donc` $\exists x P(x) \Rightarrow \forall x\, P(x)$	$\Rightarrow$ I 1,5

La conclusion de la règle $\exists E$ est $P(x)$, contrairement à la condition d'application de cette règle qui impose que la conclusion ne doit pas dépendre de x. Nous donnons une interprétation qui est un contre-modèle de la cette « conclusion » :

La règle d'introduction du quantificateur existentiel, notée $\exists I$, signifie que si une instance de la formule A où x est remplacé par **un terme** t **libre pour** x est vraie, alors la formule $\exists x A$ est vraie. Notez que cette règle est une simple application du corollaire 4.3.38, page 83. Ce qui justifie la condition d'emploi. Pour s'en convaincre, il suffit d'étudier l'exemple 4.3.37, qui précède le corollaire 4.3.38.

Ci-dessous, nous donnons un exemple d'usage correct de la règle d'introduction du quantificateur existentiel en démontrant l'une des lois de De Morgan.

Exemple 6.1.6 (Lois de De Morgan) *Nous montrons que pour toute formule A, $\neg\forall x A \Rightarrow \exists x \neg A$.*

contexte	numéro	ligne	règle
1	1	`Supposons` $\neg\forall x A$	
1,	2		
1,	3		
1,	4		
1,	5		
1,	6		
1,	7		
1,	8		
1,	9		
1	10		
1	11		
	12	`Donc` $\neg\forall x A \Rightarrow \exists x \neg A$	$\Rightarrow$ I 1,11

6.1.2 Copie

La règle de *copie* consiste à déduire d'une formule A, une autre formule A' égale au changement près des variables liées, au sens de la définition 4.4.5, page 88. Par exemple, $\forall x \exists y P(x,y)$ est une copie de $\forall y \exists x P(y,x)$.

$$\frac{A'}{A} \text{ Copie}$$

Notez qu'il n'y pas de condition d'emploi pour cette règle.

6.1.3 Les règles de l'égalité

Deux règles caractérisent l'égalité : un terme est égal à lui-même (règle de la *réflexivité*) et si deux termes sont égaux, nous pouvons les remplacer l'un par l'autre (règle de la *congruence*).

$\dfrac{}{t=t}\ Réflexivité$	
$\dfrac{s=t \quad A<x:=s>}{A<x:=t>}\ Congruence$	

Nous remarquons que la première règle n'a pas de prémisse. C'est ce que nous appelons aussi un *axiome*. Nous remarquons aussi que les conditions d'emploi de la deuxième règle sont similaires à celles des règles $\forall E$ et $\exists I$. Ces conditions d'emploi se justifient de la même manière que précédemment.

Nous donnons maintenant deux exemples d'application des règles d'égalités.

Exemple 6.1.7 *Prouvons que $s=t \Rightarrow t=s$, autrement dit prouvons que l'égalité est symétrique.*

contexte	*numéro*	*ligne*	*règle*
1	*1*	Supposons $s=t$	
1	*2*		
1	*3*		
	4	Donc $s=t \Rightarrow t=s$	

Exemple 6.1.8 *Prouvons que $s=t \wedge t=u \Rightarrow s=u$, autrement dit prouvons que l'égalité est transitive.*

1	*1*	Supposons $s=t \wedge t=u$	
1	*2*		
1	*3*		
1	*4*		
	5	Donc $s=t \wedge t=u \Rightarrow s=u$	

6.2 Tactiques de preuves

Nous instroduisons deux nouvelles tactiques pour l'application des règles $\forall I$ et $\exists E$ et nous illustrons ces tactiques avec un exemple.

6.2.1 Raisonner en avant avec une hypothèse d'existence

Soient Γ un ensemble de formules, x une variable, A et C des formules. Supposons que nous cherchons une preuve de C dans l'environnement $\Gamma, \exists x A$.

1. Supposons que x n'est libre ni dans Γ, ni dans C. Dans ce cas, la preuve peut toujours s'écrire :

`Supposons` A	
$\boxed{\text{preuve de } C \text{ dans l'environnement } \Gamma, A}$	
`Donc` $A \Rightarrow C$	$\Rightarrow I$ 1,_
C	$\exists E$

2. Supposons que x soit libre dans Γ ou C. Nous choisissons une variable y « nouvelle », c'est-à-dire non libre dans Γ, C et absente de A, puis nous nous ramenons au cas précédent, via la règle de copie. La preuve s'écrit alors :

$\exists y A{<}x := y{>}$	Copie de $\exists x A$
`Supposons` $A{<}x := y{>}$	
$\boxed{\text{preuve de } C \text{ dans l'environnement } \Gamma, A{<}x := y{>}}$	
`Donc` $A{<}x := y{>} \Rightarrow C$	$\Rightarrow I$ 1,_
C	$\exists E$

La recherche de la preuve initiale a été réduite à la recherche d'une preuve dans un environnement plus simple. C'est exactement le mode de raisonnement appliqué dans les cours de mathématiques quand nous cherchons une preuve d'une formule C avec l'hypothèse $\exists x P(x)$. Nous introduisons une constante « nouvelle » a vérifiant $P(a)$ et nous prouvons C sous l'hypothèse $P(a)$.

