Acquisition de connaissances et raisonnement en logique propositionnelle

Bruno Zanuttini

Université de Caen Basse-Normandie

U.F.R. Sciences **École doctorale SIMEM**
Département d'Informatique

Acquisition de connaissances et raisonnement en logique propositionnelle

THÈSE

présentée et soutenue publiquement le 4 juillet 2003

pour l'obtention du

Doctorat de l'Université de Caen
(arrêté du 25 avril 2002)

Spécialité Informatique

par

Bruno Zanuttini

Composition du jury

Miki Hermann	Chargé de Recherche CNRS, Ecole Polytechnique (rapporteur)
Pierre Marquis	Professeur, Université d'Artois (rapporteur)
Victor Vianu	Professeur, Université de Californie, San Diego (rapporteur)
Michel Chein	Professeur, Université de Montpellier II
Etienne Grandjean	Professeur, Université de Caen (co-directeur)
Jean-Jacques Hébrard	Maître de Conférences, Université de Caen (co-directeur)

Groupe de Recherche en Informatique, Image, Automatique et Instrumentation de Caen
CNRS UMR 6072

Mis en page avec la classe thloria.

Remerciements

Les résultats présentés dans ce mémoire sont le fruit de trois années de recherches effectuées sous la direction de Jean-Jacques Hébrard. Je tiens à le saluer bien bas, et à le remercier chaleureusement, pour son encadrement exemplaire, pour la liberté et la confiance qu'il m'a accordées, pour ses encouragements et pour son aide précieuse, tout particulièrement à la rédaction de mes travaux. Je tiens aussi à préciser que plusieurs résultats présentés dans ce mémoire sont réellement le fruit d'un travail commun avec lui. Merci également à Etienne Grandjean pour avoir accepté de me co-encadrer, pour avoir suivi mes travaux et pour son investissement tout particulier dans les dernières étapes de cette thèse.

Je suis très reconnaissant également à ceux qui m'ont ouvert de nouveaux horizons, en me faisant confiance et sans rien demander en retour. Je pense en particulier à Nadia Creignou, Bruno Crémilleux, Arnaud Durand, Nadine Lucas, Pierre Marquis et Michèle Sebag. Pour les mêmes raisons, merci à Miki Hermann, Victor Vianu et Pierre Marquis d'avoir accepté de disséquer ce mémoire, et de l'avoir fait avec une rare minutie ; leurs nombreuses et pertinentes remarques auront largement contribué à améliorer la première version. Merci encore à Michel Chein d'avoir accepté de faire partie du jury. Merci enfin à Laurent Doublier, mon prof' de maths de lycée, qui a su me donner le goût de sa matière et l'envie d'entrer à l'Université.

Mais ce travail a aussi bénéficié de l'excellente ambiance au sein du GREYC, et tout particulièrement parmi les doctorants de l'Equipe Algo. Je pense qu'Aline, Jérémie, Ben, Philippe, Régis, Guillaume, Bertrand, ainsi qu'Ali, Marc, Samir et Hosam y sont pour beaucoup. Merci tout particulièrement à Aline et à Jérémie d'avoir patiemment relu des passages de ce mémoire, à Régis pour ses conseils éclairés et patients concernant la programmation, et à Aline encore pour toutes les années côte à côte à l'Université. Et puis, toujours au GREYC, merci à Virginie, Agnès, Françoise et Béatrice pour leur disponibilité et leur sourire.

Enfin, même si presque tous écorchent régulièrement le mot «algorithme» et trouvent un peu vain de remplir des monceaux de pages blanches «de 0 et de 1», je tiens à remercier ceux qui m'ont le plus apporté, et de loin, durant cette période, ainsi qu'auparavant et sans nul doute après : tout particulièrement Dawa, Geof, Nico, Momo, Mika, Davü, Paco, Caro, Courge, Choupi, Cécile, Riké, Régis, Céline, Fred, Sandrine. Mais aussi, merci à mes parents, pour m'avoir toujours laissé choisir mes orientations, pour avoir toujours tout fait pour que je puisse suivre des études, et pour m'avoir, tout simplement, appris à être heureux. Et j'ai gardé le meilleur pour la fin : merci à Gaëlle, pour m'avoir toujours encouragé et rassuré, pour s'être intéressée à ce que je faisais, pour m'avoir supporté pendant la rédaction, pour m'avoir relu patiemment, et surtout pour me rendre heureux.

Bonne lecture...

Table des matières

Introduction générale **1**

Préliminaires généraux **7**

1 Notions de logique propositionnelle et notations 7

1.1 Formules et affectations 7

1.2 Relations et propriétés sémantiques 9

1.3 Formes normales 10

1.4 Opérations et notions syntaxiques 11

2 Rappels de complexité 14

2.1 Problèmes et complexité 14

2.2 Problèmes de décision 15

2.3 Problèmes avec sortie 16

2.4 Représentation des objets 16

3 Représentation de connaissances en logique propositionnelle 18

3.1 Ensembles d'observations et bases de connaissances 18

3.2 Un problème de carrefour 20

3.3 A propos de thèses... 21

Partie I Classes de formules **23**

Chapitre 1
Introduction

1.1 Quelle forme normale pour les formules propositionnelles ? 25

1.2 Classes de formules 27

1.3 Classes étudiées dans le mémoire 30

Chapitre 2
Bonnes formules FNC

2.1 Formules FNC de Horn 31
2.2 Formules FNC de Horn définies positives 34
2.3 Formules FNC négatives 36
2.4 Formules 2FNC 38
2.5 Classes duales 40

Chapitre 3
Formules affines

3.1 Définition et exemples 41
3.2 Propriétés algorithmiques 43
3.3 Fonctions affines et espaces vectoriels 45
3.4 Propriétés sémantiques 48
3.5 Les formules affines pour la représentation de connaissances 50

Chapitre 4
Autres classes de formules

4.1 Renommages 53
4.1.1 Définitions 53
4.1.2 Renommages et satisfaisabilité 55
4.2 Classes de formules $\mathcal{C}$-renommables 56
4.2.1 Formules FNC Horn-renommables 56
4.2.2 Formules Horn définies positives renommables 57
4.2.3 Formules monotones 58
4.3 Formules FND 59

Chapitre 5
Les classes dans leur globalité

5.1 Classes de Schaefer 63
5.1.1 Dichotomie du problème SAT-généralisé 65
5.1.2 Dichotomie du problème Inverse-SAT 66
5.2 Intersections des classes 67
5.3 Résumé des propriétés des classes 69

Partie II Acquisition de connaissances 71

Chapitre 6
Introduction

6.1 Présentation et motivations 73
6.2 Problématiques étudiées 74

Chapitre 7
Description et identification

7.1 Présentation et travaux précédents 77
7.1.1 Formalisation des problématiques 77
7.1.2 Motivations 79
7.1.3 Travaux précédents 79
7.2 Description en FNC 81
7.3 Identification 86
7.3.1 Description en FNC première 86
7.3.2 Identification avec les impliqués premiers 89
7.4 Description et identification optimisées 91
7.4.1 Description en FNC négative ou monotone 92
7.4.2 Description en FNC de Horn 93
7.5 Autres classes de formules et résumé des résultats connus 98

Chapitre 8
Approximation

8.1 Présentation et travaux précédents 101
8.1.1 Formalisation des notions d'approximation 102
8.1.2 Motivations 103
8.1.3 D'autres approches pour approximer 106
8.1.4 Travaux précédents 106
8.2 Approximations faibles 108
8.2.1 Caractérisations 108
8.2.2 Calcul d'une approximation faible 110
8.3 Approximations fortes 114
8.3.1 Calcul d'une approximation forte maximale quelconque 116
8.3.2 Calcul d'une approximation forte avec le nombre maximal de modèles 120
8.4 Résumé des résultats connus 124

Chapitre 9
PAC-apprentissage

9.1 Présentation et travaux précédents 125
9.1.1 Formalisation du problème 125

9.1.2 Motivations et remarques 127
9.1.3 Travaux précédents 129
9.2 PAC-apprenabilité de la classe des formules affines 130
9.2.1 Algorithme de mise à jour 130
9.2.2 Complexité 132
9.3 Résumé des résultats connus 134

Partie III Raisonnement **135**

Chapitre 10
Introduction

10.1 Présentation et motivations 137
10.2 Problématiques étudiées 138
10.3 Déduction, point de vue «exact» 140

Chapitre 11
Abduction

11.1 Présentation et travaux précédents 143
11.1.1 Formalisation des notions 144
11.1.2 Motivations 146
11.1.3 Travaux précédents 146
11.2 Modèle général d'algorithme 148
11.2.1 Projection 148
11.2.2 Modèle d'algorithme 150
11.3 Classes polynomiales 152
11.3.1 Formules FND 153
11.3.2 Formules affines 157
11.3.3 Classes retrouvées 158
11.4 Discussion et résumé des résultats connus 159

Chapitre 12
Raisonnement et approximation

12.1 Présentation 163
12.2 Déduction et abduction avec des approximations fortes 165
12.3 Abduction 167
12.3.1 Vision «méfiante» 167
12.3.2 Vision «restrictive» avec des approximations faibles minimales 167

12.4 Discussion et résumé des résultats connus 170

Conclusion générale **173**

Bibliographie **179**

Index **185**

Introduction générale

L'Intelligence Artificielle, champ de recherche dont on peut situer les origines dans les années 1950, vise à permettre à un système informatique d'effectuer des tâches intelligentes, ou plutôt *réputées* intelligentes : élaborer un plan pour arriver à un certain but, s'exprimer en langue naturelle, reconnaître un visage sur une image, jouer aux échecs, prouver des théorèmes, etc. Les problèmes à résoudre pour permettre à un ordinateur de réaliser de telles tâches sont nombreux. Nous en traitons deux grands groupes : les problèmes d'*acquisition de connaissances* et les problèmes de *raisonnement*.

Sujet et motivations

Il est naturel, lorsque l'on cherche à concevoir des machines ou des programmes capables d'effectuer des tâches réputées intelligentes, d'essayer de leur donner les moyens de gérer des *connaissances*. Un mécanisme de gestion de connaissances peut en effet permettre à une machine de s'adapter à son environnement en lui donnant les moyens d'acquérir et de mémoriser des *informations* à son propos ; il peut lui permettre d'apprendre de nouveaux concepts, de les décrire, ou encore de «comprendre» comment fonctionne l'univers dans lequel il évolue. La machine en question pourra alors utiliser ces connaissances pour raisonner, pour répondre à des questions.

On peut raisonnablement attendre, par exemple, d'un robot destiné à explorer Mars de façon autonome qu'il ne tombe pas deux fois dans le même ravin, et ce même si ce dernier semble avoir été placé là exprès pour le piéger. On peut également attendre qu'il retienne la manière dont il s'en est sorti, afin de ne pas essayer à nouveau toute sa panoplie d'outils s'il se retrouve dans la même situation. S'il rencontre, contre toute attente, un groupe d'êtres vivants qu'il estime faire partie d'une même espèce, on peut encore espérer qu'à son retour sur Terre, il sera capable de raconter son aventure en expliquant : «Ils sont verts et mesurent entre quatre-vingt-douze et cent vingt-trois centimètres», plutôt qu'en ramenant des images montrant indifféremment un coucher de soleil, un de ces petits végétaux verts et un ravin.

On attend donc d'un tel robot qu'il soit capable à la fois d'*acquérir* et d'*utiliser* des connaissances ; acquérir, par exemple, la connaissance des caractéristiques du ravin, ou encore le concept de «végétal martien» ; utiliser ces connaissances, par exemple, pour ne pas tomber à nouveau dans le même ravin, ou pour retrouver l'outil qui s'est montré le plus utile la dernière fois qu'il y est tombé. Notons qu'une telle machine possède également des connaissances qu'on pourrait qualifier d'«innées» : le programme qui lui permet de se déplacer, le but de sa mission si elle n'a été conçue que pour l'accomplir, etc. Ces connaissances sont cependant d'un autre type, en ce sens qu'elles ont été *programmées*, et non *acquises*. Elles sont néanmoins susceptibles d'être utilisées au même titre que les autres.

Mais on ne peut parler de connaissances sans parler de leur *représentation* : le robot se souvient-il du ravin en en conservant une image, se souvient-il de sa profondeur et de ses coor-

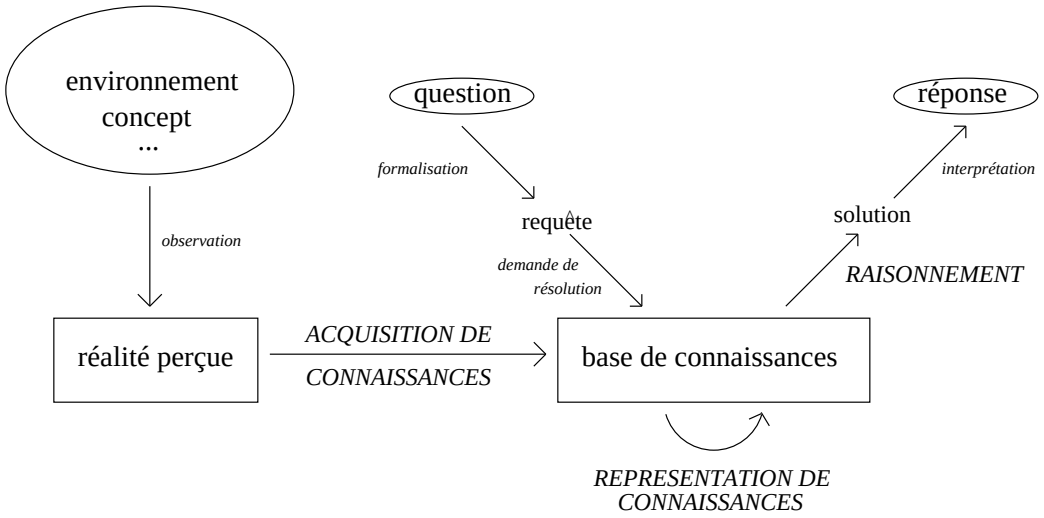

FIG. 1 – Les grandes problématiques étudiées dans ce mémoire (en majuscules)

données géographiques ou simplement de son existence ? Représente-t-il le concept de «végétal martien» comme un nouveau concept, le décrit-il par analogie aux végétaux terrestres, peut-il exprimer des généralités à son sujet ou doit-il s'en souvenir individu par individu ? Les informations qu'il transmet à son propos sont-elles compréhensibles, ou ne sont-elles que suites de chiffres ?

Nous nous intéressons donc dans ce mémoire à ces trois problématiques : acquisition, utilisation et représentation de connaissances. Nous considérerons toujours un système «intelligent», comme notre robot, muni d'une certaine *base de connaissances*, et nous nous demanderons comment il peut enrichir cette base en observant son univers, ou à partir d'exemples d'un concept (les végétaux qu'il observe, par exemple), et comment il peut l'utiliser pour déduire d'autres connaissances, ou encore pour expliquer des faits observés.

Ces processus sont replacés dans un contexte plus général sur la figure 1. Bien entendu, leur étude n'est absolument pas nouvelle, et ils sont au coeur de la plupart des ouvrages traitant de l'Intelligence Artificielle en général (voir par exemple [RN95, RK91, UPS01]). Leur étude conjointe est tout particulièrement motivée par le cadre général proposé par Khardon et Roth [KR94, KR97], où il s'agit d'*apprendre à raisonner* ; nous reviendrons sur cette approche tout au long de ce mémoire.

Pour ce qui est du langage de représentation de ces connaissances, nous considérons dans ce mémoire le langage de la *logique propositionnelle*, c'est-à-dire l'ensemble des formules construites à partir de *descripteurs*, tels que «L'objet en question est profond», «Cet exemple du concept en question est un feuillu» ou encore «Je suis tombé dans un ravin», et de *connecteurs*, tels que «et», «non» ou encore «si et seulement si». Ce langage, bien entendu, est loin d'être le plus expressif des langages de représentation de connaissances. Parmi les concepts dont il ne peut exprimer toute la subtilité, on trouve par exemple la phrase «Tous les ravins sont des pièges», qui n'est réellement accessible à la logique propositionnelle que s'il y a un nombre fini de ravins envisageables et que le robot les connaît tous ; dans le cas contraire, il faudrait par exemple utiliser la logique du *premier ordre*, la logique propositionnelle correspondant à l'ordre 0. Il existe également de nombreux autres langages de représentation, non nécessairement basés

sur la logique : les «frames», par exemple, ou encore la langue naturelle, etc. ; nous renvoyons le lecteur à l'ouvrage de Rich et Knight [RK91, partie II] ou à celui de Bertino *et al.* [BCZ01, partie 3] pour plus de détails.

Nous choisissons néanmoins la logique propositionnelle car, même en se restreignant à ce langage, il reste de nombreux problèmes qu'on ne sait pas traiter ; or, plus le langage de représentation de connaissances choisi est riche, plus les problèmes deviennent difficiles, voire impossibles à résoudre ; ainsi, on ne peut pas écrire un programme qui, pour toute assertion de la logique du premier ordre, décide si cette assertion est vraie ou fausse en un temps fini ; on le peut, en revanche, pour la logique propositionnelle. Nous choisissons donc d'étudier un langage peu expressif mais pour lequel on peut espérer écrire des programmes qui réalisent des tâches naturelles. L'étude de la logique propositionnelle pour la représentation de connaissances et le raisonnement s'inscrit également dans une longue lignée de travaux commencés dans les années 1990, parmi lesquels les travaux de Selman, Kautz [KKS95, SK96], Khardon, Roth [KR96], Gottlob, Eiter [EG95a], Ibaraki, Makino [EIM98a, IKM99, MHI99, EM02], del Val [dV95, dV96, dV00b, dV00a] ou encore Marquis [Mar00, DM02].

Nous nous intéressons principalement dans ce mémoire aux aspects *algorithmiques* des processus. Ce point de vue s'intéresse à des problèmes, comme par exemple la déduction, et aux algorithmes qui les résolvent, c'est-à-dire à des programmes abstraits. Nous nous intéressons également, bien sûr, à la rapidité de ces programmes, afin de déterminer s'ils sont utilisables en pratique ; c'est le but des études de *complexité en temps*, qui s'intéressent à une notion de rapidité des algorithmes indépendante de la rapidité de calcul des ordinateurs.

Point de vue et objectifs précis

Comme nous l'avons dit, nous étudions principalement la *complexité en temps* des problèmes. Nous cherchons des algorithmes efficaces pour résoudre les problèmes qui nous intéressent, ou essayons de montrer que le problème en question est *difficile*, c'est-à-dire certainement impossible à résoudre efficacement.

Nous traitons principalement six familles de problèmes avec ce point de vue : parmi les problèmes d'*acquisition* de connaissances, tous à partir d'observations ou d'exemples, nous traitons l'acquisition exacte, l'acquisition approchée et l'apprentissage ; parmi les problèmes d'*utilisation* de connaissances (raisonnement), nous traitons la vérification de cohérence, la déduction et l'abduction. Mais dans le cas général de la logique propositionnelle, c'est-à-dire en s'autorisant toutes les formules de cette logique, la plupart de ces problèmes sont difficiles. De fait, nous nous intéressons à des *restrictions* de ce langage, sous la forme de *classes de formules* dans lesquelles nous supposons représentées les bases de connaissances. Nous étudions (partie I du mémoire) les classes intéressantes et justifions nos choix ; ceux-ci sont notamment guidés par la possibilité de traiter efficacement les problèmes d'acquisition et de raisonnement correspondants, et définissent en retour autant de problèmes distincts pour chaque processus, selon la classe dans laquelle on suppose les bases de connaissances représentées.

Une partie de ces problèmes ont déjà été étudiés dans la littérature. Nous n'introduisons ni nouvelle classe de formules ni nouvelle problématique ; néanmoins, nous considérons certaines classes et certains problèmes sous un jour nouveau, dans un contexte différent des autres travaux. Notre but est donc de compléter et d'améliorer les résultats déjà connus, afin de constituer une sorte de «catalogue» de ce qui est possible et/ou intéressant lorsqu'on utilise la logique propositionnelle dans un contexte d'Intelligence Artificielle. En ce sens, on peut voir ce travail comme assez similaire en esprit au travail de Darwiche et Marquis sur la compilation de

connaissances [DM02].

Résultats

Les principales contributions de cette thèse sont les suivantes. Tout d'abord (partie I), nous synthétisons les définitions et propriétés des classes de formules classiques de la logique propositionnelle : notamment les classes des formules en Forme Normale Conjonctive de Horn, Horn définies positives, bijonctives, Horn-renommables, monotones, la classe des formules affines et les classes de formules duales, en Forme Normale Disjonctive. Nous discutons, dans le contexte des problèmes d'acquisition de connaissances et de raisonnement, des critères qui nous amènent à considérer ces classes comme les seules réellement intéressantes.

Ensuite (partie II), nous nous intéressons aux problèmes d'acquisition de connaissances à partir d'exemples dans ces classes de formules. Nous donnons, tout d'abord (chapitre 7), un algorithme général et efficace pour les problèmes de description et d'identification [DP92, AFP92] relatifs à ces classes [ZH02]. Puis, nous donnons des optimisations de cet algorithme pour certaines de ces classes, parmi lesquelles la classe des formules de Horn, pour laquelle nous donnons un algorithme meilleur que tous ceux connus auparavant [HZ03]. Nous traitons ensuite (chapitre 8) le calcul d'approximations faibles et fortes d'ensembles d'exemples [SK96], et donnons en particulier un algorithme glouton efficace pour le calcul d'approximations fortes maximales dans toutes les classes de formules qui nous intéressent, et des algorithmes efficaces pour le calcul d'approximations affines (résultats publiés partiellement [Zan02a, Zan02b]). Enfin, nous traitons (chapitre 9) le problème du PAC-apprentissage [Val84, Ang92], et donnons un algorithme efficace pour la classe des formules affines [Zan02a].

Puis, nous étudions (partie III) trois grandes problématiques de raisonnement, toujours pour les classes de formules présentées dans la partie I. Tout d'abord, nous synthétisons (chapitre 10) les résultats connus concernant la vérification de cohérence et la déduction. Ensuite, nous donnons (chapitre 11) un modèle d'algorithme très général pour l'abduction [EG95a], et montrons qu'il permet d'obtenir des algorithmes efficaces pour les classes de formules en Forme Disjonctive Normale qui nous intéressent ainsi que pour la classe des formules affines [Zan02c, Zan03b]. Enfin, nous étudions (chapitre 12) les problèmes de vérification de cohérence, de déduction et d'abduction lorsque la base de connaissances utilisée ne représente qu'une approximation de la réalité qu'elle est censée décrire, dans la même acception que dans le chapitre 8 ; ce chapitre ne s'intéresse donc plus à l'*algorithmique* de ces processus, mais à leur *signification* dans ce cadre. A nouveau nous synthétisons les résultats connus concernant la vérification de cohérence et la déduction, puis nous posons de premières pierres pour l'étude de l'abduction dans ce cadre, cette problématique n'étant pas réellement étudiée dans la littérature [Zan02b, Zan02c].

Enfin, une contribution de ce travail, transversale aux autres, est l'étude des formules affines pour la représentation de connaissances. Cette classe de formules était en effet déjà connue depuis longtemps, mais n'avait jamais réellement été considérée dans le contexte des problématiques d'Intelligence Artificielle. Nous l'étudions donc au même titre que les autres classes, plus souvent considérées dans ce cadre, et montrons qu'elles possèdent de nombreuses propriétés intéressantes [Zan02a, Zan03a].

Organisation du mémoire

Le mémoire est organisé en trois grandes parties. La première présente les classes de formules auxquelles nous nous intéressons et motive nos choix. Le chapitre 1 introduit et motive les critères

que nous jugeons importants, les chapitres 2, 3 et 4 présentent les différentes classes de formules, et le chapitre 5 les considère dans leur globalité.

La deuxième partie, quant à elle, est consacrée à l'acquisition de connaissances. Le chapitre 6 introduit informellement et motive les problématiques. Le chapitre 7 traite de l'acquisition *exacte* de connaissances, le chapitre 8 de l'acquisition *approximative*, et le chapitre 9 de l'acquisition *par apprentissage*.

Enfin, la troisième partie traite des processus de raisonnement. Le chapitre 10 présente les problématiques, et traite rapidement les problèmes de vérification de cohérence et de déduction avec des bases de connaissances *exactes* ; ces problèmes admettent en effet des résultats très généraux, déjà intensivement étudiés dans la littérature. Puis le chapitre 11 traite l'*abduction* avec une base de connaissances exacte, et enfin le chapitre 12 les problématiques de vérification de cohérence, de déduction et d'abduction avec une base de connaissances *approximative*.

Nous terminons par une conclusion générale, où nous résumons les résultats obtenus, les discutons ainsi que les problèmes encore ouverts, et identifions des perspectives intéressantes pour prolonger ce travail.

Chaque partie débute par une introduction informelle où sont présentés et motivés les objectifs des chapitres suivants. Chaque chapitre des parties II et III propose une présentation détaillée et formelle des problèmes qu'il traite ainsi qu'un rappel des travaux précédents sur le sujet, et se termine par un résumé des résultats connus pour les classes de formules qui nous intéressent. Dans ces résumés, nous indiquons les références bibliographiques correspondantes, ou des références pertinentes lorsque la paternité d'un résultat est difficile à attribuer ; nous distinguons les apports de cette thèse d'une étoile (*). Pour ce qui est des outils techniques, ils sont rassemblés dans les préliminaires généraux (pages 7 et suivantes) lorsqu'ils servent dans plusieurs chapitres, les autres passages techniques étant encapsulés dans les définitions et les preuves des propositions afin de permettre différents niveaux de lecture. D'autre part, nous avons choisi de donner de nouvelles preuves de résultats connus, ou de rappeler comment on peut en montrer certains, lorsque cela peut en permettre une compréhension plus profonde ; dans tous les cas, nous citons les références bibliographiques où peuvent être trouvées les preuves originales. Enfin, toujours afin d'alléger et de faciliter la lecture, deux exemples jouets, présentés dans les préliminaires généraux, nous suivront tout au long du mémoire pour illustrer les concepts et les algorithmes.

Préliminaires généraux

Nous présentons dans ce chapitre les notions techniques de logique propositionnelle et de complexité utiles pour le reste du mémoire ; toutes ces notions sont relativement élémentaires, et volontairement présentées en détail. Les classes de formules qui nous intéressent seront présentées dans la partie I, et les problématiques précises seront définies au fil des chapitres. Pour terminer, nous présentons, au travers notamment de deux exemples jouets, la manière dont nous concevons la représentation de connaissances en logique propositionnelle ; nous utiliserons ensuite ces exemples de façon récurrente dans le mémoire, pour illustrer les concepts et les algorithmes.

1 Notions de logique propositionnelle et notations

Nos objets d'étude étant des connaissances représentées en logique propositionnelle, nous présentons tout d'abord les notions de formules et d'affectations, ainsi que les relations et opérations que nous utiliserons sur ces objets. Toutes ces notions sont classiques et simples, et nous renvoyons le lecteur à l'ouvrage d'Enderton [End72] (où la logique propositionnelle est appelée «sentential logic») ou encore à celui de Russell et Norvig [RN95, chapitre 6] pour plus de détails.

1.1 Formules et affectations

Formules Le langage de la logique propositionnelle sur un ensemble V, fini, de variables booléennes est l'ensemble des formules bien formées sur un sous-ensemble de V, le connecteur unaire $\neg$ (négation), les connecteurs binaires $\rightarrow$ (implication) et $\leftrightarrow$ (équivalence logique) et les connecteurs d'arité quelconque (éventuellement 0) $\wedge$ (conjonction), $\vee$ (disjonction) et $\oplus$ (ou exclusif). Pour désigner les formules, nous utilisons la lettre ϕ, éventuellement primée ou indicée. Par exemple, la formule :

$$\phi_1 = ((x_1 \rightarrow x_2) \oplus (\neg(x_2 \vee \neg x_4 \vee x_5) \wedge (x_1 \leftrightarrow x_4))) \leftrightarrow (\neg x_2 \rightarrow (\neg x_1 \oplus x_4 \oplus x_5))$$

est une formule de la logique propositionnelle sur l'ensemble de variables $V_1 = \{x_1, x_2, x_4, x_5\}$ (mais également sur l'ensemble $V_1' = \{x_1, x_2, x_3, x_4, x_5\}$, etc.). Afin de simplifier les définitions, nous n'autorisons pas les constantes 0 et 1 à apparaître dans les formules, mais cette convention est sans perte de généralité puisqu'on peut les représenter par, respectivement, la disjonction et la conjonction vides (voir le paragraphe «Modèles» page 8).

Dans tout le mémoire, les ensembles de variables que nous considérons sont des sous-ensembles finis de l'ensemble dénombrable $\mathbb{V} = \{x_i \mid i \in \mathbb{N}\}$. Tous ces ensembles sont donc naturellement totalement ordonnés par la relation $x_i < x_j \iff i < j$. Nous utilisons la lettre x, éventuellement indicée ou primée, pour désigner les variables booléennes, et la lettre V pour les ensembles de telles variables. Si ϕ est une formule propositionnelle, l'ensemble des variables apparaissant dans ϕ est noté $Var(\phi)$. Ainsi, on a $Var(\phi_1) = V_1$ pour la formule ci-dessus. Pour un ensemble E, fini, d'objets quelconques, nous notons $|E|$ le nombre d'éléments dans E, et on a ainsi $|V_1| = 4$.

Affectations Une *affectation* m à un ensemble fini de variables V est une application de V dans $\{0,1\}$, où 0 code intuitivement le Faux et 1 le Vrai ; nous notons $m[x]$ la valeur affectée par m à une variable $x \in V$. Par exemple, l'application m_1 définie par :

$$m_1[x_1] = 1, m_1[x_2] = 0, m_1[x_4] = 1, m_1[x_5] = 0$$

est une affectation à l'ensemble de variables V_1 ci-dessus. Etant donné un ensemble de variables V, nous notons $\{0,1\}^V$ l'ensemble des affectations à V, et nous en distinguons deux éléments particuliers : l'affectation de 0 (resp. 1) à tous les éléments de V ; nous les notons $\overline{0_V}$ et $\overline{1_V}$, respectivement. Enfin, si V est un ensemble de variables, m une affectation à V et V' un sur-ensemble de V, nous appelons *prolongement* de m sur V' toute affectation m' à V' vérifiant $\forall x \in V, m'[x] = m[x]$; par abus de langage, nous considérons également une affectation m' à V' comme une affectation à V, en restreignant implicitement son domaine de définition à V. Par exemple, l'affectation m'_1 à V'_1 définie par :

$$m'_1[x_1] = 1, m'_1[x_2] = 0, m'_1[x_3] = 0, m'_1[x_4] = 1, m'_1[x_5] = 0$$

est un prolongement de m_1 sur V'_1, et est également considérée comme une affectation à V_1.

Remarquons qu'on définit habituellement une affectation comme une application dans $\{0,1\}$, définie sur l'ensemble de *toutes* les variables du langage propositionnel considéré ; notre définition comme une application définie sur un ensemble de variables donné a pour but de simplifier la présentation des notions et algorithmes dans le reste du mémoire.

Modèles Etant donnée une formule propositionnelle ϕ, une affectation m à $Var(\phi)$ est appelée un *modèle* de ϕ si elle *satisfait* ϕ ; on note alors $m \models \phi$. La satisfaction d'une formule ϕ est définie inductivement par les règles suivantes, où x est une variable quelconque, $\phi_1, \phi_2, \ldots, \phi_n$ $(n \geq 0)$ sont des formules et m est une affectation à $Var(\phi)$:

- si $\phi = x$, alors $m \models \phi \iff m[x] = 1$
- si $\phi = \neg\phi_1$, alors $m \models \phi \iff m \not\models \phi_1$
- si $\phi = \phi_1 \rightarrow \phi_2$, alors $m \models \phi \iff m \not\models \phi_1$ ou $m \models \phi_2$
- si $\phi = \phi_1 \leftrightarrow \phi_2$, alors $m \models \phi \iff (m \models \phi_1$ et $m \models \phi_2)$ ou $(m \not\models \phi_1$ et $m \not\models \phi_2)$
- si $\phi = \phi_1 \wedge \phi_2 \wedge \cdots \wedge \phi_n$, alors $m \models \phi \iff \forall i \in \{1,2,\ldots,n\}, m \models \phi_i$
- si $\phi = \phi_1 \vee \phi_2 \vee \cdots \vee \phi_n$, alors $m \models \phi \iff \exists i \in \{1,2,\ldots,n\}, m \models \phi_i$
- si $\phi = \phi_1 \oplus \phi_2 \oplus \cdots \oplus \phi_n$, alors $m \models \phi \iff |\{i \in \{1,2,\ldots,n\} \mid m \models \phi_i\}|$ est impair.

Remarquons en particulier que la disjonction et le ou exclusif d'aucune variable ne sont jamais satisfaits, tandis que la conjonction d'aucune variable l'est toujours. Remarquons également que le connecteur $\vee$ est distributif sur le connecteur $\wedge$, et inversement ; de fait, pour ϕ_1, ϕ_2, ϕ_3 trois formules et m une affectation à $Var(\phi_1) \cup Var(\phi_2) \cup Var(\phi_3)$, m satisfait la formule $\phi_1 \vee (\phi_2 \wedge \phi_3)$ si et seulement s'il satisfait la formule $(\phi_1 \vee \phi_2) \wedge (\phi_1 \vee \phi_3)$, et de même en inversant les rôles de la disjonction et de la conjonction.

Pour ϕ une formule, nous notons $\mathcal{M}(\phi) \subseteq \{0,1\}^{Var(\phi)}$ l'ensemble des modèles de ϕ ; nous disons que ϕ *décrit* $\mathcal{M}(\phi)$, ou encore que $\mathcal{M}(\phi)$ est *descriptible* par ϕ. Plus généralement, si M est un ensemble d'affectations à un même ensemble de variables V, et si ϕ est une formule, on dit que ϕ *décrit* M, ou *est une description* de M, si M est exactement l'ensemble des affectations à V qui satisfont ϕ.

Notons que pour deux formules ϕ et ϕ' sur le même ensemble de variables, on a :

$$\mathcal{M}(\phi \wedge \phi') = \mathcal{M}(\phi) \cap \mathcal{M}(\phi'), \quad \mathcal{M}(\phi \vee \phi') = \mathcal{M}(\phi) \cup \mathcal{M}(\phi'), \quad \mathcal{M}(\phi \oplus \phi') = \mathcal{M}(\phi) \Delta \mathcal{M}(\phi')$$

où Δ représente la différence symétrique. L'ensemble $\mathcal{M}(\neg\phi)$, quant à lui, est le complémentaire de $\mathcal{M}(\phi)$ dans l'ensemble $\{0,1\}^{Var(\phi)}$.

Nous utilisons toujours la lettre m (resp. M) pour désigner des affectations (resp. des ensembles d'affectations), et lorsque l'ensemble de variables $V = \{x_{i_1}, x_{i_2}, \ldots, x_{i_{|V|}}\}$, $i_1 < i_2 < \cdots < i_{|V|}$, est clairement donné par le contexte, nous représentons une affectation m de façon compacte, comme un mot de longueur V sur l'alphabet $\{0,1\}$ dont la $j^{\text{ème}}$ lettre est la valeur $m[x_{i_j}]$. Ainsi, l'affectation m_1 à V_1 est notée 1010, et l'ensemble des modèles de la formule ϕ_1 est l'ensemble :

$$\mathcal{M}(\phi_1) = \{0,1\}^{V_1} \backslash \{0001, 0010, 1001\}$$

Lorsque nous parlons d'un *ensemble d'affectations* M, nous sous-entendons toujours que tous les éléments de M sont des affectations au *même* ensemble de variables.

Affectations partielles Enfin, nous utilisons dans le mémoire la notion d'*affectation partielle* à un ensemble de variables V. Nous introduisons le symbole '?', qui signifie intuitivement «0 ou 1», et nous appelons *affectation partielle à* V une application p de V dans $\{0,1,?\}$; nous utilisons les mêmes (abus de) notations que pour les affectations, et nous introduisons la notion d'*extension*.

Pour V un ensemble de variables et p une affectation partielle à V, nous appelons *extension de* p, et notons $ext(p)$, l'ensemble d'affectations à V :

$$\{m \in \{0,1\}^V \mid \forall x \in V, p[x] \neq ? \Rightarrow m[x] = p[x]\}$$

Similairement, si P est un *ensemble* d'affectations partielles à V, nous appelons *extension de* P, et notons $ext(P)$, l'ensemble d'affectations à V $\bigcup_{p \in P} ext(p)$. Ainsi, l'affectation p_1 à V_1 définie par :

$$p_1[x_1] =?, p_1[x_2] = 0, p_1[x_4] = 1, p_1[x_5] =?$$

est une affectation partielle à V_1. Nous la notons également ?01?, et son extension est l'ensemble d'affectations à V_1 :

$$ext(p_1) = \{0010, 0011, 1010, 1011\}$$

1.2 Relations et propriétés sémantiques

Nous qualifions de *sémantique* une notion définie, soit pour des ensembles d'affectations, soit pour des formules, mais dans ce dernier cas qui ne dépendent que de leurs ensembles de modèles ; notons en effet que l'application $\phi \mapsto \mathcal{M}(\phi)$ associe de façon canonique un ensemble d'affectations à $Var(\phi)$ à une formule ϕ.

Une notion sémantique essentielle est la notion d'*équivalence logique* : deux formules ϕ et ϕ' sont dites *logiquement équivalentes*, et on note $\phi \equiv \phi'$, si les ensembles des prolongements de $\mathcal{M}(\phi)$ et de $\mathcal{M}(\phi')$ sur $Var(\phi) \cup Var(\phi')$ sont égaux. Notons que dans le cas où $Var(\phi) = Var(\phi')$, on a $\phi \equiv \phi'$ si et seulement si $\mathcal{M}(\phi) = \mathcal{M}(\phi')$. Notons également que ϕ et ϕ' peuvent être logiquement équivalentes même si les ensembles de variables $Var(\phi)$ et $Var(\phi')$ sont différents. Ainsi, la formule ϕ_1 sur $V_1 = \{x_1, x_2, x_4, x_5\}$ est logiquement équivalente à la formule :

$$\phi_1' = (x_1 \vee x_2 \vee \neg x_4 \vee x_5) \wedge (x_2 \vee x_3 \vee x_4 \vee \neg x_5) \wedge (x_2 \vee \neg x_3 \vee x_4 \vee \neg x_5)$$

puisque les prolongements de $\mathcal{M}(\phi_1)$ et $\mathcal{M}(\phi_1')$ sur $V_1' = \{x_1, x_2, x_3, x_4, x_5\}$ sont égaux.

Similairement, on dit que ϕ *implique* ϕ', et on note $\phi \models \phi'$, si l'ensemble des prolongements de $\mathcal{M}(\phi)$ sur $Var(\phi) \cup Var(\phi')$ est inclus dans l'ensemble des prolongements de $\mathcal{M}(\phi')$ sur le même

ensemble de variables ; on dit également que ϕ est *logiquement plus forte* que ϕ', et que ϕ' est *logiquement plus faible* que ϕ. Notons encore une fois que cette définition autorise les ensembles $Var(\phi)$ et $Var(\phi')$ à être différents, et que dans le cas contraire elle revient à l'*inclusion* entre ensembles de modèles. Ainsi, la formule ϕ_1 implique la formule $\phi'_1 \vee x_1$. Remarquons que pour deux formules ϕ et ϕ', on a $\phi \equiv \phi'$ si et seulement si on a $\phi \models \phi'$ et $\phi' \models \phi$.

Enfin, une formule ϕ est dite *satisfaisable* si l'ensemble d'affectations $\mathcal{M}(\phi)$ est non vide, et *tautologique* s'il est égal à $\{0,1\}^{Var(\phi)}$. Ainsi, ϕ est satisfaisable si et seulement si la formule $\neg\phi$ n'est pas tautologique.

1.3 Formes normales

Il est facile de voir qu'étant donnée une formule ϕ, il existe un nombre infini de formules, arbitrairement complexes, logiquement équivalentes à ϕ ; en effet, il suffit de considérer l'ensemble des formules :

$$\phi, \phi' = (x \vee \phi) \wedge (\neg x \vee \phi), \phi'' = (x \vee \phi') \wedge (\neg x \vee \phi') \ldots$$

où $x \in \mathbb{V}$ est une variable quelconque. Il est donc utile de définir des formes (pseudo-)normales pour les formules. Nous utilisons dans ce mémoire la *Forme Normale Conjonctive* et la *Forme Normale Disjonctive*.

Un *littéral positif* est une formule réduite à une variable, et un *littéral négatif* est la négation d'une variable ; nous disons que les littéraux x et $\neg x$ sont *formés sur* la variable x, et qu'ils sont *opposés*. Une *clause* est une disjonction finie de littéraux formés sur des variables toutes différentes, par exemple $(x_1 \vee x_2 \vee \neg x_4 \vee x_5)$; nous notons également une clause comme l'ensemble de ses littéraux, ici $\{x_1, x_2, \neg x_4, x_5\}$, et utilisons de fait les notions d'union de deux clauses, de cardinal d'une clause, d'appartenance d'un littéral à une clause etc. ; tirant parti de la commutativité de la disjonction, nous ne tenons pas compte de l'ordre des littéraux dans une clause. Dualement, un *terme* est une conjonction finie de littéraux formés sur des variables différentes, que nous notons également comme un ensemble ; ainsi, la conjonction $(x_1 \wedge \neg x_2 \wedge x_4)$ est un terme. Notons qu'étant donnée une formule ϕ, il n'existe pas en général un littéral (resp. une clause, un terme) logiquement équivalent à ϕ. En revanche, pour toute formule propositionnelle ϕ il existe une formule logiquement équivalente à ϕ et en *Forme Normale Conjonctive* (FNC) : on appelle ainsi une conjonction de clauses toutes différentes. Ainsi, la formule :

$$\phi'_1 = (x_1 \vee x_2 \vee \neg x_4 \vee x_5) \wedge (x_2 \vee x_3 \vee x_4 \vee \neg x_5) \wedge (x_2 \vee \neg x_3 \vee x_4 \vee \neg x_5)$$

du paragraphe 1.2 est en Forme Normale Conjonctive, et logiquement équivalente, comme nous l'avons vu, à la formule ϕ_1. Nous notons également une telle formule comme l'ensemble de ses clauses, en utilisant les notions d'union, d'appartenance etc. Notons que si ϕ est une formule en FNC et m une affectation à $Var(\phi)$, m satisfait ϕ si et seulement si m satisfait au moins un littéral par clause de ϕ. Remarquons enfin que le fait d'imposer que les variables apparaissant dans une même clause, ou les clauses apparaissant dans une même formule FNC, soient toutes différentes n'est pas standard ; il nous permet de parler de ces objets comme d'ensembles.

Dualement, une formule en *Forme Normale Disjonctive* (FND) est une disjonction de termes tous différents, éventuellement notée comme leur ensemble. Ainsi, la formule :

$$\phi''_1 = (\neg x_4 \wedge \neg x_5) \vee (x_4 \wedge x_5) \vee (x_2) \vee (x_1 \wedge \neg x_2 \wedge x_4)$$

est en Forme Normale Disjonctive, et logiquement équivalente à la formule ϕ_1 du paragraphe 1.1. Si ϕ est une formule en FND, une affectation m à $Var(\phi)$ satisfait donc ϕ si et seulement si elle satisfait tous les littéraux d'au moins un terme de ϕ.

Nous notons VARS (resp. LITS, CLAUSES, TERMES) la classe de toutes les formules réduites à une variable (resp. à un littéral, à une clause, à un terme). De même, nous notons FNC (resp. FND) la classe de toutes les formules en Forme Normale Conjonctive (resp. en Forme Normale Disjonctive).

Notons que, puisque nous n'autorisons que des littéraux formés sur des variables toutes différentes dans une clause, et des clauses toutes différentes dans une formule FNC, la seule formule FNC tautologique est la formule ne contenant aucune clause (formée sur l'ensemble de variables $\emptyset$), et dualement la seule formule FND insatisfaisable est la formule ne contenant aucun terme. A l'inverse, pour tout ensemble de variables $V \neq \emptyset$ il existe en général plusieurs formules FNC insatisfaisables sur V : par exemple, les formules $((x_1) \wedge (\neg x_1 \vee x_2) \wedge (\neg x_1 \vee \neg x_2))$ et $((x_2) \wedge (x_1 \vee \neg x_2) \wedge (\neg x_1 \vee \neg x_2))$ sont deux formules FNC insatisfaisables sur $V = \{x_1, x_2\}$; dualement, il existe en général plusieurs formules FND tautologiques sur un même ensemble de variables.

Il est important de remarquer qu'étant donnée une formule ϕ en FNC, on peut facilement calculer une formule FND ϕ' logiquement équivalente à la formule propositionnelle $\neg\phi$ (qui n'est ni en FNC, ni en FND) : il suffit en effet de remplacer la conjonction entre les clauses de ϕ par la disjonction, la disjonction entre les littéraux d'une clause par la conjonction, et tous les littéraux par leur opposé. Nous notons $\phi' = Neg(\phi)$. Dualement, on peut calculer facilement une formule FNC ϕ' logiquement équivalente à $\neg\phi$ pour ϕ une formule FND, et nous notons de même $\phi' = Neg(\phi)$; notons que l'on a $Neg(Neg(\phi)) = \phi$ pour toute formule ϕ en FNC ou en FND. Par exemple, pour la formule ϕ''_1 ci-dessus on a :

$$Neg(\phi''_1) = (x_4 \vee x_5) \wedge (\neg x_4 \vee \neg x_5) \wedge (\neg x_2) \wedge (\neg x_1 \vee x_2 \vee \neg x_4)$$

Pour terminer, remarquons que les Formes Normales Conjonctive et Disjonctive ne sont pas à proprement parler *normales*. En effet, comme nous l'avons vu pour les formules FNC insatisfaisables, il est facile de construire plusieurs formules FNC formées sur le même ensemble de variables V et logiquement équivalentes ; ces formules représentent donc la même fonction booléenne de $\{0,1\}^V$ dans $\{0,1\}$. Néanmoins, nous conservons la qualification de *forme normale* car elle est standard.

1.4 Opérations et notions syntaxiques

Par opposition aux notions *sémantiques*, nous qualifions de *syntaxique* une notion définie pour une formule, et qu'on ne peut définir pour un ensemble d'affectations indépendamment d'une formule l'admettant pour ensemble de modèles. Nous présenterons par exemple dans la partie I des classes de formules, et nous verrons que deux formules ϕ_1, ϕ_2 peuvent vérifier $\phi_1 \equiv \phi_2$ mais $\phi_1 \in \mathcal{C}, \phi_2 \notin \mathcal{C}$ pour une telle classe $\mathcal{C}$; l'appartenance à $\mathcal{C}$ est donc une notion syntaxique.

Propagation La première opération syntaxique que nous utilisons dans le mémoire est la *propagation d'une affectation* (opération également appelée «conditionnement» lorsqu'elle est vue sous un angle sémantique [DM02]). Nous définissons tout d'abord l'opération de *simplification des constantes* dans une *expression propositionnelle* E, c'est-à-dire une formule propositionnelle dans laquelle certaines variables sont remplacées par l'une des constantes 0 ou 1. Cette opération consiste à appliquer inductivement les transformations décrites dans la table 1, tant que 0 ou 1 apparaissent dans E (dans la table 1, E' désigne une expression propositionnelle). Tous les connecteurs, sauf celui d'implication $\rightarrow$, étant commutatifs et associatifs, au moins l'une des transformations listées dans la table 1 peut bien être appliquée à toute expression propositionnelle dans laquelle apparaît au moins une constante. D'autre part, la simplification termine bien,

si E est de la forme...	remplacer E par...	si E est de la forme...	remplacer E par...
0	() ($\vee$ d'arité 0)	1	() ($\wedge$ d'arité 0)
$\neg 0$	1	$\neg 1$	0
$E' \vee 0$	E'	$E' \vee 1$	1
$E' \wedge 0$	0	$E' \wedge 1$	E'
$E' \oplus 0$	E'	$E' \oplus 1$	$\neg E'$
$E' \rightarrow 0$	$\neg E'$	$E' \rightarrow 1$	1
$0 \rightarrow E'$	1	$1 \rightarrow E'$	E'
$0 \leftrightarrow E'$	$\neg E'$	$1 \leftrightarrow E'$	E'

TAB. 1 – Simplification des constantes dans une expression propositionnelle

puisque à chaque étape, soit on diminue le nombre de constantes apparaissant dans l'expression, soit on diminue le nombre de connecteurs binaires sans augmenter le nombre de constantes, soit on supprime une occurrence du connecteur unaire $\neg$ sans augmenter les nombres de connecteurs binaires et de constantes. Enfin, il est aisé de voir que le résultat de la simplification est indépendant de l'ordre dans lequel sont considérées les expressions composant E.

Cette opération nous permet de définir la *propagation d'une affectation* dans une formule propositionnelle. Soient ϕ une formule propositionnelle et m une affectation à un sous-ensemble V de $Var(\phi)$. Nous définissons la formule $\phi[V : m]$ comme la formule sur $Var(\phi) \backslash V$ obtenue en remplaçant dans ϕ chaque occurrence d'une variable $x \in V$ par la valeur $m[x]$, puis en simplifiant les constantes ; nous disons que $\phi[V : m]$ est la formule obtenue par *propagation de m dans ϕ*. Par exemple, si l'on considère la restriction m_1'' de m_1 à l'ensemble de variables $V_1'' = \{x_1, x_4\}$ (paragraphe 1.1), on a $m_1''[x_1] = m_1''[x_4] = 1$ et on obtient :

$$\phi_1[V_1'' : m_1''] = (x_2 \oplus \neg(x_2 \vee x_5)) \leftrightarrow (\neg x_2 \rightarrow \neg x_5)$$

Il est facile de voir, enfin, que la propagation d'affectations préserve la Forme Normale Conjonctive ou Disjonctive si l'on élimine les doublons (clauses ou termes) après la simplification.

Impliqués et impliquants premiers Une deuxième notion syntaxique, très importante également pour ce mémoire, est la notion de *formule première*. Si ϕ est une formule propositionnelle, et C une clause, on dit que C est un *impliqué* de ϕ si ϕ implique C. Si de plus, aucune sous-clause propre de C n'est un impliqué de ϕ, on dit que C est un impliqué *premier* de ϕ. Cette notion de primalité est particulièrement importante lorsque ϕ est une formule en FNC et que C est une clause de ϕ ; par construction, C est alors un impliqué de ϕ, mais ce n'en est pas nécessairement un impliqué *premier*. Remarquons ainsi que pour la formule du paragraphe 1.2 :

$$\phi_1' = (x_1 \vee x_2 \vee \neg x_4 \vee x_5) \wedge (x_2 \vee x_3 \vee x_4 \vee \neg x_5) \wedge (x_2 \vee \neg x_3 \vee x_4 \vee \neg x_5)$$

la dernière clause n'est pas un impliqué *premier* de ϕ_1', puisque sa sous-clause propre $(x_2 \vee x_4 \vee \neg x_5)$ est un impliqué de ϕ_1'.

Par abus de langage, et lorsque la formule ϕ de référence est clairement donnée par le contexte, nous disons qu'une clause C d'une formule en FNC ϕ est *première* si C est un impliqué premier de ϕ ; nous disons alors qu'une formule FNC est *première* si toutes ses clauses sont premières. C'est principalement cette notion de primalité que nous utilisons dans le mémoire. Par exemple, la formule :

$$(x_1 \vee x_2 \vee \neg x_4 \vee x_5) \wedge (x_2 \vee x_4 \vee \neg x_5)$$

est première et logiquement équivalente à ϕ'_1.

De façon générale, nous disons qu'une clause C' *subsume* une clause C si c'en est une sous-clause ; nous utilisons la notation ensembliste $C' \subseteq C$. Il est alors facile de voir que $C' \subseteq C$ implique $C' \models C$. Mais remarquons également que pour une formule FNC $\phi = C_1 \wedge C_2 \wedge \cdots \wedge C_k$, où les C_i sont des clauses, si $\phi' = C'_1 \wedge C'_2 \wedge \cdots \wedge C'_k$ est une formule première obtenue en remplaçant, pour $i = 1, 2, \ldots, k$, la clause $C_i \in \phi$ par un impliqué premier C'_i de ϕ avec $C'_i \subseteq C_i$, alors les formules ϕ et ϕ' sont logiquement équivalentes. En effet, pour tout i on a par construction $C'_i \models C_i$, et on en déduit facilement $\phi' \models \phi$; réciproquement, puisque pour tout i la clause C'_i est un impliqué (premier) de ϕ, on a par définition $\forall i \in \{1, 2, \ldots, k\}, \phi \models C'_i$, et on en déduit $\phi \models \phi'$; finalement, on a bien $\phi \equiv \phi'$. Toute clause C d'une formule en FNC ϕ étant un impliqué de ϕ, on peut de plus toujours trouver pour une telle clause une clause *première* C' subsumant C, et on en déduit finalement que pour toute formule en FNC ϕ, on peut trouver une formule en FNC *première* logiquement équivalente à ϕ et dont les clauses sont obtenues en minimisant celles de ϕ.