6.2.2 Raisonner en arrière pour généraliser

Nous reprenons les notations du paragraphe précédent. Supposons que nous cherchons une preuve de $\forall x A$ dans l'environnement Γ.

1. Supposons que x n'est pas libre dans Γ. Dans ce cas, la preuve peut toujours s'écrire :

$\boxed{\text{preuve de } A \text{ dans l'environnement } \Gamma}$	
$\forall x A$	$\forall I$

2. Supposons que x est libre dans Γ. Nous choisissons une variable y « nouvelle », c'est-à-dire non libre dans Γ, puis nous nous ramenons au cas précédent, via la règle de copie. La preuve s'écrit alors :

$\boxed{\text{preuve de } A{<}x := y{>} \text{ dans l'environnement } \Gamma}$	
$\forall y A{<}x := y{>}$	$\forall I$
$\forall x A$	Copie de la formule précédente

La recherche de la preuve initiale a été réduite à la recherche d'une preuve d'une formule plus simple dans le même environnement. C'est exactement le mode de raisonnement appliqué dans les cours de mathématiques quand nous cherchons une preuve de $\forall x P(x)$. Nous introduisons une constante « nouvelle » a et nous prouvons $P(a)$. Puis nous ajoutons : puisque le choix de a est arbitraire, nous avons $\forall x P(x)$.

6.2.3 Un exemple d'application des tactiques

Nous notons « il existe un et un seul x » par $\exists! x$. Formellement, $\exists! x P(x)$ signifie $\exists x(P(x) \wedge \forall y(P(y) \Rightarrow x = y))$. En séparant l'existence de x et son unicité, nous pouvons aussi définir $\exists! x P(x)$ par $\exists x P(x) \wedge \forall x \forall y(P(x) \wedge P(y) \Rightarrow x = y)$. Ces deux définitions sont bien sûr équivalentes et nous montrons formellement que la première implique la deuxième. Comme la preuve est longue, il faut apprendre à décomposer les preuves.

Plan de la preuve Nous appliquons les deux tactiques suivantes :
- Pour prouver $A \Rightarrow B$, supposer A et déduire B
- Pour prouver $A \wedge B$, prouver A et prouver B.

1	`Supposons` $\exists x(P(x) \wedge \forall y(P(y) \Rightarrow x = y))$	
	preuve de $\exists x P(x)$ dans l'environnement de (1)	
	preuve de $\forall x \forall y(P(x) \wedge P(y) \Rightarrow x = y)$ dans l'environnement de (1)	
	$\exists x P(x) \wedge \forall x \forall y(P(x) \wedge P(y) \Rightarrow x = y)$	$\wedge I$
	`Donc` $\exists x(P(x) \wedge \forall y(P(y) \Rightarrow x = y)) \Rightarrow \exists x P(x) \wedge \forall x \forall y(P(x) \wedge P(y) \Rightarrow x = y)$	$\Rightarrow I$

Application de la tactique utilisant une hypothèse d'existence

Nous cherchons une preuve de $\exists x P(x)$ dans l'environnement de $\exists x(P(x) \wedge \forall y(P(y) \Rightarrow x = y))$. Nous appliquons la tactique « raisonner en avant en présence d'une hypothèse existentielle ».

référence	formule	
i	$\exists x(P(x) \wedge \forall y(P(y) \Rightarrow x = y))$	
1	`Supposons` $P(x) \wedge \forall y(P(y) \Rightarrow x = y)$	
2	$P(x)$	$\wedge E1$ 1
3	$\exists x P(x)$	$\exists I$ 2, x
4	`Donc` $P(x) \wedge \forall y(P(y) \Rightarrow x = y) \Rightarrow \exists x P(x)$	$\Rightarrow I$ 1,2
5	$\exists x P(x)$	$\exists E$ i, 3

Application de la tactique pour obtenir une conclusion générale

Nous cherchons une preuve de $\forall x \forall y(P(x) \wedge P(y) \Rightarrow x = y)$ dans l'environnement de $\exists x(P(x) \wedge \forall y(P(y) \Rightarrow x = y))$. Nous appliquons, dans l'ordre ci-dessous, les tactiques suivantes :
- « raisonner en avant en utilisant d'une hypothèse existentielle ».
- Pour prouver $A \Rightarrow B$, supposer A et déduire B
- « raisonner en arrière pour obtenir une conclusion générale ».