Les notions d'*impliquants* d'une formule sont duales aux notions d'impliqués. Un terme T est ainsi appelé un *impliquant* d'une formule ϕ si on a $T \models \phi$, et il est appelé un impliqué *premier* de ϕ si aucun sous-terme propre T' de T ne vérifie $T' \models \phi$. Si ϕ est une formule FND et T est un terme de ϕ, T est par construction un impliquant de ϕ, et par abus de langage il est qualifié de *premier* s'il en est un impliquant *premier* ; ϕ est dite *première* si tous ses termes sont premiers. Comme pour les formules en *FNC* premières, il est facile de voir qu'on peut toujours trouver une formule en FND première, logiquement équivalente à une formule en FND donnée et obtenue en minimisant les termes de cette dernière. Par exemple, la formule en FND :

$$(\neg x_4 \wedge \neg x_5) \vee (x_4 \wedge x_5) \vee (x_2) \vee (x_1 \wedge x_4)$$

est première et logiquement équivalente à la formule ϕ''_1 du paragraphe 1.3.

Enfin, il est facile de voir qu'une clause C est un impliqué (resp. un impliqué premier) d'une formule en FNC ϕ si et seulement si le terme $Neg(C)$ est un impliquant (resp. un impliquant premier) de la formule en FND $Neg(\phi)$. Notons également que la clause vide (), dont l'ensemble de modèles est vide, est l'unique impliqué premier d'une formule ϕ si et seulement si ϕ est insatisfaisable ; en effet, on a alors $\phi \models ()$, et ainsi aucun autre impliqué de ϕ n'est premier. Cette remarque rend particulièrement importante la génération de l'ensemble des impliqués premiers d'une formule FNC, génération intensivement étudiée dans la littérature (voir l'état de l'art récent de Marquis [Mar00]).

Résolution Une propriété importante de l'ensemble des impliqués premiers d'une formule FNC est qu'il peut être obtenu par *résolution* des clauses de la formule [Qui59, CL73]. Tout d'abord, on dit que deux clauses C et C' sont *résolubles* s'il existe une variable x avec $x \in C$ et $\neg x \in C'$, et si pour toute variable $x' \neq x$ on n'a, ni $x' \in C$ et $\neg x' \in C'$, ni $\neg x' \in C$ et $x' \in C'$. On appelle alors *résolvant* de C et C' la clause $(C \backslash \{x\}) \cup (C' \backslash \{\neg x\})$. Par exemple, le résolvant des clauses $C = (x_1 \vee x_2 \vee \neg x_3)$ et $C' = (\neg x_2 \vee x_4)$ est la clause $(x_1 \vee \neg x_3 \vee x_4)$. Il est facile de voir que la conjonction de deux clauses résolubles implique leur résolvant, puisque, avec les mêmes notations que ci-dessus, tout modèle de $C \wedge C'$ affectant 0 à x doit satisfaire $C \backslash \{x\}$, et tout modèle affectant 1 à x doit satisfaire $C' \backslash \{\neg x\}$; finalement, tout modèle de $C \wedge C'$ doit satisfaire $C \backslash \{x\}$ ou $C' \backslash \{\neg x\}$.

Etant donnée une formule FNC ϕ, l'opération qui ajoute à ϕ le résolvant de deux de ses clauses est appelée un *pas de résolution* ; la remarque précédente nous dit que la formule ainsi obtenue est logiquement équivalente à ϕ. Le résultat suivant est alors fondamental.

Théorème ([Qui59]) *Soit ϕ une formule FNC. Alors tout impliqué premier de ϕ est ajouté à ϕ après un nombre fini de pas de résolutions.*

Ainsi, en reprenant la formule du paragraphe 1.2 :

$$\phi_1' = (x_1 \vee x_2 \vee \neg x_4 \vee x_5) \wedge (x_2 \vee x_3 \vee x_4 \vee \neg x_5) \wedge (x_2 \vee \neg x_3 \vee x_4 \vee \neg x_5)$$

on voit que les seules clauses résolubles de ϕ_1' sont les deux dernières, et en ajoutant leur résolvant à ϕ_1' on obtient la formule :

$$(x_1 \vee x_2 \vee \neg x_4 \vee x_5) \wedge (x_2 \vee x_3 \vee x_4 \vee \neg x_5) \wedge (x_2 \vee \neg x_3 \vee x_4 \vee \neg x_5) \wedge (x_2 \vee x_4 \vee \neg x_5)$$

dans laquelle on ne peut pas trouver deux nouvelles clauses résolubles. On en déduit donc qu'elle contient tous les impliqués premiers de ϕ_1'. Par construction, elle est également logiquement équivalente à ϕ_1', et le reste si on retire celles de ses clauses qui sont subsumées par d'autres, ce qui donne la formule :

$$(x_1 \vee x_2 \vee \neg x_4 \vee x_5) \wedge (x_2 \vee x_4 \vee \neg x_5)$$

La résolution *unitaire* est un cas particulier de résolution, celui où l'une des deux clauses résolues ne contient qu'un littéral.

2 Rappels de complexité

Nous abordons la plupart des problèmes de ce mémoire d'un point de vue *algorithmique* ; nous nous intéressons donc à la *complexité en temps* des problèmes et des algorithmes qui les résolvent. Nous rappelons ici les notions, encore une fois toutes classiques, que nous utiliserons ; pour plus de détails nous renvoyons le lecteur à l'ouvrage très complet de Papadimitriou [Pap94] (chapitres 2, 8 et 10 principalement).

Nous supposerons toujours le modèle de calcul classique des machines *RAM* (voir par exemple l'ouvrage d'Aho *et al.* [AHU74, chapitre 1]).

2.1 Problèmes et complexité

Les problèmes que nous étudions sont de deux types : problèmes *de décision* et problèmes *avec sortie*. Les problèmes de décision sont ceux attendant la réponse «Oui» ou «Non», et les problèmes avec sortie sont ceux impliquant le calcul d'une réponse non booléenne. Une partie des notions de complexité sont différentes pour ces deux types de problèmes.

Etant donné un problème `P`, de décision ou avec sortie, la *complexité en temps* (dans le pire cas) d'un algorithme résolvant `P` est une fonction de la taille de la donnée du problème, éventuellement mesurée par plusieurs paramètres, qui donne le nombre maximum d'opérations élémentaires effectuées par l'algorithme sur une entrée de cette taille, à une constante multiplicative près ; nous considérons comme élémentaire une opération qui requiert un temps constant d'exécution, comme l'accès à un élément d'un tableau par son indice ou encore la négation d'une valeur booléenne. Nous disons alors qu'un algorithme est de complexité en temps $O(f(n_1, n_2, \ldots, n_k))$, où f est une fonction définie sur les k-uplets d'entiers naturels, s'il existe une constante C telle que pour toute donnée de taille mesurée par les paramètres entiers $n_1, n_2, \ldots, n_k$, l'algorithme effectue moins de $Cf(n_1, n_2, \ldots, n_k)$ opérations élémentaires avant de terminer. En particulier, une complexité en $O(n)$ est dite *linéaire*, une complexité en $O(n^2)$, *quadratique*, et une complexité en $O(p(n))$ pour un certain polynôme p est dite *polynomiale*. Nous disons d'un algorithme de complexité en temps polynomiale (ou simplement *polynomial*) qu'il est *efficace*, et d'un problème admettant un tel algorithme qu'il est *traitable* ; ces termes seront motivés plus loin.

2.2 Problèmes de décision

Les problèmes de *décision* que nous étudions font quasiment tous partie de la classe P ou de la classe NP. La classe P est la classe des problèmes de décision admettant un algorithme polynomial. Pour ce qui est de la classe NP, nous avons besoin de la notion de *témoin*. Qualifions de *positive* (resp. *négative*) une donnée d'un problème de décision $\mathtt{P}$ pour laquelle la réponse est «Oui» (resp. «Non»). Etant donné un tel problème, on dit qu'une fonction $T_{\mathtt{P}}$ associe un *témoin* à toute donnée positive de $\mathtt{P}$ s'il existe un algorithme $V_{\mathtt{P}}$ qui *vérifie* le témoin $T_{\mathtt{P}}(D)$ pour toute donnée D du problème, c'est-à-dire qui répond «Oui» sur une donnée (D, T) si D est une instance positive du problème et $T = T_{\mathtt{P}}(D)$, et «Non» dans tous les autres cas. La classe NP est alors la classe des problèmes pour lesquels il existe, d'une part une fonction associant un témoin de taille polynomiale en la taille de D à toute donnée positive D du problème, et d'autre part un algorithme de complexité en temps polynomiale qui vérifie ces témoins. Intuitivement, c'est donc la classe des problèmes pour lesquels il «suffit» de deviner un témoin, le reste du travail pouvant être effectué en temps polynomial. Bien évidemment, la classe P est incluse dans la classe NP, puisque pour un problème de P la réponse «Oui» est pour toute donnée positive un témoin convenable, vérifiable en temps polynomial par un algorithme qui résout le problème.

En revanche, on ne sait pas si, réciproquement, la classe NP est incluse dans la classe P ; il est cependant largement conjecturé que ce n'est pas le cas, et donc qu'il existe des problèmes dans NP\P, qui sont plus difficiles que ceux de P : c'est la fameuse hypothèse P≠NP, que nous admettons dans ce mémoire.

Parmi les problèmes de NP, les plus difficiles sont les problèmes *NP-complets* : on appelle ainsi un problème $\mathtt{P}$ de NP auquel on peut réduire tout problème $\mathtt{Q}$ de NP en temps polynomial, c'est-à-dire tel que, si on trouvait un algorithme polynomial pour $\mathtt{P}$, on pourrait l'utiliser pour résoudre en temps polynomial tout problème $\mathtt{Q}$ de NP ; en effet, une *réduction (fonctionnelle)* de $\mathtt{Q}$ à $\mathtt{P}$ est un algorithme $\mathtt{R}$ transformant toute donnée D pour $\mathtt{Q}$ en une donnée $\mathtt{R}(D)$ pour $\mathtt{P}$ de sorte que $\mathtt{R}(D)$ soit une donnée positive pour $\mathtt{P}$ si et seulement si D est une donnée positive pour $\mathtt{Q}$.

Plus généralement, on qualifie de *NP-difficile* un problème de décision auquel on peut réduire en temps polynomial tout problème de NP ; les problèmes *NP-complets* sont donc les problèmes NP-difficiles de NP. Pour montrer qu'un problème $\mathtt{P}$ est NP-difficile, il est facile de voir qu'il suffit par exemple d'exhiber une réduction polynomiale d'un problème NP-complet à $\mathtt{P}$. Le premier problème montré NP-complet [Coo71] est le problème SAT, que nous redonnons ci-dessous.

Problème (SAT)

Donnée : *Une formule propositionnelle* ϕ
Question : ϕ *est-elle satisfaisable ?*

Pour les problèmes NP-complets, les meilleurs algorithmes connus sont de complexité en temps exponentielle dans le pire cas ; c'est pourquoi nous qualifions les algorithmes polynomiaux d'*efficaces*. On utilise donc souvent des *heuristiques* pour résoudre des problèmes NP-complets, c'est-à-dire des algorithmes qui sont expérimentalement montrés efficaces dans certaines situations pratiques, mais dont la complexité en temps dans le pire cas n'est pas polynomiale. Nous n'aborderons cependant pas ce sujet dans ce mémoire.

Enfin, la classe duale de la classe NP est la classe coNP des problèmes dont la *négation* est dans NP, c'est-à-dire les problèmes dont les témoins attestent d'une réponse négative ; les

notions de réduction, de difficulté et de complétude sont les mêmes que pour la classe NP, et le problème coNP-complet typique est le problème `TAUT`, qui consiste à déterminer si une formule FND donnée est tautologique ; la négation de ce problème est en effet le problème `SAT` pour les formules FNC, via la transformation $\phi \mapsto Neg(\phi)$ (voir paragraphe 1.3).

2.3 Problèmes avec sortie

Les classes de complexité des problèmes *avec sortie* ne sont pas définies de la même manière que celles des problèmes de décision, même si la notion de témoin a toujours un sens dans le cas des problèmes de *comptage* par exemple (voir la définition de ces problèmes par Kozen [Koz92]). Nous disons toujours d'un algorithme résolvant un tel problème qu'il est *efficace*, et le problème associé *traitable*, s'il est de complexité en temps polynomiale, mais pour montrer des résultats de difficulté nous utilisons deux méthodes.

Tout d'abord, si la réponse à calculer est de taille exponentielle (dans le pire cas) en la taille de la donnée, nous considérons que le problème est intraitable ; en effet, tout algorithme résolvant un tel problème doit au moins écrire sa réponse, et donc ne peut être de complexité en temps polynomiale. Notons qu'il existe des notions de complexité *en fonction de la taille de la réponse*, ou encore *d'énumération* (toutes introduites par Johnson *et al.* [JYP88]), pour étudier ces problèmes plus finement, mais nous ne nous en servirons presque pas dans ce mémoire, et les repréciserons en temps utile.

L'autre méthode pour montrer qu'un problème avec sortie est difficile consiste à exhiber un problème de décision *associé*, c'est-à-dire admettant les mêmes données et dont la réponse puisse être calculée efficacement à partir de la réponse au problème avec sortie ; on voit alors aisément que si le problème de décision est difficile, le problème avec sortie l'est au moins autant. Par exemple, le problème consistant à calculer, s'il en existe un, un modèle d'une formule propositionnelle donnée, est évidemment au moins aussi difficile que le problème `SAT`, puisque si l'on a résolu ce problème, on sait que la formule donnée est satisfaisable si et seulement si on a réussi à en calculer un modèle.

2.4 Représentation des objets

Pour terminer, nous discutons rapidement la représentation des objets qui nous intéressent. De cette représentation dépend en effet la complexité des algorithmes.

Formules Pour aucun des objets qui nous intéressent nous ne supposons une représentation particulière, mais simplement une représentation *raisonnable*. Pour une formule ϕ de la logique propositionnelle, nous supposons simplement que la représentation de ϕ permet de déterminer le connecteur de plus haut niveau de ϕ (par exemple, le connecteur $\wedge$ pour une formule en FNC, ou le connecteur $\vee$ pour une formule en FND) en temps constant, et d'itérer sur les opérandes de ce connecteur (les clauses de ϕ si ϕ est en FNC, ses termes si elle est en FND) avec un délai constant. Ainsi, une représentation d'une formule par un arbre, dont la racine est étiquetée par l'opérateur de plus haut niveau de ϕ, les sous-arbres représentent les formules qui sont ses opérandes, et les feuilles sont étiquetées par les variables, est une représentation raisonnable (voir la figure 2 pour la représentation sous cette forme de la formule ϕ_1 du paragraphe 1.1).

La taille (au sens de la complexité) d'une formule ϕ est mesurée par le nombre d'occurrences de variables dans ϕ ; la taille de ϕ_1, par exemple, est donc 11.

Enfin, il est important de noter que le modèle RAM nous permet de trier les clauses (resp. les termes) d'une formule FNC (resp. FND) en temps linéaire en la taille de la formule, et ainsi

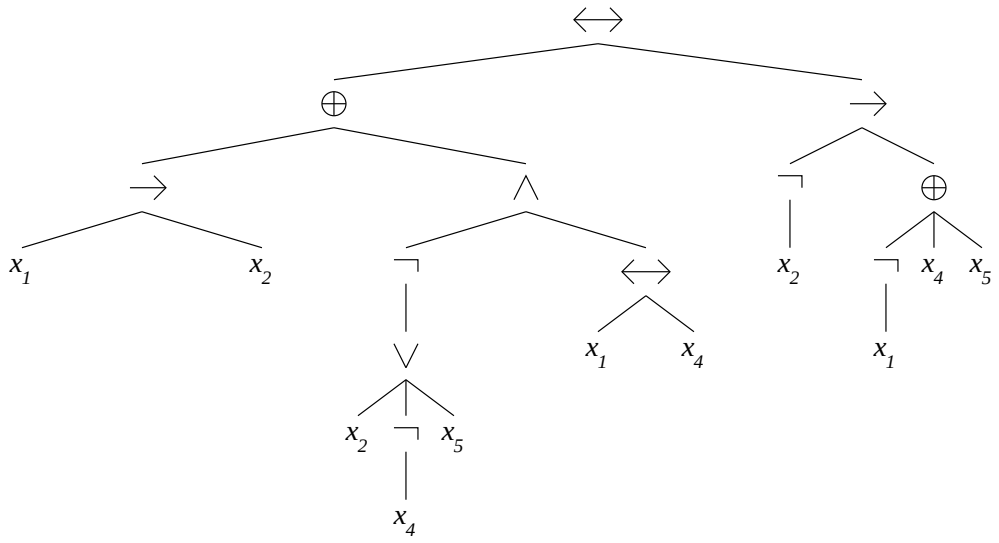

FIG. 2 – Représentation par un arbre de la formule $\phi_1 = ((x_1 \rightarrow x_2) \oplus (\neg(x_2 \vee \neg x_4 \vee x_5) \wedge (x_1 \leftrightarrow x_4))) \leftrightarrow (\neg x_2 \rightarrow (\neg x_1 \oplus x_4 \oplus x_5))$

d'obéir facilement à l'hypothèse selon laquelle les clauses (resp. les termes) d'une même formule sont toutes différentes. Pour plus de précisions sur les machines RAM et les opérations dans ce modèle, nous renvoyons encore une fois le lecteur à l'ouvrage d'Aho *et al.* [AHU74].

Affectations Pour ce qui est de la représentation d'une affectation m à un ensemble de variables V, notre seule hypothèse sera que l'on peut accéder à la valeur $m[x]$ en temps constant pour une variable $x \in V$ donnée. Ainsi, la représentation d'une affectation m à V par un tableau des valeurs $m[x_i]$, indicé par l'ensemble $\{i \in \mathbb{N} \,|\, x_i \in V\}$, est adéquate. Cette hypothèse sur la représentation des affectations est un peu plus restrictive que celle sur la représentation des formules, puisqu'elle interdit par exemple la représentation par une liste de couples de la forme $\{(x_i, m[x_i]) \,|\, x_i \in V\}$, mais elle est néanmoins naturelle. La taille d'une affectation m à un ensemble de variables V est simplement le cardinal $|V|$ de V.

Ensembles d'affectations Enfin, concernant la représentation d'un *ensemble* d'affectations à un même ensemble de variables, nous demandons simplement qu'elle permette d'itérer sur les affectations de l'ensemble avec un temps constant pour passer de l'une à l'autre, et qu'elle permette d'ajouter une nouvelle affectation en temps constant. Une liste d'affectations est donc une représentation convenable.

Néanmoins, nous utiliserons à plusieurs reprises la représentation d'un tel ensemble par un *arbre de décision*. Etant donné un ensemble d'affectations M à un ensemble de variables V, un tel arbre est construit en associant une profondeur à chaque variable de V, et un chemin de la racine à une feuille (toutes de profondeur maximale) à chaque affectation m de M ; si $m[x]$ vaut 0 on ajoute un fils gauche au noeud courant (de profondeur correspondant à x), et sinon on lui ajoute un fils droit. Une fois l'ordre (total strict) sur les variables fixé, cet arbre représente donc M de façon canonique. Par exemple, l'arbre de la figure 3 représente l'ensemble d'affectations $\mathcal{M}(\phi_1)$ à l'ensemble de variables $\{x_1, x_2, x_4, x_5\}$. L'intérêt de cette représentation est qu'elle peut être

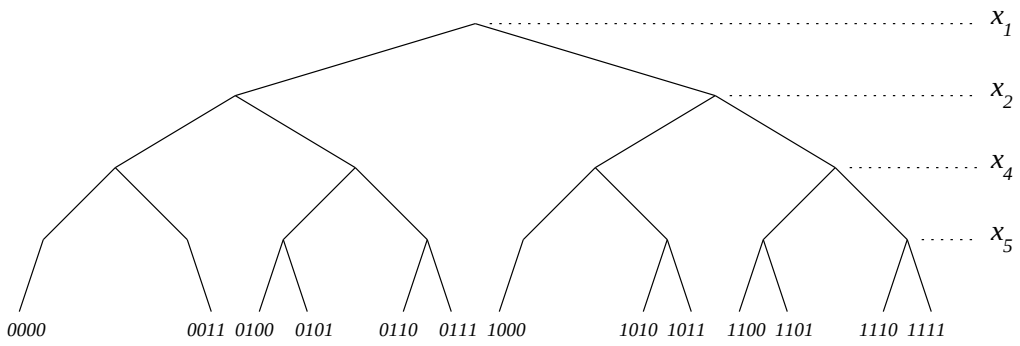

FIG. 3 – Arbre de décision associé à $\mathcal{M}(\phi_1) = \{0,1\}^{V_1} \setminus \{0001, 0010, 1001\}$

construite en temps $O(|M||V|)$ pour un ensemble d'affectations M à un ensemble de variables V, et qu'elle permet de décider en temps $O(|V|)$ si une affectation à V donnée est dans M. Ces deux assertions étant très intuitives, nous n'en redonnons pas la preuve ici. Pour plus de détails, nous renvoyons le lecteur à l'ouvrage d'Aho *et al.* [AHU83, paragraphe 5.3].

3 Représentation de connaissances en logique propositionnelle

Nous présentons, pour terminer ce chapitre, notre conception de la représentation de connaissances en logique propositionnelle. Nous terminons par deux exemples jouets, que nous utiliserons ensuite de façon récurrente à des fins d'illustration.

3.1 Ensembles d'observations et bases de connaissances

Nous considérons qu'un système capable de manipuler des connaissances a à sa disposition un ensemble fini de variables booléennes, aussi appelées *variables d'état*, *descripteurs*, ou encore *attributs*. Nous appelons *petit monde* la réalité à propos de laquelle il manipule des connaissances par le biais de ces variables. Ainsi, un système manipulant des connaissances à propos de la pièce dans laquelle il évolue pourrait avoir à sa disposition des variables indiquant si la lumière est allumée, si la porte est ouverte, si de l'air entre dans la pièce, si au moins un tableau est accroché au mur situé au nord, etc. Son petit monde est alors l'ensemble des informations relatives à la pièce qui peuvent être décrites par ces variables. Un état (ou *situation*) précis de la pièce peut donc être formalisé par une affectation aux variables.

Nous voyons la *connaissance* que notre système peut alors avoir à propos de son petit monde comme un ensemble de *situations envisageables*, décrivant implicitement la façon dont peuvent interagir les variables. Ainsi, si une situation dans laquelle la porte est ouverte mais l'air n'entre pas dans la pièce n'est pas envisageable, les variables correspondantes sont implicitement liées par la relation «Si la porte est ouverte, alors l'air entre dans la pièce».

Cette connaissance peut être représentée de différentes manières. Ainsi, l'ensemble des situations envisageables dans le petit monde de référence représente de manière non ambiguë une connaissance à son propos ; une situation étant formalisée par une affectation aux variables du système, on peut voir un ensemble d'affectations à ces variables comme une connaissance ; nous disons qu'un tel ensemble représente une connaissance *en extension*. Mais une telle connaissance peut également être représentée *en intension* : une formule est en effet une telle représentation,

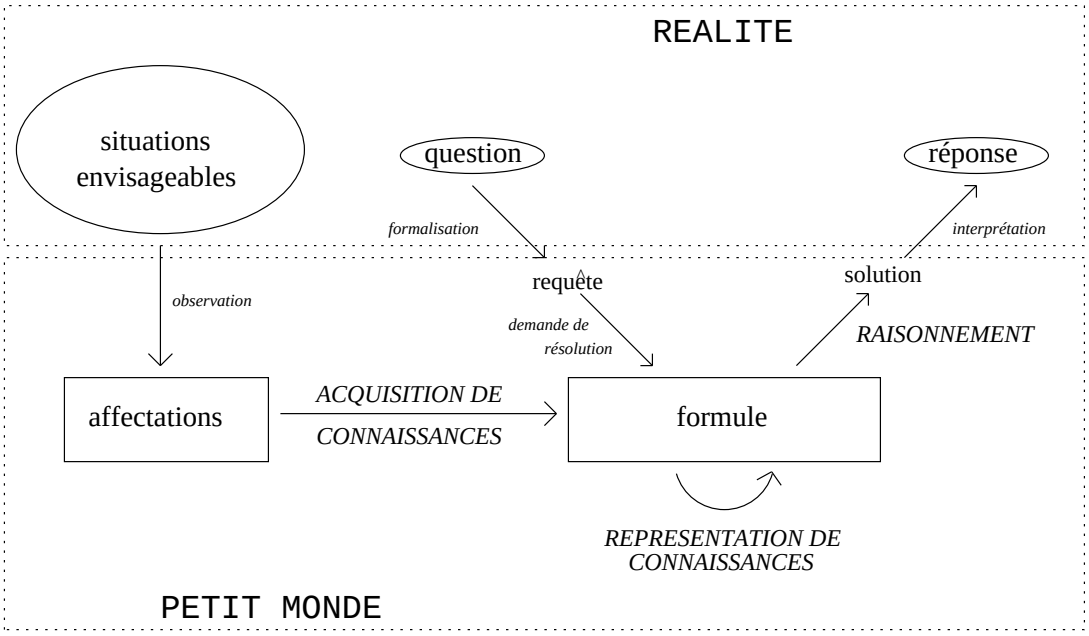

FIG. 4 – Problématiques et objets liés aux systèmes à base de connaissances

non ambiguë, si elle admet l'ensemble des affectations représentant les situations envisageables pour ensemble de modèles. De façon générale, nous appelons *base de connaissances sur* V une formule propositionnelle sur V. Nous disons alors d'un système capable de manipuler de tels objets qu'il est *à base de connaissances*. La figure 4 reprend la figure 1 de l'introduction, en ajoutant aux problèmes et aux concepts auxquels nous nous intéressons dans ce mémoire les objets précis que nous manipulons.

Ainsi, un système à base de connaissances peut utiliser ses connaissances pour *raisonner* à propos de son petit monde. Sur l'exemple, il peut ainsi se demander si la porte peut être ouverte, et la lumière allumée, en même temps. Il peut alors *calculer* la réponse (positive !) à cette question en demandant à sa base de connaissances s'il existe une situation envisageable dans laquelle la lumière est allumée et la porte, ouverte.

Si la logique propositionnelle n'est pas très expressive, la vision de la représentation des connaissances que nous adoptons est toutefois classique, et adoptée dans de nombreux travaux. Ce point de vue est notamment celui de travaux de Khardon et Roth [KR94, Kha95, KR96] ou de Selman et Kautz [KKS95, SK96]. Un exemple similaire à ceux que nous présentons dans les paragraphes 3.2 et 3.3 est décrit par Russell et Norvig [RN95, chapitre 6]. Pour des descriptions d'autres formalismes de représentation de connaissances, nous renvoyons le lecteur aux ouvrages généraux déjà cités [RK91, RN95, BCZ01].

Bien entendu, une base de connaissances décrit une certaine réalité dans la limite, notamment, de la pertinence des variables choisies, et donc de la pertinence du petit monde choisi pour formaliser cette réalité ; ainsi, des variables inutiles (par exemple, une variable indiquant s'il y a des ravins sur Mars) ne remettent pas en cause la pertinence du petit monde, mais à l'inverse il peut manquer des variables importantes à cette modélisation. Nous ne discutons cependant pas ces aspects dans ce mémoire, et supposons simplement un ensemble de variables définissant un petit monde.

x_{PV}	x_{VP}	x_{VV}	x_{DT}
0	0	0	0
0	0	1	0
0	1	0	1
1	0	0	0
1	0	1	0
1	1	0	1

TAB. 2 – Les situations envisageables dans le petit monde du carrefour

Notons enfin qu'une base de connaissances, si elle est associée par un système à une autre notion (un *concept*), peut être vue, non plus comme une représentation des situations envisageables dans un petit monde, mais comme une représentation des *exemples* de ce concept. Par exemple, dans un petit monde décrit par deux variables, x_{IR} et x_{UV}, indiquant respectivement si un rayonnement est dans le domaine de l'infrarouge ou dans celui de l'ultraviolet, une base de connaissances peut être associée au concept «visible» : alors l'affectation de 0 aux deux variables est un *exemple* de ce concept, tandis que celle de 0 à x_{IR} et de 1 à x_{UV} en est un *contre-exemple*. Une telle base de connaissances est alors celle représentée en intension par la formule $(\neg x_{IR} \wedge \neg x_{UV})$. Notons qu'alors il existe en réalité trois types de situations : celles correspondant à des *exemples* du concept, celles correspondant à des *contre-exemples*, et enfin celles qui sont absurdes (l'affectation de 1 aux deux variables, dans ce cas). Néanmoins, ceci est transparent pour un système utilisant une telle base de connaissances ; la seule conséquence du fait que certaines situations sont absurdes est que le système ne peut jamais les observer. Notons enfin que pour de telles bases de connaissances, la représentation par une formule peut logiquement recevoir la dénomination de *définition* du concept.

3.2 Un problème de carrefour

Notre premier exemple jouet consiste en l'expression en logique propositionnelle de la connaissance liée au petit monde d'un carrefour et à la traversée de ce carrefour par un piéton. Quatre variables paraissent pertinentes pour décrire ce petit monde (pour simplifier, nous supposons que les feux ne sont jamais oranges) :

- Une variable indiquant la présence (ou non) de véhicules aux abords du carrefour
- Une variable représentant la couleur du feu pour piétons (vert ou rouge)
- Une variable représentant la couleur du feu pour véhicules (vert ou rouge)
- Une variable représentant le droit (ou non) de traverser.

Nous introduisons donc les quatre variables propositionnelles x_{PV} (Présence de Véhicules), x_{VP} (Vert Piéton), x_{VV} (Vert Véhicules) et x_{DT} (Droit de Traverser). Notons qu'exceptionnellement, et afin de laisser disponible leur signification intuitive, nous n'indiçons pas ces variables par des entiers, c'est-à-dire que ce ne sont pas des éléments de $\mathbb{V}$; lorsqu'il sera nécessaire de les considérer comme totalement ordonnées, nous utiliserons l'ordre :

$$x_{PV} < x_{VP} < x_{VV} < x_{DT}$$

On voit alors aisément que toutes les situations envisageables sont celles représentées dans la table 2. Par exemple, la deuxième ligne de la table 2 représente la situation dans laquelle il n'y a pas de véhicules aux abords du feu ($x_{PV} \leftarrow 0$), le feu pour piétons est rouge ($x_{VP} \leftarrow 0$), le feu pour véhicules est vert et où on n'a (donc) pas le droit de traverser le carrefour à pied.

Le table 2 représente donc une connaissance en extension, et, dans le formalisme de la logique propositionnelle, est représentée par l'ensemble d'affectations :

$$M_C = \{0000, 0010, 0101, 1000, 1010, 1101\}$$

à l'ensemble de variables $V_C = \{x_{PV}, x_{VP}, x_{VV}, x_{DT}\}$. En conséquence, un système muni de cette connaissance peut l'utiliser pour raisonner dans le petit monde du carrefour, par exemple pour déduire que dès que le feu pour piétons est vert, celui pour véhicules est rouge.

Mais, comme nous l'avons vu dans le paragraphe précédent, un système muni de cette connaissance peut aussi la représenter en intension, par une formule admettant M_C pour ensemble de modèles. Ainsi, la formule en FNC :

$$\phi_C = (\neg x_{DT} \vee x_{VP}) \wedge (\neg x_{VP} \vee x_{DT}) \wedge (\neg x_{PV} \vee \neg x_{VV} \vee \neg x_{DT}) \wedge (\neg x_{VP} \vee \neg x_{VV})$$

vérifie $M_C = \mathcal{M}(\phi_C)$, et M_C et ϕ_C représentent donc différemment les mêmes connaissances. Intuitivement, la formule ϕ_C représente la base de connaissances comme un ensemble de règles :

- Soit il est interdit de traverser, soit le feu pour piétons est vert, *autrement dit* : si on a le droit de traverser, c'est que le feu pour piétons est vert, *et*
- Si le feu pour piétons est vert, on a le droit de traverser, *et*
- Il n'est pas envisageable qu'il y ait des véhicules aux abords du carrefour, que le feu pour véhicules soit vert et qu'on ait le droit de traverser (en même temps), *et*
- Il n'est pas envisageable que les feux pour piétons et pour véhicules soient verts (en même temps).

Mais nous avons vu que plusieurs formules pouvaient être logiquement équivalentes. En effet, la formule en FND :

$$\phi'_C = (\neg x_{PV} \wedge \neg x_{VP} \wedge \neg x_{DT}) \vee (\neg x_{VP} \wedge x_{VV} \wedge \neg x_{DT}) \vee (x_{VP} \wedge \neg x_{VV} \wedge x_{DT}) \vee (\neg x_{VP} \wedge \neg x_{VV} \wedge \neg x_{DT})$$

représente la même base de connaissances que ϕ_C, puisque l'on a $M_C = \mathcal{M}(\phi'_C)$ (et donc $\phi_C \equiv \phi'_C$). Intuitivement, on peut voir une telle formule comme une énumération des situations envisageables, de même qu'un ensemble d'affectations, mais décrites de façon plus compacte :

- Il n'y a pas de véhicules aux abords du carrefour, le feu pour piétons est rouge et il est interdit de traverser, *ou*
- Le feu pour piétons est rouge, celui pour véhicules est vert et il est interdit de traverser, *ou*
- Le feu pour piétons est vert, celui pour véhicules est rouge et on a le droit de traverser, *ou*
- Le feu pour piétons est rouge ainsi que celui pour véhicules, et il est interdit de traverser.

Dans tout le mémoire, lorsque nous utiliserons cet exemple nous y ferons référence comme «l'exemple du carrefour».

3.3 A propos de thèses...

Le deuxième exemple jouet que nous présentons concerne la lecture d'une thèse (d'informatique). Notre but est de décrire en logique propositionnelle les conditions de lecture, la qualité du travail et le souvenir que peut laisser cette lecture. Nous introduisons les variables suivantes :

- x_{me}, qui indique si la lecture est mémorable (ou non)
- x_{qu}, qui indique si le document est de qualité
- x_{ba}, qui indique si le travail est bâclé
- x_{in}, qui indique si le lecteur est intéressé par le sujet
- x_{ch}, qui indique si le lecteur est chercheur en informatique.

x_{me}	x_{qu}	x_{ba}	x_{in}	x_{ch}
0	0	1	1	1
0	1	0	0	0
1	0	1	0	0
1	1	0	1	1

TAB. 3 – Les situations envisageables dans le petit monde de la thèse

Encore une fois nous listons les différentes situations envisageables dans la table 3. Ainsi, la troisième ligne de cette table, par exemple, représente la situation où la lecture de la thèse est (tristement) mémorable, le document est médiocre, le travail bâclé, et où le lecteur n'est pas intéressé par le sujet, ni chercheur en informatique.

Encore une fois nous faisons confiance à la modélisation du petit monde, et nous considérons que la table 3 représente une connaissance lorsqu'elle est vue comme l'ensemble d'affectations suivant (à l'ensemble de variables $V_t = \{x_{me}, x_{qu}, x_{ba}, x_{in}, x_{ch}\}$, totalement ordonné par la relation $x_{me} < x_{qu} < x_{ba} < x_{in} < x_{ch}$) :

$$M_t = \{00111, 01000, 10100, 11011\}$$

Cependant, cette connaissance peut encore une fois être tout autant représentée par des formules. Ainsi, la formule en FNC :

$$\begin{array}{rll} \phi_t = & (x_{qu} \vee x_{ba}) \wedge (\neg x_{qu} \vee \neg x_{ba}) & \\ \wedge & (x_{me} \vee \neg x_{ba} \vee x_{in}) \wedge (x_{me} \vee x_{ba} \vee \neg x_{in}) \wedge (\neg x_{me} \vee x_{ba} \vee x_{in}) & \\ \wedge & (\neg x_{me} \vee \neg x_{ba} \vee \neg x_{in}) & \\ \wedge & (\neg x_{in} \vee x_{ch}) \wedge (x_{in} \vee \neg x_{ch}) & \end{array}$$

admet M_t pour ensemble de modèles, et représente donc la même connaissance. Quant à la représentation de cette dernière en FND, elle est unique, et c'est la réécriture comme disjonction de termes de l'ensemble de modèles M_t :

$$\begin{array}{rlll} \phi'_t = & (\neg x_{me} \wedge \neg x_{qu} \wedge x_{ba} \wedge x_{in} \wedge x_{ch}) & \vee & (\neg x_{me} \wedge x_{qu} \wedge \neg x_{ba} \wedge \neg x_{in} \wedge \neg x_{ch}) \\ \vee & (x_{me} \wedge \neg x_{qu} \wedge x_{ba} \wedge \neg x_{in} \wedge \neg x_{ch}) & \vee & (x_{me} \wedge x_{qu} \wedge \neg x_{ba} \wedge x_{in} \wedge x_{ch}) \end{array}$$

C'est donc essentiellement la même chose de représenter les connaissances relatives à ce petit monde par l'ensemble d'affectations M_t ou par la formule en FND ϕ'_t ; insistons néanmoins sur le fait que c'est vrai sur cet exemple, mais pas dans le cas général, où il existe plusieurs formules FND logiquement équivalentes. En revanche, nous verrons au chapitre 3 que cette base de connaissances admet encore une autre représentation, proche de la FNC mais plus compacte que ϕ_t, sous la forme d'une formule affine.

Dans tout le mémoire, nous ferons référence à cet exemple comme «l'exemple de la thèse».

Première partie

Classes de formules

Chapitre 1

Introduction

Sommaire

1.1	**Quelle forme normale pour les formules propositionnelles ?**	**25**
1.2	**Classes de formules**	**27**
1.3	**Classes étudiées dans le mémoire**	**30**

Nous nous intéressons dans cette première partie aux classes de formules propositionnelles dans lesquelles il est intéressant de représenter des connaissances. Dans ce chapitre, nous justifions tout d'abord notre choix des Formes Normales Conjonctive et Disjonctive comme langage général de représentation, puis nous présentons nos motivations et notre approche pour étudier des sous-classes de la classe de toutes les formules FNC (resp. FND), et enfin nous présentons rapidement celles de ces sous-classes que nous avons choisi d'étudier ; leurs définitions précises ainsi que des motivations plus riches seront présentées dans les chapitres suivants.

1.1 Quelle forme normale pour les formules propositionnelles ?

Comme nous l'avons vu, les représentations par une formule d'une même base de connaissances relative à un petit monde peuvent être nombreuses, très différentes et arbitrairement complexes. Le choix de formes normales, ou pseudo-normales, pour ces formules est donc nécessaire. Nous avons présenté, dans les préliminaires généraux, la *Forme Normale Conjonctive* (FNC) et la *Forme Normale Disjonctive* (FND), mais de nombreuses autres représentations existent. Mentionnons par exemple la *Forme Normale Algébrique* (voir [CG99] par exemple), ou exclusif de termes (ou *monômes*), couramment utilisée pour l'étude des codes correcteurs d'erreurs ou pour la cryptographie ; la représentation de la base de connaissances du petit monde du carrefour sous cette forme, par exemple, est :

$$1 \oplus (x_{DT}) \oplus (x_{VP}) \oplus (x_{VP} \wedge x_{VV} \wedge x_{DT})$$

Mentionnons également la *Forme Normale Négative* (voir [DM02] par exemple), dans laquelle la seule restriction est que le connecteur $\neg$ ne porte que sur des variables, et qui généralise donc la FNC et la FND. Par exemple, la formule :

$$(\neg x_{VP} \wedge \neg x_{DT}) \vee ((x_{VP}) \wedge (x_{DT}) \wedge (\neg x_{VP} \vee \neg x_{VV}))$$

admet aussi pour ensemble de modèles l'ensemble M_C correspondant au petit monde du carrefour ; elle est en Forme Normale Négative, mais ni en FNC ni en FND. Enfin, d'autres représentations que les formules existent, comme par exemple la représentation par un Arbre de

Décision, par un Diagramme de Décision Binaire [Bry86, HI02], etc. Le récent article de Darwiche et Marquis [DM02] étudie de nombreuses formes normales ou pseudo-normales pour les formules propositionnelles du point de vue de la représentation de connaissances.

Si notre choix de représenter une base de connaissances par une formule plutôt que par un ensemble d'affectations est motivé par la taille de ces différentes représentations, notre choix des Formes Normales Conjonctive et Disjonctive est motivé par l'optique de l'acquisition et de la représentation de connaissances ainsi que par celle de l'utilisation de ces connaissances pour le raisonnement.

Forme Normale Conjonctive Nous considérons tout d'abord la Forme Normale Conjonctive comme un langage de représentation adéquat. Nos principales motivations sont la stabilité de ce langage par conjonction et sa lisibilité.

La stabilité de la FNC par conjonction, tout d'abord, c'est-à-dire le fait que si ϕ_1 et ϕ_2 sont deux formules en FNC, alors la formule $\phi_1 \wedge \phi_2$ l'est également (modulo l'élimination des répétitions de clauses), permet d'intégrer aisément de nouvelles connaissances à une base déjà existante. Reprenons par exemple les petits mondes du carrefour et de la thèse ; la base de connaissances relative au premier est représentée, entre autres, par la formule en FNC ϕ_C (voir les préliminaires généraux, paragraphe 3.2), et celle relative au second par la formule en FNC ϕ_t. Il est alors facile de voir que la formule $\phi_C \wedge \phi_t$ représente les connaissances relatives au petit monde réunissant celui du carrefour et celui de la thèse (c'est-à-dire le petit monde défini par l'ensemble de variables $V_C \cup V_t$), puisque intuitivement une situation est envisageable dans ce monde si et seulement si elle l'est dans les deux à la fois ; de fait, à l'inverse la *disjonction* $\phi_C \vee \phi_t$ de ces deux formules ne représente pas la base de connaissances relative à la réunion des deux petits mondes, puisqu'elle considère également comme envisageables les situations qui ne sont envisageables que dans l'un des deux. Notons que la conjonction de deux formules ϕ_1 et ϕ_2 admet pour ensemble de modèles exactement la *jointure*, au sens des bases de données relationnelles [AHV95, chapitre 8], des ensembles $\mathcal{M}(\phi_1)$ et $\mathcal{M}(\phi_2)$ vus comme des bases de données booléennes sur les ensembles d'attributs $Var(\phi_1)$ et $Var(\phi_2)$, respectivement.

La stabilité de la Forme Normale Conjonctive par conjonction permet donc l'intégration de nouvelles connaissances à d'anciennes tout en gardant le même langage de représentation. Notons que tous les langages de représentations mentionnés dans le paragraphe 1.1 contiennent des formules logiquement équivalentes à la conjonction de deux quelconques de leurs formules ; néanmoins, pour la plupart d'entre eux cette opération, même lorsqu'elle est de complexité polynomiale, nécessite un remaniement syntaxique de la connaissance initiale, tandis que pour la FNC elle est triviale. Nous renvoyons le lecteur à [DM02] pour la complexité de cette opération dans les divers langages de représentation.

Notre deuxième motivation pour considérer la Forme Normale Conjonctive comme un bon langage de représentation de connaissances est sa lisibilité. En effet, via l'équivalence logique, pour toutes formules propositionnelles ϕ_1 et ϕ_2, entre la disjonction $(\neg\phi_1) \vee \phi_2$ et l'implication $\phi_1 \rightarrow \phi_2$, d'une part, et entre la disjonction $(\neg\phi_1) \vee (\neg\phi_2)$ et la négation $\neg(\phi_1 \wedge \phi_2)$, d'autre part, on peut voir une clause comme un ensemble d'*implications* et d'*ensembles de variables interdits* : par exemple, on peut reformuler la formule en FNC ϕ_C représentant la base de connaissances du petit monde du carrefour (préliminaires généraux, paragraphe 3.2) comme suit :

$$\phi_C \equiv (x_{DT} \rightarrow x_{VP}) \wedge (x_{VP} \rightarrow x_{DT}) \wedge (\neg(x_{PV} \wedge x_{VV} \wedge x_{DT})) \wedge (\neg(x_{VP} \wedge x_{VV}))$$

Cette formule peut alors être lue de manière très intuitive :

– Si on a le droit de traverser, alors le feu pour piétons est vert, *et*

- Si le feu pour piétons est vert, alors on a le droit de traverser, *et*
- Il ne peut pas se produire en même temps qu'il y ait des véhicules aux abords du carrefour, que le feu pour véhicules soit vert et qu'on ait le droit de traverser, *et*
- Le feu pour piétons et le feu pour véhicules ne peuvent pas être verts en même temps.

Ainsi, la syntaxe des formules en Forme Normale Conjonctive est relativement intuitive, et lisible, pour l'être humain ; il nous est également relativement naturel d'exprimer ainsi nos connaissances par des ensembles de règles et d'ensembles interdits.

Plus généralement, la Forme Normale Conjonctive est un cas particulier de langage de *contraintes*, lesquels langages sont utilisés pour modéliser naturellement de nombreux problèmes concrets. Informellement, une contrainte est une fonction que l'on utilise pour *contraindre* un ensemble de variables (non nécessairement booléennes) ; ainsi, l'équation $x_1 = x_2$ peut être vue comme l'application de la contrainte «être égales» aux variables x_1 et x_2. Un réseau de contraintes est une conjonction de telles contraintes, toutes ne portant pas nécessairement sur les mêmes variables. Ainsi, la conjonction $(x_1 = x_2 + 7) \wedge (x_1 > x_3) \wedge (10 < x_4^2 < 20)$ est un réseau de contraintes sur l'ensemble de variables $\{x_1, x_2, x_3, x_4\}$. On peut donc voir une formule en Forme Normale Conjonctive comme un réseau de contraintes sur l'ensemble de ses variables, la seule contrainte autorisée étant la disjonction.

Notons dès maintenant que, outre des classes de formules en Forme Normale Conjonctive, nous étudions dans ce mémoire la classe des formules *affines* (présentées dans le chapitre 3), qui ne sont ni en FNC ni en FND, mais qui sont des cas particuliers, toujours propositionnels, de réseaux de contraintes, ce qui motive leur étude. Nous reviendrons sur nos motivations particulières pour étudier cette classe de formules au chapitre 3.

Forme Normale Disjonctive La Forme Normale Disjonctive n'est pas stable par conjonction dans le sens défini pour la Forme Normale Conjonctive. De plus, elle permet d'encoder les problèmes moins intuitivement, car les connaissances s'expriment en général moins naturellement comme des réunions d'ensembles de situations envisageables que comme des conjonctions de contraintes sur des variables. Néanmoins, ces formules permettent par exemple d'encoder facilement des préférences, ou des critères de sélection, et elles restent dans tous les cas relativement lisibles, comme le montre, par exemple, l'explicitation de la formule ϕ'_C du paragraphe 3.2 des préliminaires généraux.

Notre principale motivation pour étudier cette représentation est sa dualité avec la Forme Normale Conjonctive. Nous avons en effet vu que grâce à la transformation $\phi \mapsto Neg(\phi)$, on pouvait calculer facilement une formule en FND logiquement équivalente à la négation $\neg\phi$ d'une formule en FNC ϕ donnée. Par conséquent, si la base de connaissances relative à un petit monde est représentée par une formule FNC, la négation de cette base est implicitement représentée par une formule FND. On peut alors utiliser cette représentation si, par exemple, on doit raisonner à propos des états défectueux d'un système électrique, alors que notre base de connaissances représente ses états normaux, ou encore, pour reprendre l'exemple des préliminaires généraux (paragraphe 3.1), si l'on doit raisonner à propos des rayonnements ne faisant pas partie du domaine visible, alors que notre base de connaissances représente les exemples du concept «visible».

1.2 Classes de formules

Malheureusement, la plupart des problèmes algorithmiques qui se posent en Intelligence Artificielle sont très difficiles si l'on n'impose pas de restriction sur les représentations des bases de connaissances. Ainsi, nous avons vu que le problème `SAT` est NP-complet ; mais on sait aussi qu'il

le reste, même si on n'autorise comme données que des formules *en FNC* [Coo71]. Or, nous verrons par la suite que ce problème est fondamental en Intelligence Artificielle, et que sa traitabilité est une condition nécessaire à la traitabilité de nombreuses autres tâches, notamment de raisonnement (voir partie III). Similairement, nous verrons que l'*abduction* est un problème intraitable pour la classe des formules FND dans sa globalité. Nous considérons donc que la classe de *toutes* les formules FNC, ou celle de toutes les formules FND, ne sont pas intéressantes pour représenter des connaissances, car elles ne permettent pas en général d'utiliser ensuite efficacement ces connaissances.

Nous devons donc rechercher des *compromis* entre l'expressivité des langages de représentation de connaissances et leur traitabilité pour des problèmes d'Intelligence Artificielle. C'est pourquoi nous nous intéressons à des restrictions supplémentaires sur les Formes Normales Conjonctive et Disjonctive. Nous perdrons donc leur *expressivité*, puisque toute base de connaissances relative à un petit monde peut être représentée en FNC ou en FND, alors que ce ne sera pas le cas pour les sous-langages présentés dans cette partie. En revanche, nous gagnerons en *traitabilité*, en ce sens que des tâches non traitables pour la classe entière des formules en FNC (resp. en FND) deviendront traitables pour ces sous-langages. La partie II abordera le problème de représenter, ou d'*acquérir*, des connaissances quelconques dans un tel sous-langage alors qu'il n'est pas à même d'exprimer n'importe quelle connaissance.

Le problème `SAT` étant fondamental pour les problématiques d'Intelligence Artificielle, puisque sa traitabilité pour une classe de formules est la plupart du temps nécessaire pour la traitabilité des autres tâches de raisonnement, nous choisissons d'écarter tout d'abord les classes de formules pour lesquelles il n'est pas traitable (nous noterons `SAT`$[\mathcal{C}]$ le problème `SAT` dont on restreint les données aux formules de la classe $\mathcal{C}$).

Une autre propriété importante d'une sous-classe de formules est sa *reconnaissabilité* : nous dirons d'une classe de formules qu'elle est *reconnaissable* s'il existe un algorithme polynomial permettant de décider si une formule propositionnelle donnée en fait partie. Les restrictions sur les représentations des formules imposées au paragraphe 2.4 des préliminaires généraux assurent qu'essentiellement, une lecture de la formule suffit pour déterminer si elle est en FNC ou en FND, mais reconnaître si elle appartient à une classe plus précise n'est pas toujours facile ; ainsi, le problème de reconnaître les formules FNC renommables en une formule de Horn définie positive est NP-complet (chapitre 4).

Classes de formules en FNC Pour ce qui est des classes de formules en FNC, nous choisissons également d'étudier des classes de formules *stables par conjonction*, comme l'est la classe des formules en FNC dans son ensemble ; nos motivations sont les mêmes que celles exposées dans le paragraphe 1.1 concernant l'intégration de nouvelles connaissances à une base préexistante. Plus généralement, nous étudierons des classes de formules en FNC définies par la *nature* de leurs clauses, et non par leur *structure*. Cette distinction, qui provient de la littérature sur les *Problèmes de Satisfaction de Contraintes*, oppose deux types de classes de formules :

- les classes définies par des propriétés *locales* : ce sont les classes de formules en FNC $\mathcal{C}$ telles que par définition, une formule en FNC appartient à $\mathcal{C}$ si et seulement si chacune de ses clauses vérifie une certaine propriété (est d'une certaine *nature*) ; par exemple, la classe des formules $3FNC$, c'est-à-dire la classe des formules en FNC dont chaque clause contient au plus trois littéraux, est définie par la nature des clauses de ses formules. On parle également de classes de formules *clausales* [dV96]
- les classes définies par des propriétés *globales*, c'est-à-dire les classes $\mathcal{C}$ telles qu'une formule en FNC appartient à $\mathcal{C}$ si et seulement l'*ensemble* de ses clauses vérifie une certaine propriété

(possède une certaine *structure*) ; par exemple, la classe des formules contenant moins de k clauses, pour une constante k, est définie par la structure de ses formules, et non par leur nature.

Si la classe de formules FNC choisie pour représenter des connaissances est définie par la *nature* de ses clauses, on peut donc intégrer de nouvelles connaissances à une base clause par clause, en n'ayant à vérifier que la nature de la clause ajoutée. En revanche, si cette classe est définie par la *structure* de ses formules, chaque ajout de connaissances demande la vérification de la structure de *toute* la nouvelle base de connaissances ainsi constituée. Notons également qu'une classe définie en nature est par définition stable par conjonction. Cette propriété peut également être vérifiée par une classe définie en structure, comme c'est le cas pour la classe des formules en FNC contenant *au moins* k clauses, pour une constante k, mais en général ce n'est pas le cas.

Pour résumer nos critères, nous nous intéressons à des langages de représentation qui permettent à un système à base de connaissances de garantir à l'avance, si on ne lui ajoute que des connaissances satisfaisant certaines contraintes relatives à la *nature* des clauses, que le raisonnement avec sa base de connaissances, ou avec sa «négation», sera efficace. Notons que c'est par exemple le principe du langage *Prolog*, au premier ordre (voir [BCZ01, chapitre 3.1] par exemple) ; ce langage garantit en effet une procédure de raisonnement «efficace» en n'autorisant que des clauses de Horn.

Nous étudierons malgré tout quelques classes de formules en FNC définies en *structure* et non stables par conjonction, les classes des formules *renommables* dans $\mathcal{C}$, lorsque $\mathcal{C}$ est l'une des classes précédentes. En effet, malgré les moins bonnes propriétés de ces classes pour la représentation et l'intégration de connaissances, elles possèdent en général de bonnes propriétés pour les tâches d'Intelligence Artificielle, propriétés fortement liées à celles des classes précédentes ; la plupart du temps, elles généralisent ces classes tout en conservant leurs bonnes propriétés algorithmiques. Des motivations plus riches pour étudier ces classes seront présentées dans le chapitre 4.