environnement			
référence		formule	
i		$\exists x(P(x) \wedge \forall y(P(y) \Rightarrow x = y))$	
contexte	numéro	preuve	règle
1	1	`Supposons` $P(x) \wedge \forall y(P(y) \Rightarrow x = y)$	
1	2		
1	3		
1	4		
1	5		
1	6		
1	7		
1	8		
1	9		
1	10		
1	11		
1	12		
1	13		
1	14		
	15		
	16	$\forall x \forall y(P(x) \wedge P(y) \Rightarrow x = y)$	

La tactique « utiliser une hypothèse d'existence » sert à produire la ligne 16. La tactique « introduire une implication » sert à produire la ligne 15 : elle introduit l'hypothèse (1) dans laquelle x est une variable libre. Donc, pour appliquer la tactique « obtenir une conclusion générale », il faut changer de variable dès la ligne 2. Observer aussi que la formule, ligne 3, est instanciée aux lignes 5 et 8.

Comme sur cet exemple, toute la difficulté des preuves est concentrée autour des règles $\forall I$ et $\exists E$:

— dans le raisonnement en avant, il faut trouver les bonnes instanciations des formules commençant par un quantificateur existentiel.

— dans le raisonnement en arrière, il faut trouver la bonne instance permettant de déduire une formule commençant par un quantificateur universel.

6.3 Cohérence du système

Nous commençons par montrer deux propriétés à propos des quantificateurs existensiel et universel. Enfin nous montrons la cohérence de notre système de déduction.

Propriété 6.3.1 *Soient Γ un ensemble de formules, x une variable et A une formule. Supposons que x ne soit pas libre dans Γ, alors nous avons : $\Gamma \models A$ si et seulement si $\Gamma \models \forall xA$.*

Preuve :

□

Si nous observons d'un œil critique cette preuve, nous observons que c'est une paraphrase, dans un autre formalisme, de la loi $\forall I$. C'est l'équivalence ci-dessus, qui explique que la tactique « raisonner en arrière pour généraliser », est une tactique utilisable sans risque. En effet, en admettant la complétude du système, lorsque x n'est pas libre dans Γ, il y a une preuve de A dans l'environnement Γ si et seulement s'il a une preuve de $\forall xA$ dans ce même environnement.

Propriété 6.3.2 *Soient Γ un ensemble de formules, x une variable, A et B deux formules. Supposons que x ne soit libre ni dans Γ, ni dans B, alors nous avons : $\Gamma \models A \Rightarrow B$ si et seulement si $\Gamma \models (\exists xA) \Rightarrow B$*

Preuve :

$\Rightarrow$ Supposons que $\Gamma \models A \Rightarrow B$. Soit (I,e) une assignation modèle de Γ. Puisque x n'est pas libre dans Γ, pour tout état f identique à e sauf pour la valeur de x, (I,f) et (I,e) donnent la même valeur aux formules de Γ, donc (I,f) est modèle de Γ. Puisque $\Gamma \models A \Rightarrow B$, pour tout état f identique à e sauf pour la valeur de x, (I,f) est modèle de $A \Rightarrow B$. Supposons que (I,e) est modèle de $\exists xA$, il existe g identique à e sauf pour la valeur de x tel que (I,g) est modèle de A. Puisque pour tout état f identique à e sauf pour la valeur de x, (I,f) est modèle de $A \Rightarrow B$, alors (I,g) est modèle de B. Puisque x n'est pas libre dans B, (I,e) est modèle de B.

$\Leftarrow$ Supposons que $\Gamma \models (\exists xA) \Rightarrow B$. Puisque la formule $A \Rightarrow (\exists xA)$ est valide (d'après le corollaire 4.3.38 page 83), nous avons $\Gamma \models A \Rightarrow B$.

□

Si nous observons d'un œil critique cette preuve, nous voyons que c'est une paraphrase, dans un autre formalisme, de la loi $\exists E$. C'est l'équivalence ci-dessus, qui explique que la tactique « raisonner en avant avec une hypothèse d'existence », est une tactique utilisable sans risque. En effet, en admettant la complétude du système, lorsque x n'est libre ni dans Γ, ni dans B, il y a une preuve de B dans l'environnement Γ, A si et seulement s'il a une preuve de B dans l'environnement $\Gamma, \exists xA$.

Théorème 6.3.3 (Cohérence de la déduction) *Si une formule est déduite d'un environnement de formules alors elle en est une conséquence.*

Preuve : La preuve obéit au même plan que celle du même théorème dans le cas propositionnel (voir théorème 3.3.1 page 65) et nous en reprenons les notations en traitant uniquement le cas des nouvelles règles.

Soit Γ un ensemble de formules. Soit P une preuve de A dans cet environnement. Soient C_i la conclusion et H_i le contexte de la i-ème ligne de la preuve P. Rappelons que les lignes d'une preuve sont numérotées à partir de 1 et que H_0 est la liste vide.

Notons par Γ, H_i l'ensemble des formules de l'ensemble Γ et de la liste H_i.