Nous considérons également comme une bonne propriété d'une classe de formules $\mathcal{C}$ sa *stabilité par propagation d'affectations* (voir les préliminaires généraux, paragraphe 1.4). Outre ses conséquences bénéfiques sur l'algorithmique de la classe, cette propriété permet de restreindre une base de connaissances relative à un petit monde en fixant des variables, tout en préservant son appartenance à la classe. Par exemple, on peut vouloir restreindre la base de connaissances de l'exemple du carrefour au cas où le feu pour piétons est vert, ce qui se traduit sur la représentation en FNC ϕ_C de cette base par la propagation de l'affectation de 1 à la variable x_{VP}, donnant la formule :

$$(x_{DT}) \wedge (\neg x_{PV} \vee \neg x_{VV} \vee \neg x_{DT}) \wedge (\neg x_{VV})$$

On retrouve cette forme de restrictions dans le cadre du «reasoning within context» de Khardon et Roth [KR94].

Enfin, rappelons que notre étude porte également sur les formules *affines*, bien que ces formules ne soient ni en FNC, ni en FND. Ce choix, quant à lui, sera longuement motivé dans le chapitre 3.

Classes de formules en FND Pour ce qui est des sous-classes de formules en FND, la traitabilité du problème `SAT` est assurée, puisqu'elle l'est pour la classe de *toutes* les formules en FND. Motivé par la discussion du paragraphe 1.1, nous choisissons cependant d'étudier des *sous-classes* de telles formules. Puisque nous considérons les formules en FND dans l'optique de représenter implicitement la *négation* de bases de connaissances, notre étude concerne les «négations» des sous-classes de formules en FNC qui nous intéressent. Ainsi, pour chaque classe

$\mathcal{C}$ de formules en FNC que nous choisissons d'étudier, nous étudions également la classe de formules en FND définie comme la classe des formules ϕ telles que la formule en FNC $Neg(\phi)$ est dans $\mathcal{C}$.

1.3 Classes étudiées dans le mémoire

Les critères présentés dans le paragraphe 1.2 nous amènent à nous intéresser dans ce mémoire aux classes de formules suivantes :

- formules FNC de Horn, négatives et Horn définies positives
- formules FNC anti-Horn, positives et anti-Horn définies positives
- formules FNC bijonctives (2FNC)
- formules affines
- formules Horn-renommables et Horn définies positives renommables
- formules monotones (négatives-renommables)
- pour chacune des classes $\mathcal{C}$ précédentes, la classe des formules ϕ en FND telles que $Neg(\phi)$ est une formule de $\mathcal{C}$.

Les classes des formules FNC (anti-)Horn, (anti-)Horn définies positives, négatives, positives et bijonctives satisfont tous les critères mentionnés dans le paragraphe 1.2 (définition en nature, stabilité par conjonction, traitabilité pour `SAT` et reconnaissabilité), ainsi que les formules affines, même si ces formules ne sont ni en FNC, ni en FND. Le choix des autres classes est déjà motivé dans le paragraphe 1.2. De plus, l'étude des formules FNC (anti-)Horn, (anti-)Horn définies positives, négatives, positives, bijonctives et affines est motivée par un résultat de dichotomie pour une variante du problème `SAT` [Sch78], et par un résultat similaire pour le problème `Inv-SAT[`S`]` [KS98], montrant que ces classes (et leurs sous-classes) sont essentiellement les seules définies en nature et traitables ; nous reviendrons sur ces deux résultats au chapitre 5.

La partie est organisée comme suit. Nous présentons tout d'abord les classes de formules FNC qui nous intéressent et qui sont définies en nature (chapitre 2), puis la classe des formules affines (chapitre 3). Nous présentons ensuite les autres classes de formules qui nous intéressent, les classes de formules $\mathcal{C}$-*renommables* et les classes de formules FND (chapitre 4), et enfin nous résumons les propriétés de toutes ces classes, les considérons les unes par rapport aux autres et donnons des motivations supplémentaires pour nos choix, à la lueur des définitions (chapitre 5).

Pour terminer, précisons que le but de cette partie n'est pas tant de donner des résultats nouveaux que de synthétiser et de discuter les propriétés connues des classes de formules, et ce de manière uniforme.

Chapitre 2

Bonnes formules FNC

Sommaire

2.1 Formules FNC de Horn 31
2.2 Formules FNC de Horn définies positives 34
2.3 Formules FNC négatives 36
2.4 Formules 2FNC 38
2.5 Classes duales 40

Nous commençons par présenter les classes de formules FNC définies par la nature de leurs clauses, et donc stables par conjonction. Pour toutes ces classes, nous exhibons les propriétés syntaxiques, algorithmiques et sémantiques qui nous semblent utiles. Les classes présentées ici possédant toutes, à quelques détails près, les mêmes propriétés, nous nous attardons plus longuement sur celle des formules de Horn, puis nous présentons les autres plus laconiquement.

2.1 Formules FNC de Horn

La classe des formules FNC de Horn, que nous notons HORN, est certainement la plus classique de toutes. Intuitivement, les formules de Horn sont des ensembles de règles que l'on peut utiliser pour déduire des littéraux positifs (faits) à partir d'autres littéraux positifs, processus souvent appelé «chaînage avant» [HK93, HK95], sans jamais avoir à essayer deux possibilités différentes ; en revanche, contrairement aux formules de Horn définies positives que nous présenterons plus loin, on peut être confronté lors d'un tel processus à un ensemble de variables interdits. En effet, les formules de Horn sont définies par la propriété syntaxique suivante.

Définition 2.1 (formule FNC de Horn) *Une* clause de Horn *est une clause dont au plus un littéral est positif. Une* formule FNC de Horn *est une conjonction de clauses de Horn.*

Notons que certains auteurs réservent l'appellation *clause de Horn* aux clauses dont *exactement* un littéral est positif, et dénomment *clauses négatives* celles dont tous les littéraux sont négatifs. Nous préférons cependant unifier les dénominations pour les clauses et les formules afin d'éviter les confusions.

On peut donc voir une clause de Horn comme, selon les cas, une *règle*, logiquement équivalente à une implication sans disjonction en partie droite, de la forme $((x_{i_1} \wedge x_{i_2} \wedge \cdots \wedge x_{i_k}) \rightarrow x_j)$, ou un ensemble de variables interdits (cf. chapitre 1, paragraphe 1.1). Ainsi, la formule représentant

la base de connaissances de l'exemple du carrefour :

$$\phi_C = (\neg x_{DT} \vee x_{VP}) \wedge (\neg x_{VP} \vee x_{DT}) \wedge (\neg x_{PV} \vee \neg x_{VV} \vee \neg x_{DT}) \wedge (\neg x_{VP} \vee \neg x_{VV})$$

est une formule FNC de Horn.

Parmi les clauses de Horn, on peut remarquer les clauses *unitaires*, qui peuvent être positives ou négatives : ce sont les clauses ne contenant qu'un littéral (positif ou négatif, respectivement). Remarquons également que la clause vide () est une clause de Horn, et donc que la formule en FNC insatisfaisable réduite à cette clause, (()), est de Horn. Enfin, il est facile de voir que la conjonction de deux formules en FNC de Horn est de Horn, d'une part, et que cette classe est stable par propagation d'affectations, d'autre part.

Les bonnes propriétés algorithmiques des formules de Horn, quant à elles, sont nombreuses. Remarquons tout d'abord qu'il est facile de décider si une formule FNC donnée est de Horn ; il suffit de lire la formule et de compter le nombre de littéraux positifs par clause. L'autre propriété fondamentale de cette classe de formules est sa traitabilité pour le problème SAT ; nous donnons simplement l'idée de l'algorithme dans la preuve de la proposition suivante.

Proposition 2.1 (SAT[HORN] [DG84]) *On peut décider si une formule FNC de Horn donnée est satisfaisable en temps linéaire, en en exhibant un modèle le cas échéant.*

Idée de la preuve L'algorithme consiste à construire une affectation aux variables à partir des clauses unitaires de la formule. Supposons que cette dernière contient une telle clause C, et supposons par symétrie $C = (x)$, où x est une variable. On augmente alors l'affectation courante en affectant 1 à x, en supprimant C de la formule et en y propageant la nouvelle affectation ; si C est de la forme $(\neg x)$, on affecte 0 à x. Si en propageant l'affectation on obtient la clause vide, alors la formule est insatisfaisable, et sinon on recommence tant qu'elle contient au moins une clause unitaire. Lorsque la formule ne contient plus de telle clause, alors elle ne contient que des clauses de longueur 2 ou plus, qui contiennent donc chacune au moins un littéral négatif. Il suffit alors d'affecter 0 à toutes les variables restantes pour obtenir un modèle de la formule initiale.

Des algorithmes linéaires sont donnés par Dowling et Gallier [DG84] ou par Minoux [Min88]. □

Mais au-delà de ses propriétés syntaxiques et algorithmiques, la classe HORN possède également des propriétés *sémantiques* très fortes. Celle que nous utiliserons le plus souvent est la suivante ; pour deux affectations m, m' à un même ensemble de variables V, nous notons $m \wedge m'$ l'affectation à V définie par $\forall x \in V, (m \wedge m')[x] = m[x] \wedge m'[x]$; rappelons qu'un ensemble d'affectations (à un même ensemble de variables) est dit *descriptible* dans une classe de formules $\mathcal{C}$ s'il est l'ensemble des modèles d'au moins une formule de $\mathcal{C}$.

Proposition 2.2 (HORN-clôture [McK43, DP92]) *Un ensemble d'affectations M est descriptible dans* HORN *si et seulement si pour tous $m, m' \in M$, on a $m \wedge m' \in M$.*

Preuve Supposons que M est descriptible dans HORN, et soit $\phi \in$ HORN avec $M = \mathcal{M}(\phi)$. Soient $m, m' \in M$; nous montrons que l'affectation $m \wedge m'$ satisfait toutes les clauses de ϕ. Soit donc C une telle clause ; on a donc $m \models C$ et $m' \models C$. Si C ne contient que des littéraux négatifs, puisque m satisfait C il existe au moins une variable $x \in Var(C)$ avec $\neg x \in C$ et $m[x] = 0$; or 0 est absorbant pour l'opération $\wedge$, donc on a $(m \wedge m')[x] = 0$ et donc $m \wedge m' \models C$. Si C contient un littéral positif, nous distinguons deux cas. S'il existe une variable $x \in Var(C)$ avec $\neg x \in C$ et, soit $m[x] = 0$, soit $m'[x] = 0$, on est ramené au cas précédent. Supposons donc $\forall x \in Var(C), \neg x \in C \Longrightarrow m[x] = m'[x] = 0$. Alors, puisque m et m' satisfont C, en notant $x \in$

$Var(C)$ le littéral positif de C on a $m[x] = m'[x] = 1$; on a donc $(m \wedge m')[x] = m[x] \wedge m'[x] = 1$, donc l'affectation $m \wedge m'$ satisfait C. Finalement, on a $m \wedge m' \models C$ pour toute clause C de ϕ, donc $m \wedge m' \models \phi$ et donc, $m \wedge m' \in \mathcal{M}(\phi) = M$.

Pour la réciproque, nous renvoyons à [DP92], ou au chapitre 7 de ce mémoire, où sont données des constructions (la construction du chapitre 7 dépend seulement du sens direct de l'énoncé, via la proposition 2.3). □

Pour un ensemble M, quelconque, d'affectations à un même ensemble de variables V, nous définissons donc la HORN-*clôture* de M comme l'ensemble d'affectations à V :

$$Cl_{\text{HORN}}(M) = \{m \wedge m' \mid m, m' \in M\}$$

Par exemple, la HORN-clôture de l'ensemble d'affectations :

$$M_t = \{00111, 01000, 10100, 11011\}$$

de l'exemple de la thèse est l'ensemble d'affectations :

$$Cl_{\text{HORN}}(M_t) = \{00111, 01000, 10100, 11011, 00000, 00100, 00011, 10000\}$$

au même ensemble de variables.

Ainsi, un ensemble d'affectations M à V est l'ensemble de modèles d'au moins une formule de Horn si et seulement si on a $Cl_{\text{HORN}}(M) = M$. On peut ainsi vérifier que l'ensemble d'affectations M_C représentant la base de connaissances de l'exemple du carrefour est bien descriptible dans HORN (par exemple par ϕ_C). En revanche, celui représentant la connaissance du petit monde de la thèse, M_t, ne vérifie pas cette propriété, puisque l'on a $M_t \neq Cl_{\text{HORN}}(M_t)$. De fait, non seulement la formule ϕ_t (paragraphe 3.3 des préliminaires généraux) n'est pas en FNC de Horn, mais il n'existe *aucune* formule en FNC de Horn sur le même ensemble de variables et pouvant représenter la même connaissance, c'est-à-dire logiquement équivalente à ϕ_t.

Cette propriété sémantique, enfin, permet d'obtenir une caractérisation très importante de cette classe en termes de formules FNC *premières* ; notons cependant que cette propriété peut également être obtenue en raisonnant sur la *syntaxe* de ces formules, comme le font Hammer et Kogan [HK92].

Proposition 2.3 (FNC première de Horn [HK92]) *Un ensemble d'affectations M est descriptible dans* HORN *si et seulement si toute formule en FNC et première décrivant M est de Horn.*

Preuve Si toute formule en FNC et première décrivant M est de Horn, tout d'abord, alors par définition de la descriptibilité M est descriptible dans HORN. Supposons donc, réciproquement, que M est descriptible dans HORN. Alors (proposition 2.2) on a $\forall m, m' \in M, m \wedge m' \in M$; nous montrons que tout impliqué premier d'une description de M est une clause de Horn. Soit C une telle clause, et soient par l'absurde $x, x' \in Var(C)$ deux littéraux positifs différents de C. Puisque C est première, il existe une affectation $m \in M$ (resp. $m' \in M$) qui ne satisfait pas la clause $C \backslash \{x\}$ (resp. $C \backslash \{x'\}$). Donc m affecte 1 à x, mais ne satisfait aucun autre littéral de C ; en particulier, on a $m[x'] = 0$, et symétriquement on a $m'[x'] = 1$, mais m' ne satisfait aucun autre littéral de C. On en déduit que l'affectation $m \wedge m'$ affecte 0 à x et x', et ne satisfait aucun littéral de C formé sur une variable différente de x et x', donc qu'elle ne satisfait pas C, ce qui est absurde puisqu'elle est dans M. Finalement, tout impliqué premier d'une description de M est de Horn, et donc toute description de M en FNC et première est de Horn. □

Ainsi, il est facile de vérifier que la formule en FNC ϕ_t du petit monde de la thèse est première, et puisqu'elle n'est pas de Horn on peut à nouveau déduire qu'il n'existe aucune formule de Horn logiquement équivalente à ϕ_t.

Les formules FNC de Horn sont certainement les formules de la logique propositionnelle les plus étudiées. Citons par exemple les travaux de Hammer et Kogan [HK92, HK93, HK94, HK95] sur la minimisation de ces formules, ainsi que ceux d'Ibaraki et Makino [EIM98b, IKM99]. La notion de clause de Horn est également à la base des langages PROLOG [RK91, chapitre 6] ou DATALOG [AHV95, chapitre 12]. Enfin, on la retrouve pour des formules bâties sur des variables *multivaluées* plutôt que booléennes [BHM99].

2.2 Formules FNC de Horn définies positives

Nous présentons maintenant la classe des formules FNC de Horn définies positives, que nous notons HDP. Cette classe est une sous-classe de HORN, celle des formules ne contenant pas d'ensembles de variables interdits. Intuitivement, avec une formule de la classe HDP on peut donc déduire des faits à partir d'autres faits sans jamais avoir à essayer deux possibilités différentes, comme pour HORN, mais sans jamais non plus être confronté à un ensemble de variables interdits (voir par exemple les travaux d'Hammer et Kogan [HK93, HK95]). Précisément, ces formules sont définies comme suit.

Définition 2.2 (formule FNC de Horn définie positive) *Une* clause de Horn définie positive *est une clause dont* exactement *un littéral est positif. Une* formule FNC de Horn définie positive *est une conjonction de clauses de Horn définies positives.*

Par exemple, la formule en FNC de l'exemple du carrefour :

$$\phi_C = (\neg x_{DT} \vee x_{VP}) \wedge (\neg x_{VP} \vee x_{DT}) \wedge (\neg x_{PV} \vee \neg x_{VV} \vee \neg x_{DT}) \wedge (\neg x_{VP} \vee \neg x_{VV})$$

dont nous avons vu dans le paragraphe 2.1 qu'elle est de Horn, n'est en revanche pas de Horn *définie positive*, puisque la dernière de ses clauses, $(\neg x_{VP} \vee \neg x_{VV})$, ne contient aucun littéral positif. Pour ce qui est de l'exemple de la thèse, nous avons vu (paragraphe 2.1) qu'aucune formule FNC de Horn ne pouvait représenter la base de connaissances correspondante, et donc *a fortiori* aucune formule FNC de Horn définie positive.

Bien que la classe plus générale des formules de Horn possède, dans toute sa généralité, de nombreuses bonnes propriétés algorithmiques dont hérite la classe des formules de Horn définies positives, cette dernière classe mérite attention ; en effet, nous verrons par exemple au chapitre 11 que l'on peut faire du raisonnement abductif efficace avec les formules de Horn définies positives, mais pas avec les formules de Horn en général. Comme HORN, la classe HDP est stable par conjonction, mais seules les clauses unitaires *positives* sont de Horn définies positives ; ni la clause vide, ni les clauses unitaires *négatives* ne le sont. Enfin, cette classe n'est en général stable que par propagation d'affectations *positives*, c'est-à-dire d'affectations de la forme $\overline{1_V}$ pour un certain ensemble de variables V ; en effet, la propagation de l'affectation de 0 à x_1 dans la formule FNC $((x_1 \vee \neg x_2)) \in$ HDP, par exemple, donne la formule $((\neg x_2)) \notin$ HDP.

Remarquons encore que, comme pour HORN, la reconnaissance des formules FNC de Horn définies positives est linéaire ; il suffit d'un parcours de la formule. En revanche, le problème `SAT` est encore plus facile pour la classe HDP que pour HORN ; la proposition suivante découle directement de la présence d'un littéral positif par clause de Horn définie positive.

Proposition 2.4 (SAT[HDP]) *Toute formule FNC de Horn définie positive ϕ admet $\overline{1_{Var(\phi)}}$ comme modèle.*

Enfin, nous terminons ce paragraphe par les propriétés *sémantiques* de la classe HDP. Elles sont fortement liées à celles de la classe HORN, en vertu des deux propositions suivantes.

Proposition 2.5 (HORN vs. HDP, affectations) *Un ensemble d'affectations M à un même ensemble de variables V descriptible dans* HORN *est descriptible dans* HDP *si et seulement si l'affectation $\overline{1_V}$ est dans M.*

Preuve La proposition 2.4 montre l'implication dans le sens direct. Supposons maintenant $\overline{1_V} \in M$. Puisque M est descriptible dans HORN, il existe une formule FNC de Horn ϕ décrivant M, et comme $\overline{1_V}$ est dans M, c'est un modèle de ϕ, et on en déduit que ϕ ne contient pas de clause négative. Donc ϕ est une formule FNC de Horn définie positive. □

Proposition 2.6 (HORN vs. HDP, formules) *Soit ϕ une formule FNC de Horn. Alors il existe une formule FNC de Horn définie positive logiquement équivalente à ϕ si et seulement si ϕ elle-même est Horn définie positive.*

Preuve S'il existe une formule de HDP ϕ' logiquement équivalente à ϕ, alors la proposition 2.4 nous dit que l'affectation $\overline{1_{Var(\phi')}}$ satisfait ϕ', et donc que tous ses prolongements à $Var(\phi') \cup Var(\phi)$ satisfont ϕ. En particulier, l'affectation $\overline{1_{Var(\phi) \cup Var(\phi')}}$ satisfait ϕ, donc ϕ ne contient pas de clause négative, et comme elle est de Horn elle est de Horn définie positive. La réciproque est triviale. □

En d'autres termes, la proposition 2.6 nous dit qu'on ne peut pas reformuler une formule de HDP en une formule FNC de Horn *non* définie positive logiquement équivalente. Plus généralement, on peut même voir avec la même preuve que l'on ne peut pas reformuler une telle formule en une formule FNC dont une des clauses ne contient que des littéraux négatifs. Ainsi, pour le petit monde du carrefour, le fait que la formule FNC ϕ_C contienne la clause négative $(\neg x_{VP} \vee \neg x_{VV})$, par exemple, montre qu'il n'existe aucune formule de HDP logiquement équivalente à ϕ_C.

Nous terminons ce paragraphe en précisant le parallèle des propositions 2.2 et 2.3 pour la classe HDP. La proposition suivante, tout d'abord, est un simple corollaire des propositions 2.2 et 2.5.

Proposition 2.7 (HDP-clôture) *Un ensemble d'affectations M à un même ensemble de variables V est descriptible dans* HDP *si et seulement si on a $\overline{1_V} \in M$ et $\forall m, m' \in M, m \wedge m' \in M$.*

Nous définissons donc la HDP-*clôture* d'un ensemble d'affectations M, quelconque, à un ensemble de variables V comme l'ensemble :

$$Cl_{\text{HDP}}(M) = Cl_{\text{HORN}}(M) \cup \{\overline{1_V}\}$$

Le parallèle de la proposition 2.3, enfin, est également un corollaire des résultats précédents.

Proposition 2.8 (FNC première de Horn définie positive) *Soit M un ensemble d'affectations. Alors M est descriptible dans* HDP *si et seulement si toute formule en FNC et première décrivant M est de Horn définie positive.*

Preuve Si M est descriptible dans HDP, alors en particulier il est descriptible dans HORN, donc (proposition 2.3) toute formule en FNC et première le décrivant est de Horn, et par la proposition 2.6 de Horn définie positive. La réciproque est triviale. □

2.3 Formules FNC négatives

La classe des formules négatives, que nous notons NÉG, est également une sous-classe de la classe des formules de Horn, par définition disjointe de son autre sous-classe HDP si on excepte les formules tautologiques, mais comme celle-ci elle mérite une attention particulière car, par exemple, l'abduction est traitable pour les formules de cette classe, mais pas pour les formules de Horn (voir chapitre 11). Les formules de Horn définies positives sont les formules de Horn sans clause négative ; les formules négatives sont les formules de Horn sans clause de Horn définie positive.

Définition 2.3 (formule FNC négative) *Une* clause négative *est une clause dont tous les littéraux sont négatifs. Une* formule FNC négative *est une conjonction de clauses négatives.*

On peut donc voir une formule en FNC négative comme une formule dont la satisfaisabilité est une fonction *décroissante* en chacune des variables, en ce sens que, informellement, si une affectation m est un modèle d'une formule FNC négative, en remplaçant dans m des affectations à 1 par des affectations à 0 on est assuré que l'affectation obtenue sera toujours un modèle de la formule. Ainsi, une formule FNC négative représente par exemple naturellement le concept de la défaite au tennis, si l'on décrit ce petit monde via, pour chacun des (trois) sets d'une partie, une variable x_i indiquant si le set a été gagné ; en effet, la formule FNC négative :

$$\phi_{tennis} = (\neg x_1 \vee \neg x_2) \wedge (\neg x_1 \vee \neg x_3) \wedge (\neg x_2 \vee \neg x_3)$$

est satisfaite par exactement les affectations à $\{x_1, x_2, x_3\}$ qui représentent des exemples de défaite (pour simplifier l'exemple, nous considérons que les trois sets d'une partie sont toujours joués). Intuitivement, le fait que cette formule soit négative correspond au fait que moins l'on gagne de sets (plus nombreuses sont les variables affectées à 0), plus l'on perd la partie (plus la formule est satisfaite).

On remarque que les clauses unitaires *négatives*, évidemment, mais aussi la clause vide sont des clauses négatives, et que la classe des formules FNC négatives est stable par conjonction. Cette classe est également stable par propagation d'affectations *quelconques*. Mais remarquons également qu'il existe toujours une formule FNC négative logiquement équivalente à la *disjonction* de deux formules FNC négatives ; en effet, si ϕ et ϕ' sont deux formules négatives, alors elles ne contiennent que des littéraux négatifs, et donc la formule FNC ϕ'', logiquement équivalente à la formule $\phi \vee \phi'$ et obtenue en y distribuant la disjonction sur la conjonction, ne contient elle aussi que des littéraux négatifs.

Enfin, comme pour la classe HDP, la reconnaissance syntaxique est triviale, ainsi que le problème `SAT`.

Proposition 2.9 (SAT[NÉG]) *Soit ϕ une formule FNC négative. Si ϕ contient une clause vide, alors elle est insatisfaisable, sinon elle admet $\overline{0_{Var(\phi)}}$ comme modèle.*

En revanche, le lien sémantique avec la classe HORN est moins évident que pour HDP. On dit qu'une affectation m' à un ensemble de variables V est *inférieure* à une affectation m aux mêmes variables, et on note $m' \preceq m$, si on a $\forall x \in V, m[x] = 0 \Longrightarrow m'[x] = 0$.

Proposition 2.10 (NÉG-clôture) *Un ensemble d'affectations M à un même ensemble de variables V est descriptible dans* NÉG *si et seulement si pour tout $m \in M$ et pour toute affectation m' à V vérifiant $m' \preceq m$, on a $m' \in M$.*

Preuve Si M est descriptible dans NÉG, alors il existe une formule FNC négative ϕ le décrivant. Soit $m \in M$ une affectation ; par définition m satisfait ϕ, et donc satisfait au moins un littéral (négatif) par clause de ϕ, et comme pour toute affectation m' à V avec $m' \preceq m$ on a par définition $\forall x \in V, m[x] = 0 \Longrightarrow m'[x] = 0$, m' satisfait au moins les mêmes littéraux négatifs de ϕ que m, donc m' satisfait ϕ et par conséquent m' est dans M.

Réciproquement, si on a $\forall m \in M, m' \in \{0,1\}^V, m' \preceq m \Longrightarrow m' \in M$, on peut construire une formule FNC négative décrivant M comme suit. Soit $Max(M)$ l'ensemble des affectations maximales de M (pour l'ordre partiel $\prec$), soit ϕ la formule définie par :

$$\phi = \bigvee_{m \in Max(M)} \left(\bigwedge_{x \in V, m[x]=0} \neg x \right)$$

Il est facile de voir que l'ensemble des modèles de ϕ est exactement M. Soit alors ϕ' la formule FNC obtenue en distribuant dans ϕ la disjonction sur la conjonction ; par construction ϕ' est logiquement équivalente à ϕ, et comme elle est en FNC négative on a le résultat. □

Nous définissons donc la NÉG-*clôture* d'un ensemble d'affectations M, quelconque, à un même ensemble de variables comme l'ensemble d'affectations

$$Cl_{\text{NÉG}}(M) = \{m' \mid \exists m \in M, m' \preceq m\}$$

Ainsi, on voit que la base de connaissances de l'exemple du carrefour ne peut pas être représentée par une formule négative, puisque l'on a :

$$0101 \in M_C, 0001 \preceq 0101 \text{ mais } 0001 \notin M_C$$

Pour terminer, la proposition suivante caractérise les connaissances que l'on peut représenter par une formule de NÉG en termes de formules FNC premières. Remarquons que cette proposition est beaucoup plus forte que la proposition correspondante pour les classes HORN et HDP.