Cas de base : Supposons que A est déduite de Γ par la preuve vide. Alors A est élément de Γ, donc $\Gamma \models A$. Puisque H_0 est la liste vide, nous pouvons conclure : $\Gamma, H_0 \models A$.

Induction : Supposons que pour toute ligne $i < k$, $\Gamma, H_i \models C_i$. Montrons que $\Gamma, H_k \models C_k$.

Nous examinons uniquement le cas des nouvelles règles et pour simplifier nous ne faisons pas de distinctions entre deux formules égales aux abréviations près de la négation et de l'équivalence.

— Supposons que $C_k = \forall x A$ et que cette ligne ait été déduite, par la règle $\forall I$, de la formule A avec $A = C_i$ et $0 \leq i < k$ ou $A \in \Gamma$. Si $A = C_i$ et $0 < i < k$, par hypothèse de récurrence nous avons, $\Gamma, H_i \models A$. Si $A \in \Gamma$ alors $\Gamma \models A$. Puisque H_0 est la liste vide, il existe i où $0 \leq i < k$ tel que $\Gamma, H_i \models A$. Vu les conditions d'application de la règle, x n'est pas libre dans Γ, H_i. Donc, d'après la propriété 6.3.1 page ci-contre, nous avons aussi $\Gamma, H_i \models \forall x A$. Puisque la ligne i est utilisable sur la ligne $k-1$ et que H_0 est la liste vide, H_i est préfixe de H_{k-1}. Puisque le contexte n'est pas modifié par la ligne k, nous avons $H_{k-1} = H_k$, donc $\Gamma, H_k \models C_k$.

— Supposons que $C_k = A{<}x := t{>}$ et que cette ligne ait été déduite, par la règle $\forall E$, de la formule $\forall x A$ avec $\forall x A = C_i$ et $0 < i < k$ ou $\forall x A \in \Gamma$. Par hypothèse de récurrence ou parce que H_0 est la liste vide, il existe i où $0 \leq i < k$ tel que $\Gamma, H_i \models \forall x A$. D'après les conditions d'application de la règle, le terme t est libre pour la variable x dans la formule A. Donc, d'après le corollaire 4.3.38 page 83, la formule $\forall x A \Rightarrow A{<}x := t{>}$ est valide et par suite $\Gamma, H_i \models A{<}x := t{>}$. Puisque la ligne i est utilisable sur la ligne $k-1$, et que H_0 est la liste vide, H_i est préfixe de H_{k-1}. Puisque le contexte n'est pas modifié par la ligne k, nous avons $H_{k-1} = H_k$, donc $\Gamma, H_k \models C_k$.

— Supposons que $C_k = \exists x A$ et que cette ligne ait été déduite, par la règle $\exists I$, de la formule $A{<}x := t{>}$ avec $A{<}x := t{>} = C_i$ et $0 < i < k$ ou $A{<}x := t{>} \in \Gamma$. Par hypothèse de récurrence ou parce H_0 est la liste vide, il existe i où $0 \leq i < k$ tel que $\Gamma, H_i \models A{<}x := t{>}$. D'après les conditions d'applications de la règle, le terme t est libre pour la variable x dans la formule A. Donc, d'après le corollaire 4.3.38 page 83, la formule $A{<}x := t{>} \Rightarrow \exists x A$ est valide et par suite $\Gamma, H_i \models \exists x A$. Puisque la ligne i est utilisable sur la ligne $k-1$, et que H_0 est la liste vide, H_i est préfixe de H_{k-1}. Puisque le contexte n'est pas modifié par la ligne k, nous avons $H_{k-1} = H_k$, donc $\Gamma, H_k \models C_k$.

— Supposons que $C_k = B$ et que cette formule ait été déduite, par la règle $\exists E$, de la formule $\exists x A$ avec $\exists x A = C_i$ et $0 < i < k$ ou $\exists x A \in \Gamma$ et de la formule $A \Rightarrow B$ avec $A \Rightarrow B = C_j$ et $0 < j < k$ ou $A \Rightarrow B \in \Gamma$. Par hypothèse de récurrence ou parce H_0 est la liste vide, il existe i et j tels que $0 \leq i < k$, $0 \leq j < k$, $\Gamma, H_i \models \exists x A$ et $\Gamma, H_j \models A \Rightarrow B$. Vu les conditions d'application de la règle, x n'est libre ni dans Γ, H_j, ni dans B. Donc, d'après la propriété 6.3.2 page précédente, nous avons aussi $\Gamma, H_j \models (\exists x A) \Rightarrow B$. Puisque les lignes i et j sont utilisables sur la ligne $k-1$, et que H_0 est la liste vide, H_i et H_j sont préfixes de H_{k-1}. Puisque le contexte n'est pas modifié par la ligne k, nous avons $H_{k-1} = H_k$, donc $\Gamma, H_k \models \exists x A$ et $\Gamma, H_k \models (\exists x A) \Rightarrow B$. Par suite $\Gamma, H_k \models C_k$.

— Supposons que $C_k = A'$ et que cette formule ait été déduite, par la règle de copie de la formule A avec $A = C_i$ et $0 < i < k$ ou $A \in \Gamma$. Par hypothèse de récurrence ou parce H_0 est la liste vide, il existe i tel que $0 \leq i < k$, $\Gamma, H_i \models A$. Puisque, d'après le théorème 4.4.6 page 88, les formules A et A' sont équivalentes, nous avons $\Gamma, H_i \models A'$. Puisque la ligne i est utilisable sur la ligne $k-1$, et que H_0 est la liste vide, H_i est préfixe de H_{k-1}. Puisque le contexte n'est pas modifié par la ligne k, nous avons $H_{k-1} = H_k$, donc $\Gamma, H_k \models C_k$.

— Supposons que $C_k = (t = t)$. Puisque cette formule est valide (dans le sens attribué à l'égalité), $\Gamma, H_k \models C_k$.

— Supposons que $C_k = A{<}x := t{>}$ et que cette ligne ait été déduite, par la règle de congruence, de la formule $s = t$ avec $(s = t) = C_i$ et $0 < i < k$ ou $(s = t) \in \Gamma$ et de la formule $A{<}x := s{>}$ avec $A{<}x := s{>} = C_j$ et $0 < j < k$ ou $A{<}x := s{>} \in \Gamma$. Par hypothèse de récurrence ou parce H_0 est la liste vide, il existe i et j tels que $0 \leq i < k$, $0 \leq j < k$, $\Gamma, H_i \models (s = t)$ et $\Gamma, H_j \models A{<}x := s{>}$. Puisque les lignes i et j sont utilisables sur la ligne $k-1$, et que H_0 est la liste vide, H_i et H_j sont préfixes de H_{k-1}. Puisque le contexte n'est pas modifié par la ligne k, nous avons $H_{k-1} = H_k$, donc $\Gamma, H_k \models (s = t)$ et $\Gamma, H_k \models A{<}x := s{>}$. D'après le théorème 4.3.36 page 83 et les conditions d'application de la règle, nous avons : $s = t, A{<}x := s{>} \models A{<}x := t{>}$. Par suite $\Gamma, H_k \models C_k$.

Puisque la dernère ligne d'une preuve a un contexte vide, nous obtenons que sa conclusion est conséquence de Γ. □

6.4 Exercices

Exercice 102 (Déductions naturelles) *Prouver en déduction naturelle du premier ordre les formules suivantes :*

1. *le fameux syllogisme, « Tout homme est mortel, Socrate est un homme, donc est mortel », que nous formalisons par* $\forall x(H(x) \Rightarrow M(x)) \wedge H(socrate) \Rightarrow M(socrate)$.
2. $\forall x P(x) \Rightarrow \exists y P(y)$.
3. $\forall x P(x) \Rightarrow \exists x P(x)$.
4. $\forall x(P(x) \wedge Q(x)) \Rightarrow \forall x P(x) \wedge \forall x Q(x)$. *Notez que l'exemple 6.1.2 page 117 donne la preuve de la réciproque.*
5. $\exists x(P(x) \vee Q(x)) \Rightarrow \exists x P(x) \vee \exists x Q(x)$.
6. $\forall x(P(x) \Rightarrow Q(x)) \wedge \exists x P(x) \Rightarrow \exists x Q(x)$.
7. $\exists x P(x) \vee \exists x Q(x) \Rightarrow \exists x(P(x) \vee Q(x))$. *(**)*
8. $\forall x(P(x) \Rightarrow Q(x)) \wedge \forall x(Q(x) \Rightarrow R(x)) \Rightarrow \forall x(P(x) \Rightarrow R(x))$.
9. $\forall x\ (P(x) \Rightarrow Q(x)) \wedge \exists x\ \neg Q(x) \Rightarrow \exists x\ \neg P(x)$.

Dans cet exercice, les formules $P(x)$ et $Q(x)$ peuvent être remplacées par des formules quelconques. □

Exercice 103 (Déductions naturelles) *Prouver les formules suivantes :*

1. $\forall x \forall y P(x,y) \Rightarrow \forall y \forall x P(x,y)$.
2. $\exists x \exists y P(x,y) \Rightarrow \exists y \exists x P(x,y)$.
3. $\exists x \forall y P(x,y) \Rightarrow \forall y \exists x P(x,y)$.
4. $\forall x(Q(x) \Rightarrow \forall y(R(y) \Rightarrow P(x,y))) \Rightarrow \forall y(R(y) \Rightarrow \forall x(Q(x) \Rightarrow P(x,y)))$. *(*)*

Dans cet exercice, la formule $P(x,y)$ peut être remplacée par une formule quelconque. Par contre $Q(x)$ peut être remplacée seulement par une formule n'ayant pas y comme variable libre, et $R(y)$ peut être remplacée par une formule n'ayant pas x comme variable libre : expliquez la raison de ces contraintes. □

Exercice 104 (Chercher la faute) *Considérons la formule suivante :*

$$\exists x\ P(x) \wedge \forall x\ Q(x) \Rightarrow \exists x\ (P(x) \wedge Q(x)).$$

Parmi les trois preuves suivantes une seule est correcte par déduction naturelle. Identifier la preuve correcte et justifier pourquoi les deux autres ne le sont pas.