Proposition 2.11 (FNC première négative) *Un ensemble d'affectations M à un ensemble de variables V est descriptible dans* NÉG *si et seulement s'il existe une formule première négative décrivant M ; dans ce cas, cette formule est la seule formule première décrivant M, et elle contient tous ses impliqués premiers.*

Preuve Le sens indirect découle encore une fois de la définition même de la descriptibilité. Nous montrons donc le sens direct.

[existence] Nous montrons tout d'abord que si M est descriptible dans NÉG, alors il l'est par une formule *première*. Comme pour HORN (proposition 2.3), soit C un impliqué premier d'une description de ϕ, et supposons par l'absurde que C contient un littéral positif x. Puisque C est première, il existe une affectation $m \in M$ qui ne satisfait pas la clause $C \backslash \{x\}$, mais comme m est dans M elle satisfait C. On en déduit que le seul littéral de C satisfait par m est le littéral

x, et donc $m[x] = 1$. Soit alors m' l'affectation à V définie par $m'[x] = 0$ et $\forall x' \in V, x' \neq x \Longrightarrow m'[x] = m[x]$; on a par construction $m' \prec m$ mais $m' \not\models C$, donc $m' \notin M$, ce qui contredit la proposition 2.10.

[unicité et impliqués premiers] Pour montrer que la formule FNC première décrivant M est unique, il suffit d'utiliser le théorème de Quine (préliminaires généraux, page 14). Soit ϕ une formule FNC première décrivant M. Puisque ϕ est négative, on ne peut pas trouver deux de ses clauses qui soient résolubles. Le théorème de Quine nous dit donc que ϕ contient tous ses impliqués premiers. Si ϕ' est une autre formule FNC décrivant M, et C une clause de ϕ' non présente dans ϕ, alors C peut être minimisée en une clause première, donc en une clause de ϕ, et de fait C n'est pas première, donc ϕ n'est pas première. Si à l'inverse il existe une clause C dans ϕ non présente dans ϕ', alors ϕ' ne contient pas tous ses impliqués premiers, donc ϕ' n'est pas première. Finalement, il n'existe pas de formule FNC première différente de ϕ et décrivant M. □

On voit ainsi aisément que la formule FNC négative ϕ_{tennis} du début de ce paragraphe est la seule formule FNC première représentant le concept de la défaite au tennis (sur l'ensemble de variables $\{x_1, x_2, x_3\}$).

2.4 Formules 2FNC

Nous présentons maintenant la classe BIJ des formules *2FNC*, aussi appelées formules FNC *bijonctives*. Contrairement aux deux classes précédentes, cette classe n'est pas incluse dans celle des formules de Horn ; elle n'en possède pas moins d'aussi bonnes propriétés algorithmiques. Informellement, c'est la classe des formules qui ne lient les variables que deux par deux.

Définition 2.4 (formule 2FNC) *Une* 2clause*, ou* clause bijonctive*, est une clause contenant deux littéraux ou moins. Une* formule 2FNC*, ou* formule FNC bijonctive*, est une conjonction de* 2*clauses.*

Remarquons que les 2clauses sont parfois appelées clauses *quadratiques* [HK95].

On voit donc que les formules FNC ϕ_C et ϕ_t des exemples du carrefour et de la thèse ne sont pas en 2FNC, mais qu'en revanche la formule ϕ_{tennis} du paragraphe 2.3 l'est.

Remarquons tout de suite que les clauses unitaires ainsi que la clause vide sont des 2clauses, et que la classe BIJ est stable par conjonction comme par propagation d'affectations quelconques. Remarquons également que la reconnaissance syntaxique de ces formules est triviale.

On peut voir une 2clause $(\ell \vee \ell')$ comme l'implication $(\overline{\ell} \rightarrow \ell')$, ou encore comme sa contraposée $(\overline{\ell'} \rightarrow \ell)$; si la clause ne contient qu'un littéral ℓ, on peut la voir comme l'implication $(\overline{\ell} \rightarrow 0)$, ou $(1 \rightarrow \ell)$. Ainsi, en supposant une formule 2FNC donnée ϕ ne contenant pas la clause vide, et en introduisant les littéraux constants 0 et 1, on peut voir ϕ comme un *graphe d'implications*, c'est-à-dire un graphe orienté dont les sommets sont les littéraux formés sur $Var(\phi)$, et dont deux sommets sont liés par un arc (ℓ, ℓ') si et seulement si la formule contient la clause $(\overline{\ell} \vee \ell')$ ou, de façon équivalente, l'implication $(\ell \rightarrow \ell')$; notons que pour deux littéraux ℓ, ℓ', on a l'arc (ℓ, ℓ') dans ce graphe si et seulement si on a l'arc $(\overline{\ell'}, \overline{\ell})$. Cette construction permet de montrer la proposition suivante.

Proposition 2.12 (SAT[BIJ] [APT79]) *On peut décider si une formule 2FNC donnée ϕ est satisfaisable en temps linéaire, en en exhibant un modèle le cas échéant.*

Idée de la preuve L'idée est de calculer les *composantes fortement connexes* du graphe d'implications de ϕ (qui peut être construit en temps linéaire de façon directe), c'est-à-dire les sous-ensembles CFC maximaux de sommets du graphe tels que pour tous sommets ℓ, ℓ' de CFC il existe un chemin de ℓ vers ℓ' et un chemin de ℓ' vers ℓ. Supposons qu'il existe une variable $x \in Var(\phi)$ telle que les sommets associés aux littéraux x et $\neg x$ dans le graphe soient dans la même composante fortement connexe. Si l'on voit les clauses de ϕ comme des implications, cela signifie qu'affecter 0 à x implique de lui affecter également 1, et inversement. On conclut donc que ϕ est insatisfaisable. Dans le cas contraire, on peut calculer un modèle de ϕ avec une variante de tri topologique. L'algorithme linéaire de Tarjan pour calculer les composantes fortement connexes d'un graphe donné termine la preuve [Tar72, APT79]. □

Quant aux propriétés sémantiques des formules 2FNC, elles sont de la même nature que celles des formules FNC de Horn. La clôture de l'ensemble de modèles d'une telle formule, en revanche, implique trois modèles au lieu de deux. Pour m, m', m'' trois affectations à un même ensemble de variables V, nous notons $maj(m, m', m'')$ l'affectation à V définie par :

$$\forall x \in V, (maj(m, m', m''))[x] = (m[x] \wedge m'[x]) \vee (m[x] \wedge m''[x]) \vee (m'[x] \wedge m''[x])$$

Autrement dit, la fonction maj renvoie, pour chaque variable, l'affectation majoritaire parmi celles de ses trois arguments.

Proposition 2.13 (BIJ-clôture [Pos41, Sch78]) *Un ensemble d'affectations M à un ensemble de variables V est descriptible dans* BIJ *si et seulement si pour tous $m, m', m'' \in M$, on a $maj(m, m', m'') \in M$.*

Preuve Supposons tout d'abord M descriptible dans BIJ, et soit ϕ une formule 2FNC le décrivant. Soient $m, m', m'' \in M$, et soit C une clause de ϕ. Alors m, m' et m'' satisfont C, et comme C contient au plus deux littéraux il en existe au moins un qui est satisfait par deux affectations parmi m, m' et m'' ; il est donc également satisfait par l'affectation $maj(m, m', m'')$, donc $maj(m, m', m'')$ satisfait C. Ceci étant vrai pour une clause $C \in \phi$ quelconque, $maj(m, m', m'')$ satisfait ϕ, et donc est dans M.

Pour la réciproque, comme pour la proposition 2.2 nous renvoyons au chapitre 7, où une construction est donnée (de même que pour la proposition 2.2, cette construction ne dépend que du sens direct de l'énoncé). □

Nous posons donc, comme pour les autres classes et pour un ensemble d'affectations M quelconque :

$$Cl_{\mathrm{BIJ}}(M) = \{maj(m, m', m'') \mid m, m', m'' \in M\}$$

On voit donc que la base de connaissances de l'exemple du carrefour peut être représentée par une formule 2FNC, puisque l'on a $Cl_{\mathrm{BIJ}}(M_C) = M_C$. En effet, la formule FNC :

$$(\neg x_{DT} \vee x_{VP}) \wedge (\neg x_{VP} \vee x_{DT}) \wedge (\neg x_{VV} \vee \neg x_{DT}) \wedge (\neg x_{VP} \vee \neg x_{VV})$$

admet bien M_C pour ensemble de modèles. En revanche, l'ensemble M_t d'affectations de l'exemple de la thèse vérifie

$$00111, 01000, 10100 \in M_t \text{ mais } maj(00111, 01000, 10100) = 00100 \notin M_t$$

et cette base de connaissances ne peut donc être représentée par une formule 2FNC sur les mêmes variables.

Nous terminons encore une fois par la caractérisation de cette classe par les formules FNC premières, similaire à celle des autres classes.

Proposition 2.14 (formule 2FNC première) *Un ensemble d'affectations M est descriptible dans* BIJ *si et seulement si toute formule en FNC et première décrivant M est en 2FNC.*

Preuve La preuve est similaire à celle de la proposition 2.3. Le sens indirect est trivial, et nous supposons donc M descriptible dans BIJ. Soit C un impliqué premier d'une description de M, et par l'absurde soient ℓ, ℓ', ℓ'' trois littéraux différents de C. Puisque C est première, il existe une affectation m (resp. m', m'') satisfaisant ℓ (resp. ℓ', ℓ'') mais pas les autres littéraux de C. De fait, pour chaque littéral $\ell_0 \in \{\ell, \ell', \ell''\}$ il existe deux affectations ne satisfaisant pas ℓ_0 parmi m, m', m'' ; comme aucune des trois ne satisfait les autres littéraux de C, on en déduit que l'affectation $maj(m, m', m'')$ ne satisfait pas C, donc qu'elle ne satisfait pas ϕ ; or, d'après la proposition 2.13 elle est dans M, ce qui est absurde puisque ϕ décrit M. □

2.5 Classes duales

Dans ce dernier paragraphe, nous présentons rapidement les classes «duales» des classes HORN, HDP et NÉG. Ces trois dernières classes sont en effet définies par le signe des littéraux qui composent leurs clauses. Si l'on inverse alors les restrictions sur les signes, en imposant par exemple un littéral *négatif* au plus par clause, on obtient ainsi la classe des formules FNC *anti-*Horn, notée ANTI-HORN. Cette classe possède exactement les mêmes propriétés que la classe HORN, du moment que l'on traite les problèmes eux aussi «opposés». Ainsi, on peut calculer efficacement le (seul) modèle minimal d'une formule de Horn, c'est-à-dire celui qui affecte 1 au moins de variables ; on peut donc calculer efficacement le (seul) modèle *maximal* d'une formule anti-Horn avec l'algorithme «symétrique». En revanche, la génération efficace des modèles *minimaux* d'une formule anti-Horn est un problème ouvert (nous renvoyons à l'article de Kavvadias *et al.* [KSS00] pour plus de détails sur la génération des modèles minimaux/maximaux).

Nous obtenons ainsi à partir des classes HORN, HDP et NÉG les classes de formules suivantes :

- ANTI-HORN, la classe des formules FNC *anti-Horn*, dont chaque clause possède au plus un littéral *négatif*
- ANTI-HDP, la classe des formules FNC *anti-Horn définies positives*, dont chaque clause contient exactement un littéral négatif (pour que la dénomination de cette classe soit cohérente, lire «anti-(Horn définie positive)» plutôt que «(anti-Horn) définie positive»)
- POS, la classe des formules FNC *positives*, dont chaque clause ne contient que des littéraux positifs.

Bien entendu, la classe duale de la classe BIJ est elle-même, puisque la définition de cette classe ne prend pas en compte le signe des littéraux dans les clauses. Notons que les propriétés de clôture des ensembles de modèles sont également duales ; ainsi, l'ensemble des modèles d'une formule de ANTI-HORN est clos par $\vee$ entre affectations, celui d'une formule ANTI-HDP, par $\vee$ et il contient l'affectation de toutes les variables à 0, et enfin, si une affectation est un modèle d'une formule de POS, alors toute affectation *supérieure* en est également un.

Par souci de complétude, nous mentionnerons ces classes dans les résultats des différents chapitres de ce mémoire, mais, puisque ce sera toujours possible, omettrons les preuves en invoquant les raisonnements symétriques.

Chapitre 3

Formules affines

Sommaire

3.1	**Définition et exemples**	**41**
3.2	**Propriétés algorithmiques**	**43**
3.3	**Fonctions affines et espaces vectoriels**	**45**
3.4	**Propriétés sémantiques**	**48**
3.5	**Les formules affines pour la représentation de connaissances**	**50**

Nous présentons maintenant les formules *affines*. Comme nous l'avons dit dans le chapitre 1, ces formules ne sont ni en FNC ni en FND, mais nous avons choisi de les étudier au même titre que les autres classes de formules FNC car on peut les voir comme des réseaux de contraintes. Cette classe étant très peu étudiée, dans la littérature, dans l'optique de la représentation de connaissances, nous nous attardons plus longuement sur elle que sur les autres, plus classiques.

3.1 Définition et exemples

Les formules affines, aussi appelées formules *XOR-SAT*, ou encore *biconditionnelles*, sont en fait les *systèmes d'équations linéaires sur le corps* $\{0, 1\}$ [Cur84] ; elles ne sont donc absolument pas nouvelles. On les retrouve dans la littérature d'Algorithmique ou d'Intelligence Artificielle principalement en tant que classe polynomiale pour des problèmes difficiles, envisagée la plupart du temps pour montrer des résultats de dichotomie (montrant qu'un problème devient soit polynomial, soit NP-complet lorsque l'on restreint ses données, par exemple). Le premier résultat de dichotomie établi est celui de Schaefer [Sch78] pour le problème SAT-Généralisé (nous reparlerons de ce résultat dans le chapitre 5). Depuis, d'autres résultats de dichotomie ont pu être établis (parmi lesquels [CH96, KS98, KK01, BHRV02], voir aussi [CKS01]), tous impliquant les formules affines. Néanmoins, peu de travaux les concernent dans le contexte de l'Intelligence Artificielle [KK01, DH03], et un de nos objectifs est de montrer qu'elles peuvent pourtant être considérées pour la représentation de connaissances, au même titre que d'autres classes de formules que l'on utilise pour leurs bonnes propriétés algorithmiques.

La classe des formules affines, que nous notons AFFINE, n'étant pas une classe de formules FNC, sa définition passe, non par celle de *clauses* particulières, mais par celle d'une *équation linéaire*. Pour les notions (élémentaires) d'algèbre linéaire que nous utilisons, nous renvoyons le lecteur aux ouvrages de Curtis [Cur84] (chapitre 8 principalement) ou de Schrijver [Sch86] (chapitre 3), par exemple.

Définition 3.1 (formule affine) *Une* équation linéaire *est une équation de la forme* $(x_{i_1} \oplus x_{i_2} \oplus \cdots \oplus x_{i_k} = a)$*, où les* x_{i_j} *sont des variables toutes différentes et où* a *vaut* 0 *ou* 1*. Une* formule affine *est une conjonction d'équations linéaires toutes différentes, c'est-à-dire formées sur des variables différentes ou de membres droits différents.*

Ainsi, la formule :

$$\phi''_t = (x_{qu} \oplus x_{ba} = 1) \wedge (x_{me} \oplus x_{ba} \oplus x_{in} = 0) \wedge (x_{in} \oplus x_{ch} = 0)$$

est une formule affine sur l'ensemble de variables $V_t = \{x_{me}, x_{qu}, x_{ba}, x_{in}, x_{ch}\}$, et elle est logiquement équivalente à la formule ϕ_t représentant la connaissance relative au petit monde de la thèse (préliminaires généraux, paragraphe 3.3).

Nous disons qu'une variable x *appartient* à une équation linéaire E, ou que E *contient* x, si x apparaît dans la partie gauche de E. Nous voyons également, comme pour les formules FNC, une formule affine comme l'*ensemble* de ses équations. Enfin, si $(x_{i_1} \oplus x_{i_2} \oplus \cdots \oplus x_{i_k} = a)$ est une équation linéaire, a est appelé son *membre droit*.

Remarquons que, avec la définition 3.1, une formule affine n'est pas à proprement parler une formule de la logique propositionnelle, mais que de façon équivalente, on peut définir une équation linéaire comme le ou exclusif (somme modulo 2) d'un ensemble de littéraux : ainsi l'équation $(x_{me} \oplus x_{ba} \oplus x_{in} = 0)$ de ϕ''_t a-t-elle la même table de vérité que la formule $(\neg x_{me} \oplus x_{ba} \oplus x_{in})$, puisque pour toute variable x on a $\neg x \equiv x \oplus 1$; pour ce qui est de l'équation $(0 = 0)$, qui ne contient aucune variable, on peut la représenter par la *négation* d'une somme vide de littéraux, $\neg()$. Nous choisissons cependant de conserver la notation sous forme d'équation, car la notation comme ensemble de littéraux laisse penser que les signes de ces derniers ont une influence sur les affectations satisfaisantes, alors que c'est seulement la *parité du nombre* de littéraux négatifs qui en a une, et car nous utilisons des outils d'algèbre linéaire, comme le pivot de Gauss, qui s'expriment plus classiquement avec cette notation. Remarquons tout de même qu'avec la notation ensembliste, on peut voir les formules affines comme des formules FNC dans lesquelles on aurait substitué le connecteur $\oplus$ au connecteur $\vee$. Remarquons également, même si nous n'utilisons pas directement cette transformation, qu'une équation linéaire $(x_{i_1} \oplus x_{i_2} \oplus \cdots \oplus x_{i_k} = a)$ peut être représentée de façon canonique par la conjonction de toutes les clauses sur $\{x_{i_1}, x_{i_2}, \ldots, x_{i_k}\}$ contenant un nombre pair de littéraux négatifs (si a vaut 0, impair sinon) : ainsi, la formule ϕ''_t ci-dessus est logiquement équivalente à la formule FNC ϕ_t de l'exemple de la thèse :

$$\begin{array}{rll} \phi_t = & & (x_{qu} \vee x_{ba}) \wedge (\neg x_{qu} \vee \neg x_{ba}) \\ & \wedge & (x_{me} \vee \neg x_{ba} \vee x_{in}) \wedge (x_{me} \vee x_{ba} \vee \neg x_{in}) \wedge (\neg x_{me} \vee x_{ba} \vee x_{in}) \\ & \wedge & (\neg x_{me} \vee \neg x_{ba} \vee \neg x_{in}) \\ & \wedge & (\neg x_{in} \vee x_{ch}) \wedge (x_{in} \vee \neg x_{ch}) \end{array}$$

Remarquons que la formule affine ϕ''_t est plus concise que la formule FNC ϕ_t ; nous verrons dans le paragraphe 3.2 que les formules affines peuvent toujours être rendues très concises.

Notons enfin qu'il existe une équation linéaire insatisfaisable $((0 = 1)$, autrement dit le ou exclusif d'aucun littéral) et une équation linéaire tautologique $((0 = 0))$; pour toute clause unitaire (x) (resp. $(\neg x)$), il existe également une équation linéaire logiquement équivalente, $(x = 1)$ (resp. $(x = 0)$). Notons également que les deux seules formules affines ϕ tautologiques sont la formule sans équation $\phi = ()$ et la formule réduite à une équation $((0 = 0))$, et que la formule $\phi = ((0 = 1))$ est insatisfaisable. Remarquons enfin que la classe des formules affines est stable par conjonction ainsi que par propagation d'affectations quelconques.

3.2 Propriétés algorithmiques

Nous présentons maintenant les principales propriétés algorithmiques des formules affines. Notons tout d'abord qu'essentiellement, un parcours suffit pour reconnaître si une formule propositionnelle donnée est affine. Toutes les autres propriétés découlent de l'efficacité du même outil, le *pivot de Gauss* [Sch86, chapitre 3.3], destiné à la résolution des systèmes d'équations linéaires. Nous rappelons brièvement son mécanisme et ses propriétés.

L'algorithme du pivot de Gauss permet de *minimiser* le nombre d'équations d'une formule affine donnée ϕ, en ce sens qu'il permet d'obtenir une formule affine logiquement équivalente à ϕ et telle qu'aucune autre telle formule n'a strictement moins d'équations. En particulier, la seule formule affine insatisfaisable minimale (en ce sens) étant la formule $((0 = 1))$, le pivot de Gauss permet de décider la satisfaisabilité de ϕ. De plus, la formule qu'il calcule contient au plus $|Var(\phi)|$ équations. Nous redonnons cet algorithme sur la figure 3.1 ; rappelons que son principe consiste à *trianguler* la formule donnée selon un certain ordre total sur les variables, c'est-à-dire, en supposant également un ordre total sur les équations de la formule, à faire en sorte que la première équation contienne une certaine variable, que la seconde contienne une variable supérieure mais pas la première, et ainsi de suite. Par exemple, la formule affine ϕ''_t :

$$(x_{qu} \oplus x_{ba} = 1) \wedge (x_{me} \oplus x_{ba} \oplus x_{in} = 0) \wedge (x_{in} \oplus x_{ch} = 0)$$

est triangulaire en considérant l'ordre $(x_{qu}, x_{me}, x_{in}, x_{ba}, x_{ch})$ sur les variables, puisque la première équation contient x_{qu}, la deuxième, x_{me} mais pas x_{qu}, et la troisième, x_{in} mais ni x_{qu}, ni x_{me}. On peut de fait la représenter sous la forme d'une matrice triangulaire :

$$\left(\begin{array}{ccccc|c} x_{qu} & & & x_{ba} & & 1 \\ & x_{me} & x_{in} & x_{ba} & & 0 \\ & & x_{in} & & x_{ch} & 0 \end{array}\right)$$

Etant données deux équations linéaires E et E', nous notons $E \oplus E'$ l'équation obtenue en additionnant (modulo 2) E et E' membre à membre, et en simplifiant ; par exemple, pour $E = (x_1 \oplus x_2 \oplus x_3 = 1)$ et $E' = (x_1 \oplus x_3 \oplus x_4 = 1)$, on a $E \oplus E' = (x_2 \oplus x_4 = 0)$. Remarquons que si E et E' contiennent une même variable, alors $E \oplus E'$ ne la contient plus. Notons aussi que nous supposons fixé un ordre total sur les variables apparaissant dans ϕ ; nous utiliserons cet ordre plus loin (chapitre 11), mais quel qu'il soit l'algorithme reste correct, avec la même complexité.

Nous illustrons le fonctionnement de cet algorithme sur l'exemple de la formule ϕ''_t. Supposons que l'on veuille appliquer l'algorithme du pivot de Gauss à cette formule, sur l'ordre de variables $(x_{ba}, x_{in}, x_{qu}, x_{me}, x_{ch})$; pour simplifier la présentation, nous utilisons la notation sous forme d'une matrice, comme ci-dessus, selon l'ordre de variables considéré, et représentons les variables ϕ et ϕ' de l'algorithme dans la même matrice (version «sur place» de l'algorithme). Nous avons donc :

$$\phi''_t = \left(\begin{array}{ccccc|c} x_{ba} & & x_{qu} & & & 1 \\ x_{ba} & x_{in} & & x_{me} & & 0 \\ & x_{in} & & & x_{ch} & 0 \end{array}\right) \begin{array}{l} (E_1, \in \phi) \\ (E_2, \in \phi) \\ (E_3, \in \phi) \end{array}$$

Le pivot x de l'algorithme est donc x_{ba}, et nous choisissons l'équation E_1 pour la variable E de l'algorithme. Nous obtenons alors les formules ϕ et ϕ' suivantes :

$$\phi''_t \equiv \left(\begin{array}{ccccc|c} x_{ba} & & x_{qu} & & & 1 \\ & x_{in} & x_{qu} & x_{me} & & 1 \\ & x_{in} & & & x_{ch} & 0 \end{array}\right) \begin{array}{l} (E'_1 = E_1, \in \phi') \\ (E'_2 = E_2 \oplus E_1, \in \phi) \\ (E'_3 = E_3, \in \phi) \end{array}$$

Entrée : Une formule affine ϕ
Sortie : Une formule affine $\phi' \equiv \phi$ contenant un nombre minimal d'équations.

Début
  $\phi' \leftarrow \emptyset$;
  Tant Que ϕ n'est pas vide **Faire**
    $x \leftarrow$ la plus petite variable apparaissant dans ϕ ; /* *le* pivot */
    $E \leftarrow$ une équation de ϕ contenant x ;
    $\phi' \leftarrow \phi' \cup \{E\}$;
    $\phi \leftarrow \phi \backslash \{E\}$;
    Pour toute équation $E' \in \phi$ contenant x avec $E' \neq E$ **Faire**
      remplacer E' par $E' \oplus E$ dans ϕ ;
    Fin Pour ;
  Fin Tant Que ;
  retourner ϕ' ;
Fin

FIG. 3.1 – Algorithme `PivotGauss`

Le pivot devient alors la variable x_{in}, et pour la variable E nous choisissons l'équation E'_2. Nous obtenons alors les formules :

$$\phi''_t \equiv \left(\begin{array}{ccccc|c} x_{ba} & & x_{qu} & & & 1 \\ & x_{in} & x_{qu} & x_{me} & & 1 \\ & & x_{qu} & x_{me} & x_{ch} & 1 \end{array}\right) \begin{array}{l} (E''_1 = E'_1, \in \phi') \\ (E''_2 = E'_2, \in \phi') \\ (E''_3 = E'_3 \oplus E'_2, \in \phi) \end{array}$$

Comme il ne reste plus qu'une équation dans ϕ, l'algorithme la déplace dans ϕ', et nous obtenons finalement la formule affine :

$$(x_{ba} \oplus x_{qu} = 1) \wedge (x_{in} \oplus x_{qu} \oplus x_{me} = 1) \wedge (x_{qu} \oplus x_{me} \oplus x_{ch} = 1)$$

qui est par construction logiquement équivalente à ϕ''_t et minimale dans le sens défini plus haut.

Nous ne donnons pas ici la preuve de la correction de l'algorithme du pivot de Gauss, qui est extrêmement classique ; nous énonçons simplement le résultat (rappelons que nous voyons une formule affine comme l'ensemble de ses équations, et donc que la notation $|\phi|$ désigne le nombre d'équations qui composent une formule affine ϕ).