1.

contexte	*numéro*	*preuve*	*règle*
1	*1*	`Supposons` $\exists x\ P(x) \wedge \forall x\ Q(x)$	
1	*2*	$\exists x\ P(x)$	$\wedge$*E1 1*
1	*3*	$\forall x\ Q(x)$	$\wedge$*E2 1*
1,4	*4*	`Supposons` $P(x)$	
1,4	*5*	$Q(x)$	$\forall$*E 3, x*
1,4	*6*	$P(x) \wedge Q(x)$	$\wedge$*I 4,5*
1,4	*7*	$\exists x(P(x) \wedge Q(x))$	$\exists$*I 6, x*
1	*8*	`Donc` $P(x) \Rightarrow \exists x(P(x) \wedge Q(x))$	$\Rightarrow$ *I 4,7*
1	*9*	$\exists x(P(x) \wedge Q(x))$	$\exists$*E 2,8*
	10	`Donc` $\exists x\ P(x) \wedge \forall x\ Q(x) \Rightarrow \exists x(P(x) \wedge Q(x))$	$\Rightarrow$ *I 1,9*

2.

contexte	numéro	preuve	règle
1	1	Supposons $\exists x\, P(x) \wedge \forall x\, Q(x)$	
1	2	$\exists x\, P(x)$	$\wedge E1$ 1
1	3	$\forall x\, Q(x)$	$\wedge E2$ 1
1,4	4	Supposons $P(x)$	
1,4	5	$Q(x)$	$\forall E$ 3, x
1,4	6	$P(x) \wedge Q(x)$	$\wedge I$ 4,5
1	7	Donc $P(x) \Rightarrow P(x) \wedge Q(x)$	$\Rightarrow I$ 4,6
1	8	$P(x) \wedge Q(x)$	$\exists E$ 2,7
1	9	$\exists x(P(x) \wedge Q(x))$	$\exists I$ 8, x
	10	Donc $\exists x\, P(x) \wedge \forall x\, Q(x) \Rightarrow \exists x(P(x) \wedge Q(x))$	$\Rightarrow I$ 1,9

3.

contexte	numéro	preuve	règle
1	1	Supposons $\exists x\, P(x) \wedge \forall x\, Q(x)$	
1	2	$\exists x\, P(x)$	$\wedge E1$ 1
1	3	$\forall x\, Q(x)$	$\wedge E2$ 1
1,4	4	Supposons $P(x)$	
1	5	Donc $P(x) \Rightarrow P(x)$	$\Rightarrow I$ 4,4
1	6	$P(x)$	$\exists E$ 2,5
1	7	$Q(x)$	$\forall E$ 3, x
1	8	$P(x) \wedge Q(x)$	$\wedge I$ 6,7
1	9	$\exists x(P(x) \wedge Q(x))$	$\exists I$ 8, x
	10	Donc $\exists x\, P(x) \wedge \forall x\, Q(x) \Rightarrow \exists x(P(x) \wedge Q(x))$	$\Rightarrow I$ 1,9

□

Exercice 105 (Copie) *Prouver par déduction naturelle que $\forall x \forall y P(x,y) \Rightarrow \forall x \forall y P(y,x)$ en utilisant la règle de copie un minimum de fois.* □

Exercice 106 (Déduction naturelle) *Prouver les formules suivantes grâce à la déduction naturelle (notez que Q est une variable propositionnelle) :*

1. $\forall x(Q \wedge P(x)) \Rightarrow Q \wedge \forall x P(x)$.
2. $Q \wedge \forall x P(x) \Rightarrow \forall x(Q \wedge P(x))$.
3. $\forall x(Q \vee P(x)) \Rightarrow Q \vee \forall x P(x)$. *(**)*
4. $Q \vee \forall x P(x) \Rightarrow \forall x(Q \vee P(x))$.
5. $\exists x(Q \wedge P(x)) \Rightarrow Q \wedge \exists x P(x)$.
6. $Q \wedge \exists x P(x) \Rightarrow \exists x(Q \wedge P(x))$.
7. $\exists x(Q \vee P(x)) \Rightarrow Q \vee \exists x P(x)$.
8. $Q \vee \exists x P(x) \Rightarrow \exists x(Q \vee P(x))$. *(*)*

Dans cet exercice, la formule $P(x)$ peut être remplacée par une formule quelconque. Par contre, Q peut être remplacée seulement par une formule n'ayant pas x comme variable libre. □

Exercice 107 (Preuve) *Prouver la formule $\neg \exists x P(x) \Rightarrow \forall x \neg P(x)$. Vérifier que $P(x)$ peut être remplacée par une formule quelconque. Remarque : cette formule est la réciproque de la formule prouvée à l'exemple 6.1.6 page 118.* □

Exercice 108 (Égalité) *Prouver les formules suivantes :*

1. $R(a,c) \wedge (a = b) \Rightarrow R(b,c)$.
2. $x = y \Rightarrow f(x,z) = f(y,z)$.
3. $\forall x \exists y(x = y)$.

4. $\exists x \forall y\ x = y \Rightarrow \forall x \forall y\ x = y$. (*)

□

Exercice 109 (Égalité,)** *Prouver que la deuxième définition de « il existe un et un seul x » (voir sous-section 6.2.3 page 120) implique la première, autrement dit prouver la formule :* $\exists x P(x) \wedge \forall x \forall y (P(x) \wedge P(y) \Rightarrow x = y) \Rightarrow \exists x (P(x) \wedge \forall y (P(y) \Rightarrow x = y))$. □

Exercice 110 (Induction et déduction naturelle,*)** *Nous pouvons définir l'addition grâce aux deux formules :*
(a) $\forall n (n + 0 = n)$.
(b) $\forall n \forall p (n + s(p) = s(n + p))$.
Ces deux formules permettent de faire des additions : nous pouvons grâce à elles prouver que $s(0) + s(0) = s(s(0))$. *Mais elles ne permettent pas de prouver les propriétés générales de l'addition, qui nécessitent d'avoir le principe de récurrence.*

1. *Montrer que des hypothèses (a) et (b), nous ne pouvons pas déduire* $\forall n (0 + n = n)$.
2. *Nous donnons un nom à la propriété ci-dessus et nous posons le principe de récurrence sur cette propriété :*
   (c) $\forall n (P(n) \Leftrightarrow 0 + n = n)$.
   (d) $P(0) \wedge \forall n (P(n) \Rightarrow P(s(n))) \Rightarrow \forall n P(n)$.
   Prouver par la déduction naturelle que $(a) \wedge (b) \wedge (c) \wedge (d) \Rightarrow \forall n P(n)$.

□

Exercice 111 (Examen 2009) *Les preuves demandées ci-dessous devront être justifiées.*

1. *Donner une preuve en* déduction naturelle avec les justifications *de la validité de la formule :*
   $(\exists x\ p(x) \Rightarrow \forall x\ q(x)) \Rightarrow \forall x (p(x) \Rightarrow q(x))$.
2. *Donner une preuve en* déduction naturelle avec les justifications *de la validité de la formule :*
   $\exists x\ p(x) \wedge \forall x (p(x) \Rightarrow p(f(x))) \Rightarrow \exists x\ p(f(f(x)))$.
3. *Notons* $f^n(x)$ *le terme obtenu en appliquant n fois f à x.*
   Par exemple $f^0(x) = x, f^1(x) = f(x), f^2(x) = f(f(x))$.
   Soient Γ, Δ *deux ensembles de formules et* A, B *deux formules. On rappelle que* $\Gamma \vdash A$ *est vrai s'il y a une preuve de A dans l'*environnement Γ.
   On donne ci-dessous des propriétés triviales de la relation $\vdash$.
   Monotonie : si $\Gamma \vdash A$ *et* $\Gamma \subset \Delta$ *alors* $\Delta \vdash A$.
   Composition : si $\Gamma \vdash A$ *et* $\Gamma \vdash A \Rightarrow B$ *alors* $\Gamma \vdash B$.
   (a) *Donnez une preuve* en déduction naturelle avec les justifications *de ce que la propriété suivante est vraie :*
   $\forall x (p(x) \Rightarrow p(f(x))) \vdash \exists x p(f^n(x)) \Rightarrow \exists x\ p(f^{n+1}(x))$
   (b) *Déduire de la propriété ci-dessus, de la monotonie et de la composition que* pour tout *entier naturel n :*
   $\exists x\ p(x), \forall x (p(x) \Rightarrow p(f(x))) \vdash \exists x\ p(f^n(x))$.

□

Exercice 112 (Examen 2012) *Prouver les formules suivantes par déduction naturelle au premier ordre.*

1. $\neg \forall x P(x) \vee \neg \exists y Q(y) \Rightarrow \neg (\forall x P(x) \wedge \exists y Q(y))$.
2. $\forall x \forall y (P(y) \Rightarrow R(x)) \Rightarrow \exists y P(y) \Rightarrow \forall x R(x)$.
3. $\neg \forall x \neg P(x) \Rightarrow \exists x P(x)$.