Proposition 3.1 (pivot de Gauss [Cur84, Sch86]) *Soit ϕ une formule affine. L'algorithme* `PivotGauss` *calcule une formule affine ϕ' logiquement équivalente à ϕ, triangulaire et (donc) contenant un nombre minimal d'équations en temps $O(|\phi|^2|Var(\phi)|)$; de plus, ce nombre d'équations est au plus $|Var(\phi)|$.*

Comme nous l'avons remarqué précédemment, quel que soit le nombre de variables la seule formule affine *minimale* insatisfaisable est la formule $((0 = 1))$, et donc le pivot de Gauss résout le problème `SAT` pour les formules affines. Si une formule affine est satisfaisable, on peut de plus calculer un de ses modèles à partir de la formule triangulaire obtenue, en procédant de façon gloutonne à partir de la dernière équation ; en effet, puisque chaque équation contient au moins une variable que ne contient pas la suivante, en procédant ainsi il suffit d'affecter une valeur à cette variable en fonction de l'équation courante. Par exemple, pour la formule ϕ''_t, la dernière

équation étant $(x_{in} \oplus x_{ch} = 0)$, nous pouvons affecter 0 aux variables x_{in} et x_{ch} ; après propagation de ces valeurs dans le reste de la formule, la deuxième équation devient $(x_{me} \oplus x_{ba} = 0)$, et nous pouvons affecter 0 aux variables x_{me} et x_{ba} ; enfin, la première équation devient après propagation $(x_{qu} = 1)$, et nous avons finalement calculé l'affectation m_t définie par :

$$m_t[x_{qu}] = 1 \text{ et } \forall x \in V_t, x \neq x_{qu}, m_t[x] = 0$$

dont on peut vérifier (table 3 page 22) que c'est un modèle de la formule affine $\phi''_t \equiv \phi_t$.

Proposition 3.2 (SAT[AFFINE]) *On peut décider si une formule affine ϕ donnée est satisfaisable en temps $O(|\phi|^2|Var(\phi)|)$, en en exhibant un modèle le cas échéant.*

Il est intéressant de remarquer que, tout comme la procédure de résolution (voir les préliminaires généraux) le fait avec les *clauses* d'une formule FNC, le Pivot de Gauss procède en éliminant une variable commune entre deux équations linéaires d'une formule affine. Néanmoins, remarquons également que la nouvelle équation ainsi calculée peut remplacer l'une des deux équations sommées dans la formule en préservant l'équivalence logique, tandis que ce n'est pas le cas en général avec le résolvant de deux clauses d'une formule FNC.

Enfin, les formules affines possèdent une propriété importante, que ne vérifient pas les autres classes de formules présentées dans ce mémoire ; c'est même la *seule* classe de formules définie localement et *maximale* pour l'inclusion qui la vérifie [CH96].

Proposition 3.3 (comptage des modèles) *On peut calculer le nombre de modèles d'une formule affine ϕ donnée en temps $O(|\phi|^2|Var(\phi)|)$.*

Preuve Soit ϕ une formule affine, et soit ϕ_{tr} une formule affine triangulaire logiquement équivalente à ϕ. Une telle formule peut être calculée en temps $O(|\phi|^2|Var(\phi)|)$ avec l'algorithme `PivotGauss`, et elle admet $2^{|Var(\phi_{tr})|-|\phi_{tr}|}$ modèles. Or l'ensemble des modèles de ϕ est par construction l'ensemble des prolongements des modèles de ϕ_{tr} à $Var(\phi)$, ces modèles sont donc au nombre de $2^{|Var(\phi_{tr})|-|\phi_{tr}|} \times 2^{|Var(\phi)|-|Var(\phi_{tr})|} = 2^{|Var(\phi)|-|\phi_{tr}|}$. Ce nombre peut donc bien être calculé en temps $O(|\phi|^2|Var(\phi)|)$ au total. □

3.3 Fonctions affines et espaces vectoriels

Les formules affines sont bien entendu fortement liées à l'algèbre linéaire, et plus particulièrement aux espaces vectoriels. Le but de ce paragraphe est de présenter ce lien et les notions d'algèbre utiles. Ces notions concernent principalement la *sémantique* des formules affines.

En tant que système d'équations linéaires, une formule affine admet un *ensemble de solutions*, que l'on peut identifier à ses *modèles* : il suffit de voir une affectation m à un ensemble de variables $V = \{x_{i_1}, x_{i_2}, \ldots, x_{i_{|V|}}\}$, $i_1 < i_2 < \ldots < i_{|V|}$ comme le $|V|$-uplet $(m[x_{i_1}], m[x_{i_2}], \ldots, m[x_{i_{|V|}}])$ de l'espace $\{0,1\}^{|V|}$. Il est bien connu que cet ensemble de solutions est un espace vectoriel sur le corps $\{0,1\}$ si (et seulement si) le système est homogène, c'est-à-dire si tous les membres droits des équations valent 0. Sinon, on obtient un tel espace vectoriel en remplaçant chacune de ces solutions par sa somme, modulo 2 et composante par composante, avec une même solution particulière du système ; on retrouve ainsi le cas particulier des systèmes homogènes, puisque l'affectation $\overline{0_V}$ est alors une telle solution. Nous formalisons ce lien dans la proposition suivante ; nous identifions les affectations à un ensemble de variables V et les éléments de $\{0,1\}^{|V|}$ comme

ci-dessus, et, pour deux affectations m, m' à V, nous notons $m \oplus m'$ l'affectation à V définie par $\forall x \in V, (m \oplus m')[x] = m[x] \oplus m'[x]$.

Proposition 3.4 (espaces vectoriels [Cur84]) *Soient ϕ une formule affine satisfaisable et m un modèle de ϕ. Alors l'ensemble d'affectations à $Var(\phi)$ défini comme l'ensemble $\{m \oplus m' \mid m' \models \phi\}$ est un espace vectoriel sur le corps $\{0, 1\}$.*

Par exemple, si nous considérons l'ensemble d'affectations M_t de l'exemple de la thèse (cf. préliminaires généraux, paragraphe 3.3) :

$$M_t = \{00111, 01000, 10100, 11011\}$$

en distinguant l'affectation $m = 00111$, l'ensemble d'affectations :

$$\{m \oplus m' \mid m' \in M_t\} = \{00000, 01111, 10011, 11100\}$$

est un espace vectoriel sur $\{0, 1\}$.

De façon générale, nous nous ramènerons donc toujours au cas où l'affectation $\overline{0_{Var(\phi)}}$ est un modèle de la formule affine ϕ considérée, afin de simplifier les calculs et les preuves. Pour cela, il suffit donc de choisir un modèle particulier m de ϕ et de considérer l'ensemble $\mathcal{M}(\phi)_{\oplus m} = \{m \oplus m' \mid m' \in \mathcal{M}(\phi)\}$ au lieu de l'ensemble $\mathcal{M}(\phi)$; quant à la formule ϕ, on voit aisément que la formule $\phi_{\oplus m}$ obtenue en remplaçant x par $x \oplus 1$ dans ϕ pour toute variable $x \in Var(\phi)$ avec $m[x] = 1$, et en simplifiant les constantes, admet exactement $\mathcal{M}(\phi)_{\oplus m}$ comme ensemble de modèles et peut être calculée en un seul parcours de ϕ. D'autre part, on voit aisément que l'on a $(\mathcal{M}(\phi)_{\oplus m})_{\oplus m} = \mathcal{M}(\phi)$ et $(\phi_{\oplus m})_{\oplus m} = \phi$. Par exemple, si l'on distingue à nouveau l'affectation $m = 00111$ de M_t, on obtient la formule :

$$(\phi''_t)_{\oplus m} = (x_{qu} \oplus x_{ba} = 0) \wedge (x_{me} \oplus x_{ba} \oplus x_{in} = 0) \wedge (x_{in} \oplus x_{ch} = 0)$$

Via cette reformulation, nous pouvons donc utiliser la notion de *base* d'un espace vectoriel. Rappelons qu'un sous-ensemble B de $\{0, 1\}^n$ est dit *linéairement indépendant* si aucun vecteur $b \in B$ n'est égal à la somme, modulo 2 et composante par composante, d'autres vecteurs de B. Rappelons également qu'un espace vectoriel sur $\{0, 1\}$ contient toujours 2^m vecteurs pour un certain $m \in \mathbb{N}$.

Définition 3.2 (base) *Soit $E \subseteq \{0, 1\}^n$ un espace vectoriel sur $\{0, 1\}$, et soit $m \in \mathbb{N}$ tel que $|E| = 2^m$. Un sous-ensemble B de E tel que $|B| = m$ et que B est linéairement indépendant est appelé une* base *de E.*

Notons que cette définition est habituellement considérée comme un théorème, corollaire direct de la définition d'une base comme ensemble générateur minimal ; nous avons cependant choisi de simplifier la présentation, car cette notion est très classique.

Une base B d'un espace vectoriel E caractérise complètement E, puisque E est par définition d'une base l'ensemble des combinaisons linéaires d'éléments de B. Cette notion nous sera donc très utile. Par exemple, une base de l'espace vectoriel $(M_t)_{\oplus m} = \{00000, 01111, 10011, 11100\}$ ci-dessus est l'ensemble (de cardinal $\log |M_t| = 2$) $\{01111, 10011\}$, dont l'ensemble des combinaisons linéaires sur $\{0, 1\}$ est bien :

$$(M_t)_{\oplus m} = \{0 \times (01111 \oplus 10011) = 00000, 01111, 10011, 1 \times (01111 \oplus 10011) = 11100\}$$

Il est également important de rappeler que l'algorithme du pivot de Gauss permet de trouver un sous-ensemble linéairement indépendant d'affectations dans un ensemble d'affectations donné. Nous ne détaillons pas ce point ici, mais notons qu'on peut donc en particulier calculer une base d'un espace vectoriel donné M, sur un ensemble de variables V, en temps $O(|M|^2|V|)$. Par exemple, pour l'espace vectoriel $(M_t)_{\oplus m}$ ci-dessus, un calcul possible est le suivant :

$$(M_t)_{\oplus m} = \begin{pmatrix} 1 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 \end{pmatrix} \begin{matrix} (m_1) \\ (m_2) \\ (m_3) \end{matrix} \quad \hookrightarrow \quad \begin{pmatrix} 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 & 1 \end{pmatrix} \begin{matrix} (m_1) \\ (m_2 \oplus m_1) \\ (m_3) \end{matrix}$$

Après suppression de la redondance, on obtient une matrice triangulaire, et on a donc calculé la base $\{10011, 01111\}$ de $(M_t)_{\oplus m}$.

Nous terminons ce paragraphe en montrant comment on peut calculer une formule affine, admettant pour ensemble de modèles un espace vectoriel donné par une de ses bases B, en temps polynomial en la taille de B. Le calcul, bien que simple, est un peu technique, et est encapsulé dans la preuve de la proposition suivante.

Proposition 3.5 (base en formule) *Soient V un ensemble de variables et $B \subseteq \{0,1\}^{|V|}$ un ensemble linéairement indépendant. On peut calculer une formule affine décrivant l'espace vectoriel de base B en temps $O(|V|^3)$.*

Preuve Nous décrivons tout d'abord le calcul, puis prouvons qu'il est correct. Notons $B = \{m_1, m_2, \ldots, m_{|B|}\}$, et notons M l'espace vectoriel de base B.

[construction] La première étape du calcul consiste à compléter B avec $|V| - |B|$ affectations $m_{|B|+1}, m_{|B|+2}, \ldots, m_{|V|}$ à V, de sorte que l'ensemble $\{m_1, m_2, \ldots, m_{|V|}\}$ soit une base de l'espace vectoriel $\{0,1\}^{|V|}$; ces affectations peuvent être calculées en temps $O(|B|^2|V| + |V|^2)$ en triangulant l'ensemble B (en temps $O(|B|^2|V|)$ avec le pivot de Gauss) puis en le complétant tout en maintenant la triangularité (en temps $O(|V|^2)$). Notons maintenant $V = \{x_1, x_2, \ldots, x_{|V|}\}$, sans perte de généralité. Il s'agit alors d'associer l'équation linéaire $E_j = (\bigoplus_{i \in \{1,2,\ldots,|V|\}} c_{ij} x_i = 0)$ à m_j pour tout $j \in \{|B|+1, |B|+2, \ldots, |V|\}$, où pour un j fixé les coefficients c_{ij} sont déterminés par le système d'équations :

$$S_j = \begin{cases} m_1[x_1] \quad c_{1j} \oplus \cdots \oplus m_1[x_{|V|}] \quad c_{|V|j} = 0 \\ \cdots \\ m_{j-1}[x_1] \quad c_{1j} \oplus \cdots \oplus m_{j-1}[x_{|V|}] \quad c_{|V|j} = 0 \\ m_j[x_1] \quad c_{1j} \oplus \cdots \oplus m_j[x_{|V|}] \quad c_{|V|j} = 1 \\ m_{j+1}[x_1] \quad c_{1j} \oplus \cdots \oplus m_{j+1}[x_{|V|}] \quad c_{|V|j} = 0 \\ \cdots \\ m_{|V|}[x_1] \quad c_{1j} \oplus \cdots \oplus m_{|V|}[x_{|V|}] \quad c_{|V|j} = 0 \end{cases}$$

Pour un j fixé, ce système peut être construit en temps $O(|V|^2)$; comme il est triangulaire (par construction) et contient $|V|$ équations, il admet une unique solution, qui peut être calculée en temps $O(|V|^2)$. Notons ϕ la formule affine $\bigwedge_{j=|B|+1}^{|V|} E_j$; ϕ peut être construite en temps $O(|B|^2|V| + (|V| - |B|)|V|^2)$, et donc en temps $O(|V|^3)$ puisque l'on a nécessairement $|B| \leq |V|$.

[correction] Nous montrons maintenant que ϕ décrit M. Par construction de S_j, on a tout d'abord $m_i \models E_j$ pour tout $i \in \{1, 2, \ldots, |B|\}$; comme ϕ est une formule homogène, on en déduit que les combinaisons linéaires d'affectations de $\{m_1, m_2, \ldots, m_{|B|}\}$ satisfont E_j, et donc que M est inclus dans $\mathcal{M}(\phi)$.

Réciproquement, soit $m \notin M$ une affectation. Puisque, par construction, l'ensemble $B' = B \cup \{m_{|B|+1}, m_{|B|+2}, \ldots, m_{|V|}\}$ est une base de l'espace vectoriel $\{0,1\}^{|V|}$, il existe des coefficients $\lambda_1, \lambda_2, \ldots, \lambda_{|V|} \in \{0,1\}$ tels que l'on ait $m = \bigoplus_{j=1}^{|V|} \lambda_j m_j$; de $m \notin M$ on déduit alors qu'il existe $j \in \{|B|+1, |B|+2, \ldots, |V|\}$ avec $\lambda_j = 1$, et il est facile de voir que m ne satisfait pas l'équation E_j, et donc que m ne satisfait pas ϕ. □

Nous illustrons enfin cette construction sur l'exemple précédent ; il s'agit donc de calculer une description affine de l'espace vectoriel $(M_t)_{\oplus m}$ étant donnée sa base $\{m_1 = 10011, m_2 = 01111\}$. Nous complétons tout d'abord cette base avec les affectations $m_3 = 00100$, $m_4 = 00010$ et $m_5 = 00001$. Nous devons ensuite résoudre les systèmes d'équations :

$$S_3 = \begin{cases} c_{13} & & & & \oplus & c_{43} & \oplus & c_{53} & = & 0 \\ & c_{23} & \oplus & c_{33} & \oplus & c_{43} & \oplus & c_{53} & = & 0 \\ & & & c_{33} & & & & & = & 1 \\ & & & & & c_{43} & & & = & 0 \\ & & & & & & & c_{53} & = & 0 \end{cases}$$

$$S_4 = \begin{cases} c_{14} & & & & \oplus & c_{44} & \oplus & c_{54} & = & 0 \\ & c_{24} & \oplus & c_{34} & \oplus & c_{44} & \oplus & c_{54} & = & 0 \\ & & & c_{34} & & & & & = & 0 \\ & & & & & c_{44} & & & = & 1 \\ & & & & & & & c_{54} & = & 0 \end{cases}$$

$$S_5 = \begin{cases} c_{15} & & & & \oplus & c_{45} & \oplus & c_{55} & = & 0 \\ & c_{25} & \oplus & c_{35} & \oplus & c_{45} & \oplus & c_{55} & = & 0 \\ & & & c_{35} & & & & & = & 0 \\ & & & & & c_{45} & & & = & 0 \\ & & & & & & & c_{55} & = & 1 \end{cases}$$

On obtient ainsi les équations $E_3 = (x_{qu} \oplus x_{ba} = 0)$, $E_4 = (x_{me} \oplus x_{qu} \oplus x_{in} = 0)$, et $E_5 = (x_{me} \oplus x_{qu} \oplus x_{ch} = 0)$, et donc la formule affine $\phi = (E_3 \wedge E_4 \wedge E_5)$ décrit $(M_t)_{\oplus m}$.

3.4 Propriétés sémantiques

Le lien entre les formules affines et les espaces vectoriels confère aux premières de très fortes propriétés sémantiques, que nous formulons ici dans les mêmes termes que pour les classes de formules du chapitre 2. Rappelons que pour deux affectations m, m' à un même ensemble de variables V, nous notons $m \oplus m'$ l'affectation à V qui pour toute variable $x \in V$, affecte la valeur $m[x] \oplus m'[x]$ à x.

Proposition 3.6 (AFFINE-clôture [Pos41, Sch78]) *Un ensemble d'affectations M à un ensemble de variables V est descriptible dans* AFFINE *si et seulement si on a $\forall m, m', m'' \in M, m \oplus m' \oplus m'' \in M$.*

Preuve Supposons M descriptible dans AFFINE. Si M est vide, la proposition est triviale. Sinon, soit ϕ une formule affine décrivant M. D'après la proposition 3.4 il existe un modèle m_0 de ϕ tel que l'ensemble d'affectations $\mathcal{M}(\phi)_{\oplus m_0} = \{m_0 \oplus m' \mid m' \in \mathcal{M}(\phi)\}$ est un espace vectoriel. Par définition d'un espace vectoriel, on a donc pour tous $m, m', m'' \in \mathcal{M}(\phi)$, $(m_0 \oplus m) \oplus (m_0 \oplus m') \oplus (m_0 \oplus m'') = (m \oplus m' \oplus m'') \oplus m_0 \in \mathcal{M}(\phi)_{\oplus m_0}$, et donc $m \oplus m' \oplus m'' \in \mathcal{M}(\phi)$. Puisque ϕ décrit M, M est l'ensemble des prolongements des modèles de ϕ à V, et on a le résultat.

Pour la réciproque, encore une fois le cas $M = \emptyset$ est trivial. Si M n'est pas vide, soit m une affectation de M. L'ensemble d'affectations $M_{\oplus m}$ contient $\overline{0_V}$, et pour tous $m'_{\oplus m}, m''_{\oplus m} \in M_{\oplus m}$ on a donc $m'_{\oplus m} \oplus m''_{\oplus m} = m' \oplus m \oplus m'' \oplus m = m' \oplus m'' = m' \oplus m'' \oplus \overline{0_V} \in M$; on en déduit que $M_{\oplus m}$ est un espace vectoriel sur $\{0,1\}$, et par la proposition 3.5 qu'il est descriptible par une formule affine ϕ ; donc la formule affine $\phi_{\oplus m}$ décrit M. □

L'AFFINE-clôture d'un ensemble d'affectations M quelconque sera donc définie par :

$$Cl_{\text{AFFINE}}(M) = \{m \oplus m' \oplus m'' \mid m, m', m'' \in M\}$$

Quant à la caractérisation des formules affines en termes de formules premières, elle ne peut être donnée comme pour les formules (FNC) du chapitre 2. Nous montrons en revanche que les formules affines et les formules FNC premières décrivant un même ensemble d'affectations sont très fortement liées. Soit en effet M un ensemble d'affectations, et soit ϕ une formule FNC première décrivant M. Nous montrons que l'on peut obtenir une formule affine ϕ' décrivant M (et donc $\phi' \equiv \phi$) à partir de ϕ uniquement par des transformations syntaxiques. Pour C une clause, soit $aff(C)$ l'équation linéaire obtenue à partir de C en posant la somme des littéraux de C égale à 1, en remplaçant les littéraux négatifs de la forme $\neg x$ par $x \oplus 1$ et en simplifiant les constantes ; essentiellement, il s'agit donc de remplacer la disjonction $\vee$ à l'intérieur de C par le ou exclusif $\oplus$. Pour ϕ une formule FNC, soit $aff(\phi)$ la formule affine $\{aff(C) \mid C \in \phi\}$. Alors on a la propriété suivante.

Proposition 3.7 (formules affines et FNC premières) *Soit M un ensemble d'affectations à un ensemble de variables V descriptible dans* AFFINE*, et soit ϕ une formule FNC première décrivant M. Alors la formule affine $aff(\phi)$ décrit M.*

Preuve Nous montrons tout d'abord que toute affectation à V satisfaisant $aff(\phi)$ satisfait ϕ, et donc est dans M ; en effet, si C est une clause de ϕ alors toute affectation à V satisfaisant $aff(C)$ satisfait par construction un nombre impair de littéraux de C, donc en particulier au moins 1.

Réciproquement, soient $m \in M$ une affectation à V et C une clause de ϕ, et supposons par l'absurde $m \not\models aff(C)$. Alors par construction m satisfait un nombre pair de littéraux de C, que nous notons $\ell_1, \ell_2, \ldots, \ell_{2n}$. Puisque C est première, pour tout $i \in \{1, 2, \ldots, 2n\}$ il existe une affectation $m_i \in M$ qui ne satisfait par $C \backslash \{\ell_i\}$, mais qui satisfait C puisque ϕ décrit M. Donc m_i satisfait ℓ_i, mais aucun autre littéral de C. Considérons alors l'affectation à V $m' = m \oplus \bigoplus_{i=1}^{2n} m_i$; puisque pour tout $i \in \{1, 2, \ldots, 2n\}$ on a $m \models \ell_i$, $m_i \models \ell_i$ et $\forall j \in \{1, 2, \ldots, 2n\}, j \neq i \Longrightarrow m_j \not\models \ell_i$, on voit que pour tout $i \in \{1, 2, \ldots, 2n\}$ on a $m' \not\models \ell_i$, et finalement on a $m' \not\models C$. Donc m' n'est pas dans M, or m' est la somme modulo 2 d'un nombre impair d'affectations de M, donc on obtient une contradiction avec la proposition 3.6. □

Par exemple, la formule FNC :

$$\begin{array}{rcl} \phi_t = & & (x_{qu} \vee x_{ba}) \wedge (\neg x_{qu} \vee \neg x_{ba}) \\ & \wedge & (x_{me} \vee \neg x_{ba} \vee x_{in}) \wedge (x_{me} \vee x_{ba} \vee \neg x_{in}) \wedge (\neg x_{me} \vee x_{ba} \vee x_{in}) \\ & \wedge & (\neg x_{me} \vee \neg x_{ba} \vee \neg x_{in}) \\ & \wedge & (\neg x_{in} \vee x_{ch}) \wedge (x_{in} \vee \neg x_{ch}) \end{array}$$

de l'exemple de la thèse est une formule première, et la formule $aff(\phi_t)$ est exactement la formule du paragraphe 3.1 :

$$\phi''_t = (x_{qu} \oplus x_{ba} = 1) \wedge (x_{me} \oplus x_{ba} \oplus x_{in} = 0) \wedge (x_{in} \oplus x_{ch} = 0)$$

Comme l'ensemble M_t des modèles de ϕ_t vérifie $M_t = Cl_{\text{AFFINE}}(M_t)$, on peut en déduire que les formules ϕ_t et ϕ''_t sont logiquement équivalentes, et donc que l'on a $M_t = \mathcal{M}(\phi''_t)$.

Remarquons que dans le cas général où l'ensemble d'affectations M n'est pas nécessairement descriptible dans AFFINE, la formule $aff(\phi)$ est logiquement plus forte que ϕ; en effet, comme le montre la preuve de la proposition 3.7, pour une clause $C \in \phi$ quelconque tout modèle de $aff(C)$ affecte 1 à un nombre impair de littéraux de ϕ, en particulier à au moins 1, et donc satisfait C également.

3.5 Les formules affines pour la représentation de connaissances

Nous terminons ce chapitre en discutant les types de connaissances que peuvent intuitivement représenter les formules affines.

Remarquons tout d'abord que, si une clause exprime une *implication*, par exemple l'implication «Si le feu pour piétons est vert, alors on a le droit de traverser» pour la clause $(\neg x_{VP} \vee x_{DT})$ de l'exemple du carrefour, une équation linéaire exprime une *équivalence* : ainsi l'équation $(x_{in} \oplus x_{ch} = 0)$ de l'exemple de la thèse exprime-t-elle l'équivalence «le lecteur est intéressé si et seulement s'il est chercheur en informatique», et l'équation $(x_{qu} \oplus x_{ba} = 1)$, l'équivalence «le document est de qualité si et seulement s'il n'est pas bâclé». Plus complexe, l'équation $(x_{me} \oplus x_{ba} \oplus x_{in} = 0)$ exprime l'équivalence «pour que la lecture soit mémorable, il faut et il suffit que le travail soit bâclé si et seulement si le lecteur n'est pas intéressé par le sujet». Une équation linéaire peut ainsi spécifier qu'une variable est le témoin de l'égalité, ou de la différence, entre deux autres variables. Plus généralement, les équations linéaires peuvent être vues comme des contraintes de *parité* ou d'*imparité*.

Des propriétés plus générales peuvent cependant être énoncées concernant les connaissances que peuvent représenter les formules affines. Les deux premiers résultats concernent la sensibilité de ces formules aux modifications de leurs modèles, autrement dit leur *dépendance* en leurs variables [LLM02].

Proposition 3.8 (sensibilité à un bit) *Soient ϕ une formule affine satisfaisable, $x \in Var(\phi)$ et $m \in \mathcal{M}(\phi)$. Alors l'affectation m' à $Var(\phi)$ définie par $m'[x] = m[x] \oplus 1$ et $\forall x' \in Var(\phi), x' \neq x \Longrightarrow m'[x'] = m[x']$ ne satisfait pas ϕ.*

Preuve Puisque x est dans $Var(\phi)$, ϕ contient au moins une équation E de la forme $(x_{i_1} \oplus \cdots \oplus x_{i_k} \oplus x \oplus x_{i_{k+1}} \oplus \cdots \oplus x_{i_\ell} = a)$ avec $x_{i_1}, \ldots, x_{i_\ell} \in Var(\phi)$ et $a \in \{0, 1\}$. Puisque m satisfait ϕ on a $m[x_{i_1}] \oplus \cdots \oplus m[x_{i_k}] \oplus m[x] \oplus m[x_{i_{k+1}}] \oplus \cdots \oplus m[x_{i_\ell}] = a$, et donc $m'[x_{i_1}] \oplus \cdots \oplus m'[x_{i_k}] \oplus m'[x] \oplus m'[x_{i_{k+1}}] \oplus \cdots \oplus m'[x_{i_\ell}] = a \oplus 1$ et par conséquent $m' \not\models E$, donc $m' \not\models \phi$. □

Corollaire 3.1 (variables utiles) *Soient ϕ une formule affine satisfaisable et $x \in Var(\phi)$. Alors il n'existe aucune formule propositionnelle logiquement équivalente à ϕ et dans laquelle x n'apparaît pas.*

Un dernier résultat concerne la répartition des affectations de 0 et de 1 aux variables par les modèles d'une formule affine.