□

Exercice 113 (Examen 2013) *Prouver les formules suivantes par déduction naturelle au premier ordre.*

1. $\exists x (Q(x) \Rightarrow P(x)) \wedge \forall x Q(x) \Rightarrow \exists x P(x)$.
2. $\forall x\ \forall y (R(x, y) \Rightarrow \neg R(y, x)) \Rightarrow \forall x \neg R(x, x)$.

□

Exercice 114 (Quelques questions posées en examen) *Démontrer les formules suivantes par déduction naturelle.*

1. $\exists x (P(x) \vee Q(x)) \wedge \forall x \neg Q(x) \Rightarrow \exists x P(x)$.
2. $\forall x (P(x) \Rightarrow Q(x)) \wedge \exists x (P(x) \wedge R(x)) \Rightarrow \exists x (Q(x) \wedge R(x))$.
3. $\exists x \neg (P(x) \vee \neg P(x)) \Rightarrow \forall x P(x)$.

□

Bibliographie

[1] Collin Allen and Michael Hand. *Logic Primer*. MIT, 2001.

[2] Peter B. Andrews. *An introduction to mathematical logic : to truth through proof.* Academic Press, 1986.

[3] Franz Baader and Wayne Snyder. Unification theory. In John Alan Robinson and Andrei Voronkov, editors, *Handbook of Automated Reasoning (in 2 volumes)*, pages 445–532. Elsevier and MIT Press, 2001.

[4] Alonzo Church. A note on the entscheidungsproblem. *Journal of Symbolic Logic*, 1(1) :40–41, 1936.

[5] Stephen A. Cook. The complexity of theorem-proving procedures. In *Proceedings of the third annual ACM symposium on Theory of computing*, STOC '71, pages 151–158, New York, NY, USA, 1971. ACM.

[6] René Cori and Daniel Lascar. *Logique mathématique Cours et exercices I. Calcul propositionnel, algèbres de Boole, calcul des prédicats*. Masson, 1993.

[7] René Cori and Daniel Lascar. *Logique mathématique Cours et exercices II. Fonctions récursives, théorème de Gödel, théorie des ensembles, théories des modèles*. Masson, 1993.

[8] René David, Karim Nour, and Christophe Raffali. *Introduction à la logique*. Dunod, 2001.

[9] Martin Davis, George W. Logemann, and Donald W. Loveland. A machine program for theorem-proving. *Communications of The ACM*, 5 :394–397, 1962.

[10] Martin Davis and Hilary Putnam. A computing procedure for quantification theory. *Journal of the ACM (JACM)*, 7(3) :201–215, 1960.

[11] Herbert B. Enderton. *A mathematical Introduction to Logic*. Academic Press, 2001.

[12] Gerhard Gentzen. Untersuchungen über das logische Schließen I. *Mathematische Zeitschrift*, 39 :176–210, 1935. 10.1007/BF01201353.

[13] Gerhard Gentzen. Untersuchungen über das logische Schließen II. *Mathematische Zeitschrift*, 39 :405–431, 1935. 10.1007/BF01201363.

[14] Roger Godement. *Cours d'algèbre*. Hermann, 1973.

[15] Gérard P. Huet. unification in typed lambda calculus. In Corrado Böhm, editor, *Lambda-Calculus and Computer Science Theory, Proceedings of the Symposium Held in Rome, March 25-27, 1975*, volume 37 of *Lecture Notes in Computer Science*, pages 192–212. Springer, 1975.

[16] Gérard P. Huet. Higher order unification 30 years later. In Victor Carreño, César Muñoz, and Sofiène Tahar, editors, *Theorem Proving in Higher Order Logics, 15th International Conference, TPHOLs 2002, Hampton, VA, USA, August 20-23, 2002, Proceedings*, volume 2410 of *Lecture Notes in Computer Science*, pages 3–12. Springer, 2002.

[17] Michael Huth and Mark Ryan. *Logic in Computer Science*. Cambridge University Press, 2004.

[18] Jacquemin. *Logique et Mathématiques 109 exercices corrigées*. Masson, 1994.

[19] Stephen C. Kleene. *Logique Mathématique*. Armand Colin, 1971.

[20] Thierry Lucas, Isabelle Berlanger, and Isabelle De Greef. *Initiation à la logique formelle*. De Boeck, 2003.

[21] Alberto Martelli and Ugo Montanari. An efficient unification algorithm. *ACM Trans. Program. Lang. Syst.*, 4 :258–282, April 1982.

[22] William McCune. Prover9 and mace4. `http://www.cs.unm.edu/~mccune/prover9/`, 2005–2010.

[23] J. Alan Robinson. A machine-oriented logic based on the resolution principle. *Journal of the ACM (JACM)*, 12(1) :23–41, January 1965.

[24] Alan Turing. On computable numbers, with an application to the entscheidungsproblem. *Proceedings of the London Mathematical Society*, 42 :230–265, 1937.

Index