Proposition 3.9 (symétrie des affectations) *Soit ϕ une formule affine, et $x \in Var(\phi)$ une variable telle que ϕ n'implique ni x, ni $\neg x$. Alors il y a autant de modèles de ϕ qui affectent 1 à x que de modèles qui lui affectent 0.*

Preuve Soient m_0, m_1 deux modèles de ϕ avec $m_0[x] = 0$ et $m_1[x] = 1$. Alors pour tout modèle $m'_0 \neq m_0$ de ϕ avec $m'_0[x] = 0$, on a par la proposition 3.6 $m'_1 = m_1 \oplus m_0 \oplus m'_0 \in \mathcal{M}(\phi)$; or on a par construction $m'_1[x] = 1$ et $m'_1 \neq m_1$. On en déduit $|\{m \in \mathcal{M}(\phi) \,|\, m[x] = 1\}| \geq |\{m \in \mathcal{M}(\phi) \,|\, m[x] = 0\}|$, et par le raisonnement symétrique qu'il y a égalité. □

Remarquons enfin, pour terminer ce chapitre, que la classe des formules affines possède de nombreuses autres bonnes propriétés, que nous ne détaillons cependant pas ici car elles seraient hors-sujet. Mentionnons simplement qu'on peut définir, parallèlement à la notion d'impliqué premier des formules FNC, une notion d'*impliqué linéaire premier* d'une formule affine ϕ, c'est-à-dire, intuitivement, une équation linéaire E impliquée par ϕ mais dont on ne peut retirer aucune variable, même en inversant le membre droit, en conservant la propriété $\phi \models E$. Le pivot de Gauss permet encore une fois de décider efficacement si une équation linéaire est un impliqué linéaire premier d'une formule affine donnée. De plus, pour une formule affine ϕ cette notion d'impliqué premier caractérise totalement les dépendances fonctionnelles que satisfait l'ensemble $\mathcal{M}(\phi)$, vu comme une base de données sur l'ensemble d'attributs $Var(\phi)$: en effet, les dépendances fonctionnelles minimales vérifiées par $\mathcal{M}(\phi)$ sont exactement les $(X \to x)$, avec $X \subseteq Var(\phi)$ et $x \in Var(\phi)$, telles que l'équation $(x \oplus \bigoplus_{x' \in X} x' = 0)$ ou l'équation $(x \oplus \bigoplus_{x' \in X} x' = 1)$ est un impliqué linéaire premier de ϕ.

Ainsi, si l'on reprend l'exemple de la thèse, l'équation $(x_{me} \oplus x_{ba} \oplus x_{in} = 0)$ est un impliqué linéaire premier de la formule ϕ''_t ; en effet, puisque c'en est une équation, elle est bien impliquée par ϕ''_t, mais aucune des équations

$$(x_{me} \oplus x_{ba} = 0), (x_{me} \oplus x_{ba} = 1), (x_{me} \oplus x_{in} = 0), (x_{me} \oplus x_{in} = 1), (x_{ba} \oplus x_{in} = 0), (x_{ba} \oplus x_{in} = 1)$$

n'est impliquée par ϕ''_t. En conséquence, les dépendances fonctionnelles

$$(\{x_{me}, x_{ba}\} \to x_{in}), (\{x_{me}, x_{in}\} \to x_{ba}), (\{x_{ba}, x_{in}\} \to x_{me})$$

sont toutes des dépendances fonctionnelles minimales vérifiées par $M_t = \mathcal{M}(\phi''_t)$.

Pour terminer, mentionnons au sujet des dépendances fonctionnelles le travail d'Ibaraki *et al.* [IKM99], qui s'y sont intéressés pour les formules de *Horn*, montrant notamment que le problème de la *vérification* est polynomial pour ces formules.

Dans le reste du mémoire, nous accordons une place particulière aux formules affines, car elles n'ont jamais été réellement étudiées dans l'optique de la représentation de connaissances. Nous verrons en particulier qu'elles se prêtent tout particulièrement à l'*approximation* de connaissances et au raisonnement.

Chapitre 4

Autres classes de formules

Sommaire

4.1 Renommages 53
4.1.1 Définitions 53
4.1.2 Renommages et satisfaisabilité 55
4.2 Classes de formules $\mathcal{C}$-renommables 56
4.2.1 Formules FNC Horn-renommables 56
4.2.2 Formules Horn définies positives renommables 57
4.2.3 Formules monotones 58
4.3 Formules FND 59

Nous présentons maintenant les classes de formules que nous considérons dans ce mémoire, mais qui ne satisfont pas tous les critères présentés au chapitre 1. Ces classes diffèrent en effet de celles présentées dans les chapitres 2 et 3 en particulier car elles ne sont pas stables par conjonction, et plus généralement car on ne peut pas les voir comme des réseaux de contraintes. Néanmoins, elles ont toutes de bonnes propriétés algorithmiques. Nous présentons tout d'abord les classes des formules FNC renommables dans l'une des classes précédentes, puis les classes de formules FND exprimant les «négations» de ces classes.

4.1 Renommages

Nous présentons tout d'abord les définitions et principaux outils concernant la notion de *renommage*, qui est indépendante des classes de formules concernées.

4.1.1 Définitions

Les classes de formules «renommables», que nous présentons maintenant, sont par définition des généralisations d'autres classes. Etant donnée une classe $\mathcal{C}$, une formule est en effet dite $\mathcal{C}$-*renommable* si on peut la transformer en une formule de $\mathcal{C}$ en remplaçant certains de ses littéraux par leur négation. Par exemple, la formule de l'exemple du carrefour :

$$\phi_C = (\neg x_{DT} \vee x_{VP}) \wedge (\neg x_{VP} \vee x_{DT}) \wedge (\neg x_{PV} \vee \neg x_{VV} \vee \neg x_{DT}) \wedge (\neg x_{VP} \vee \neg x_{VV})$$

est HDP-renommable, puisqu'en y remplaçant les littéraux formés sur les variables x_{VP} et x_{DT} par leurs littéraux opposés respectifs, on obtient la formule :

$$\phi''_C = (x_{DT} \vee \neg x_{VP}) \wedge (x_{VP} \vee \neg x_{DT}) \wedge (\neg x_{PV} \vee \neg x_{VV} \vee x_{DT}) \wedge (x_{VP} \vee \neg x_{VV})$$

qui est bien de Horn définie positive.

Ainsi, en particulier, les formules de $\mathcal{C}$ sont $\mathcal{C}$-renommables ; de même, les formules de l'éventuelle classe *duale* de $\mathcal{C}$ (voir paragraphe 2.5) sont $\mathcal{C}$-renommables : il suffit de remplacer *tous* les littéraux par leurs opposés. Plus formellement, nous définissons tout d'abord la notion de renommage et l'opération associée.

Définition 4.1 (renommage) *Soit V un ensemble de variables. Un* renommage de V *est une application de V dans $\{0,1\}$.*

Définition 4.2 (renommer) *Soit ϕ une formule propositionnelle, et soit ρ un renommage de $Var(\phi)$. Le* renommé de ϕ par ρ, *noté $\phi_{\oplus\rho}$, est la formule obtenue en remplaçant, pour toute variable $x \in Var(\phi)$ avec $\rho(x) = 1$, toute occurrence de $\neg x$ dans ϕ par x, et toute occurrence de x par $\neg x$, simultanément ; on dit que l'on* renomme x *dans ϕ. Similairement, si V est un ensemble de variables, le* renommé *d'un ensemble d'affectations M à V par un renommage ρ de V, noté $M_{\oplus\rho}$, est l'ensemble d'affectations à V obtenu de M en remplaçant $m[x]$ par $m[x] \oplus 1$ pour tous $m \in M$ et $x \in V$ avec $\rho(x) = 1$.*

Ainsi, on convient que $\rho(x)$ vaut 1 si x doit être renommé, et 0 sinon. Remarquons que les renommages d'un ensemble de variables V sont définis de même que les affectations à V, comme des applications ; lorsqu'il n'y aura pas de confusion possible, nous identifierons donc ces deux objets, et utiliserons indifféremment la notation $\rho(x)$ ou $\rho[x]$ pour les renommages, ou encore parlerons du renommé d'une formule par une *affectation* à ses variables. Comme pour les affectations, nous considérons également comme renommage d'un ensemble de variables V un renommage d'un sur-ensemble de V, en restreignant implicitement son domaine de définition. Enfin, remarquons que les notations $\phi_{\oplus\rho}$ et $M_{\oplus\rho}$ sont cohérentes avec les notations $\phi_{\oplus m}$ et $\mathcal{M}(\phi)_{\oplus m}$ introduites pour les formules affines (chapitre 3, paragraphe 3.3). De façon générale, nous utilisons la lettre ρ, éventuellement primée ou indicée, pour désigner les renommages.

La formule ϕ''_C ci-dessus est donc le *renommé* de la formule ϕ_C par le renommage ρ_C de l'ensemble de variables $V_C = \{x_{PV}, x_{VP}, x_{VV}, x_{DT}\}$ défini par $\rho_C[x_{VP}] = \rho_C[x_{DT}] = 1$ et $\rho_C[x_{PV}] = \rho_C[x_{VV}] = 0$. Similairement, si l'on renomme par ρ_C l'ensemble des modèles de ϕ_C :

$$M_C = \{0000, 0010, 0101, 1000, 1010, 1101\}$$

on obtient l'ensemble d'affectations

$$(M_C)_{\oplus\rho_C} = \{0101, 0111, 0000, 1101, 1111, 1000\}$$

Il est facile de voir que pour toute formule ϕ et tout renommage ρ de $Var(\phi)$, on a $(\phi_{\oplus\rho})_{\oplus\rho} = \phi$, et de même pour les ensembles d'affectations ; ainsi, les formules ϕ_C et ϕ''_C vérifient $\phi''_C = (\phi_C)_{\oplus\rho_C}$ et $\phi_C = (\phi''_C)_{\oplus\rho_C}$.

Enfin, si $\mathcal{C}$ est une classe de formules, nous disons qu'une formule ϕ est *$\mathcal{C}$-renommable* s'il existe un renommage ρ de $Var(\phi)$ tel que la formule $\phi_{\oplus\rho}$ est dans $\mathcal{C}$; nous disons alors que ρ est un renommage *de ϕ dans $\mathcal{C}$*. La formule ϕ_C, par exemple, est HDP-renommable, et ρ_C est un renommage de ϕ_C dans HDP. Remarquons également que ϕ_C est HORN-renommable, puisqu'elle est HDP-renommable, d'une part, mais aussi parce qu'elle est de Horn ; le renommage $\overline{0_{V_C}}$ est

donc un renommage de ϕ_C dans HORN. A l'inverse, la formule de l'exemple de la thèse :

$$\phi_t = \begin{array}{ll} & (x_{qu} \vee x_{ba}) \wedge (\neg x_{qu} \vee \neg x_{ba}) \\ \wedge & (x_{me} \vee \neg x_{ba} \vee x_{in}) \wedge (x_{me} \vee x_{ba} \vee \neg x_{in}) \wedge (\neg x_{me} \vee x_{ba} \vee x_{in}) \\ \wedge & (\neg x_{me} \vee \neg x_{ba} \vee \neg x_{in}) \\ \wedge & (\neg x_{in} \vee x_{ch}) \wedge (x_{in} \vee \neg x_{ch}) \end{array}$$

n'est pas HORN-renommable. En effet, pour la renommer en une formule de Horn il faudrait renommer au moins deux des trois variables x_{me}, x_{ba}, x_{in}, afin de transformer chacune des clauses de la deuxième ligne, $(x_{me} \vee \neg x_{ba} \vee x_{in})$, $(x_{me} \vee x_{ba} \vee \neg x_{in})$ et $(\neg x_{me} \vee x_{ba} \vee x_{in})$, en une clause de Horn ; or, renommer deux de ces variables introduit deux littéraux positifs dans la clause $(\neg x_{me} \vee \neg x_{ba} \vee \neg x_{in})$.

Pour terminer, rappelons que contrairement aux classes de formules présentées dans les chapitres 2 et 3, la classe des formules $\mathcal{C}$-renommables, pour une classe de formules FNC $\mathcal{C}$, n'est en général pas stable par conjonction. En effet, si ϕ_1 et ϕ_2 sont deux formules $\mathcal{C}$-renommables, la formule $\phi_1 \wedge \phi_2$ n'est en général pas $\mathcal{C}$-renommable si ϕ_1 et ϕ_2 ne le sont pas *par le même renommage de* $Var(\phi_1) \cup Var(\phi_2)$. Néanmoins, ces classes de formules peuvent être intéressantes pour la représentation de connaissances. En effet, soit $\mathcal{C}$ une classe de formules FNC définies par la nature de leurs clauses, et soit un système manipulant des connaissances représentées par des formules de $\mathcal{C}$. Si le système doit renommer certaines variables pour intégrer de nouvelles connaissances, utilisant ainsi implicitement la classe des formules $\mathcal{C}$-renommables, il reste par la suite à même de spécifier la *nature* des clauses qui peuvent être ajoutées par conjonction à la base : c'est simplement la nature des clauses des formules de la classe $\mathcal{C}$, modifiée en fonction du renommage utilisé.

4.1.2 Renommages et satisfaisabilité

On voit aisément que renommer une formule ou un ensemble d'affectations se fait en temps linéaire. La principale propriété du renommage est qu'il conserve la satisfaisabilité, et même le nombre de modèles ; donc, si on peut décider la satisfaisabilité des formules d'une classe $\mathcal{C}$ en temps polynomial, on peut décider efficacement la satisfaisabilité d'une classe $\mathcal{C}'$ de formules $\mathcal{C}$-renommables dès que l'on peut calculer efficacement, pour toute formule ϕ de $\mathcal{C}'$, un renommage de ϕ dans $\mathcal{C}$.

Plus généralement, le renommage ne change en rien la signification des formules, et consiste intuitivement à accorder la signification opposée aux variables renommées. Par exemple, lorsque l'on renomme la variable x_{VP} dans la formule ϕ_C de l'exemple du carrefour, sa signification intuitive n'est plus «Le feu pour piétons est vert», mais «Le feu pour piétons est rouge» ; de même, renommer x_{DT} revient à changer sa signification en «Il est interdit de traverser». Ainsi, le renommé ϕ''_C de ϕ_C par ρ_C se lit :

- Si le feu pour piétons est rouge, alors il est interdit de traverser
- S'il est interdit de traverser, alors le feu pour piétons est rouge
- S'il y a des véhicules aux abords du carrefour et si le feu pour véhicules est vert, alors il est interdit de traverser
- Si le feu pour véhicules est vert, alors le feu pour piétons est rouge.

La conservation de la satisfaisabilité par l'opération de renommage est alors une conséquence de la proposition suivante, plus générale et dont la preuve est immédiate.

Proposition 4.1 (renommage et modèles) *Soient ϕ une formule propositionnelle et ρ un renommage de $Var(\phi)$. Alors on a $\mathcal{M}(\phi_{\oplus\rho}) = \mathcal{M}(\phi)_{\oplus\rho}$.*

L'ensemble des modèles de la formule ϕ''_C est donc bien l'ensemble $(M_C)_{\oplus\rho_C}$. On déduit également de la proposition que l'opération de renommage conserve le nombre de modèles d'une formule, ainsi que l'implication et la primalité, en ce sens que si une formule ϕ implique une formule ϕ', et si ρ est un renommage de $Var(\phi) \cup Var(\phi')$, alors $\phi_{\oplus\rho}$ implique $\phi'_{\oplus\rho}$, et de même pour un impliqué premier C de ϕ. Ainsi, la clause $C_C = (\neg x_{VV} \vee \neg x_{DT})$ («Si le feu pour véhicules est vert, on n'a pas le droit de traverser») est un impliqué (premier) de ϕ_C, donc la clause $(C_C)_{\oplus\rho_C} = (\neg x_{VV} \vee x_{DT})$ («Si le feu pour véhicules est vert, il est interdit de traverser») est un impliqué (premier) de la formule $(\phi_C)_{\oplus\rho_C} = \phi''_C$.

Nous déduisons notamment de ces remarques la proposition suivante, que nous formulons de façon générale, mais qui nous intéresse particulièrement pour les classes de formules introduites dans ce chapitre.

Proposition 4.2 (descriptions premières et renommage) *Soit $\mathcal{C}$ une classe de formules FNC telle que pour tout ensemble d'affectations M, M est descriptible dans $\mathcal{C}$ si et seulement si toute formule en FNC et première décrivant M est une formule de $\mathcal{C}$. Alors la classe $\mathcal{C}$-*Ren *des formules $\mathcal{C}$-renommables vérifie la même propriété.*

Bien entendu, de façon générale il est important qu'il existe un algorithme efficace pour décider si une formule donnée est bien $\mathcal{C}$-renommable, c'est-à-dire que la classe des formules $\mathcal{C}$-renommables soit reconnaissable (efficacement). Mais il est également important qu'il existe un algorithme efficace pour, le cas échéant, *calculer* un renommage de cette formule dans la classe $\mathcal{C}$. C'est à ces conditions seulement que l'on peut exploiter toutes les propriétés des renommages, et nous verrons dans le paragraphe 4.2.2 qu'elle ne sont pas toujours remplies.

4.2 Classes de formules $\mathcal{C}$-renommables

Nous présentons maintenant les classes des formules FNC renommables dans les classes présentées dans le chapitre 2. Notons que les classes Bij et Affine sont invariantes par renommage, et que les classes des formules Bij-renommables et Affine-renommables n'ont donc pas réellement de sens.

4.2.1 Formules FNC Horn-renommables

La classe des formules FNC Horn-renommables [Asp80, Héb94], que nous notons Horn-Ren, est donc la classe des formules FNC renommables en une formule de Horn. Remarquons tout de suite que la formule FNC insatisfaisable (()) est dans Horn-Ren, et que cette classe est stable par propagation d'affectations.

La propriété importante de cette classe est qu'on peut la reconnaître en temps polynomial, et même linéaire, en calculant un renommage adéquat le cas échéant (c'est-à-dire un renommage de la formule considérée dans Horn). De fait, le problème `SAT` est également linéaire pour cette classe.

Proposition 4.3 (Horn-renommabilité [Asp80, Héb94]) *On peut décider si une formule FNC donnée ϕ est* Horn*-renommable en temps linéaire, en calculant un renommage de ϕ dans* Horn *le cas échéant.*

Corollaire 4.1 (SAT[HORN-REN]) *On peut décider si une formule FNC* HORN*-renommable donnée est satisfaisable en temps linéaire, en en exhibant un modèle le cas échéant.*

A l'inverse, on sait dire peu de choses sur l'ensemble des modèles d'une formule Horn-renommable, sinon en reformulant les propriétés de clôture de la classe HORN modulo le renommage. Toutefois, le résultat suivant est du plus haut intérêt, puisqu'il affirme en particulier qu'étant donné un ensemble d'affectations M descriptible dans HORN-REN, on peut chercher un renommage de M en un ensemble descriptible dans HORN parmi seulement $|M|$ renommages «candidats» (Boufkhad [Bou98] montre cette propriété dans un cadre différent).

Proposition 4.4 (HORN-renommages [Bou98]) *Un ensemble d'affectations $M \neq \emptyset$ à un ensemble de variables V est descriptible dans* HORN-REN *si et seulement s'il existe un renommage $m \in M$ tel que l'ensemble d'affectations $M_{\oplus m}$ est descriptible dans* HORN.

Preuve Supposons que M est descriptible dans HORN-REN. Alors il existe un renommage ρ de V tel que l'ensemble d'affectations $M_{\oplus\rho}$ est descriptible dans HORN. Supposons tout d'abord que pour toute variable $x \in V$ il existe une affectation $m \in M_{\oplus\rho}$ avec $m[x] = 0$; alors par la proposition 2.2 (chapitre 2) l'affectation $\overline{0_V}$ est dans $M_{\oplus\rho}$, donc il existe une affectation $m \in M$ avec $m \oplus \rho = \overline{0_V}$, c'est-à-dire $m = \rho$. Dans le cas général, si $V' \subseteq V$ est l'ensemble des variables x avec $\forall m \in M_{\oplus\rho}, m[x] = 1$, soit ρ' le renommage de V défini par :

$$\forall x \in V \backslash V', \rho'[x] = \rho[x] \text{ et } \forall x \in V', \rho'[x] = \rho[x] \oplus 1$$

Alors il est facile de voir que l'ensemble d'affectations $M_{\oplus\rho'}$ est descriptible dans HORN et contient l'affectation $\overline{0_V}$, et on en déduit comme précédemment que ρ' est dans M. La réciproque est triviale. □

Cette proposition nous permet par exemple de montrer facilement que l'ensemble d'affectations M_t de l'exemple de la thèse :

$$M_t = \{00111, 01000, 10100, 11011\}$$

n'est pas descriptible dans HORN-REN ; en effet, il suffit de constater qu'aucun renommé de M_t par l'un de ses éléments n'est descriptible dans HORN. En particulier, on peut en déduire que la *formule* ϕ_t n'est pas HORN-renommable.

Remarquons toutefois que la proposition 4.4 ne caractérise pas *tous* les renommages de M dans HORN. Par exemple, tout ensemble d'affectations M de cardinal 1 est descriptible dans HORN, donc son renommé par l'affectation de 0 à toutes les variables est descriptible dans HORN, mais cette affectation n'est en général pas dans M.

4.2.2 Formules Horn définies positives renommables

Contrairement à la classe HORN-REN, et bien qu'elle en soit par définition une sous-classe, la classe HDP-REN des formules FNC renommables en une formule de Horn définie positive n'admet pas d'algorithme de reconnaissance polynomial (sous l'hypothèse P≠NP). Ceci est une conséquence de la proposition suivante.

Proposition 4.5 (HDP-renommabilité) *La reconnaissance des formules FNC* HDP*-renommables est un problème NP-complet, même si on restreint les données du problème aux formules dont toutes les clauses sont de longueur* 3 *ou moins.*

Preuve Le problème est dans NP car un renommage de la formule donnée dans HDP est un témoin convenable. Pour montrer sa complétude, nous lui réduisons le problème NP-complet 1-Parmi-3-SAT [Sch78], qui consiste à déterminer, étant donnée une formule dont toutes les clauses contiennent trois littéraux ou moins, si elle admet un modèle qui satisfait exactement un littéral dans chacune de ses clauses. Soient ϕ une telle formule et m une affectation à $Var(\phi)$, et soit ρ le renommage de $Var(\phi)$ défini par $\forall x \in Var(\phi), \rho(x) = m[x] \oplus 1$; nous montrons que ρ est un renommage de ϕ dans HDP si et seulement si m satisfait exactement un littéral par clause de ϕ. En effet, ρ renomme un littéral négatif $\neg x$ dans ϕ si et seulement m affecte 0 à x, autrement dit si m satisfait ce littéral, et de même ρ *ne* renomme *pas* un littéral positif x dans ϕ si et seulement si m affecte 1 à x, autrement dit si m satisfait x. Finalement, ρ renomme en des littéraux positifs les littéraux satisfaits par m, et en des littéraux négatifs ceux non satisfaits par m ; finalement, ρ renomme ϕ dans HDP si et seulement si m satisfait exactement un littéral par clause de ϕ, donc ϕ est renommable dans HDP si et seulement si c'est une donnée positive pour 1-Parmi-3-SAT, et la NP-complétude de ce dernier problème conclut [Sch78]. □

Nous étudions donc cette classe par souci de complétude principalement. En effet, comme nous l'avons déjà remarqué dans le chapitre 1, du point de vue de la représentation de connaissances il est peu intéressant d'utiliser une classe de formules non reconnaissable. Notons tout de même que le problème SAT restreint à cette classe est trivial, même si ce résultat n'a de fait pas beaucoup d'applications.

Proposition 4.6 (SAT[HDP-Ren]) *Toute formule* HDP*-renommable admet au moins un modèle.*

Du point de vue sémantique, nous pouvons formuler la propriété suivante, parallèle à la proposition 4.4.

Proposition 4.7 (HDP-renommages) *Soit $M \neq \emptyset$ un ensemble d'affectations à un même ensemble de variables V. Alors M est descriptible dans* HDP-Ren *si et seulement s'il existe une affectation $m \in M$ telle que l'ensemble d'affectations $M_{\oplus(m \oplus \overline{1_V})}$ soit descriptible dans* HDP.

Preuve Le sens indirect est trivial. Supposons donc M descriptible dans HDP-Ren, et soit ρ un renommage de V tel que $M_{\oplus\rho}$ est descriptible dans HDP. Alors $M_{\oplus\rho}$ contient l'affectation $\overline{1_V}$, donc on a $\overline{1_V} \oplus \rho \in M$; en notant $\overline{1_V} \oplus \rho = m \in M$, on a bien $\rho = m \oplus \overline{1_V}$. □

Ainsi, puisque la formule ϕ_C de l'exemple du carrefour est HDP-renommable (paragraphe 4.1.1), on sait qu'un de ses modèles $m \in \phi_C$ est tel que le renommage $m \oplus \overline{1_{V_C}}$ la renomme dans HDP ; en effet, le renommage ρ_C exhibé au paragraphe 4.1.1 vérifie $\rho_C = 0101 = 1010 \oplus 1111$, et 1010 est bien un modèle de ϕ_C.

Remarquons donc en particulier que les renommages candidats de M en un ensemble descriptible dans HDP ne sont pas plus nombreux que les affectations dans M, puisque à chaque affectation $m \in M$ correspond au plus un tel renommage. C'est principalement cette propriété que nous utiliserons par la suite.

4.2.3 Formules monotones

Nous présentons enfin la classe des formules Nég-renommables, plus couramment appelées *monotones* ; nous notons cette classe Monot. La plupart des propriétés de cette classe sont similaires à celles de la classe Horn-Ren, dont elle est par ailleurs une sous-classe ; néanmoins,

encore une fois cette sous-classe mérite une attention particulière, par exemple parce qu'elle permet une abduction polynomiale alors que la classe HORN-REN, non (cf. chapitre 11). Comme la classe HORN-REN, cette classe admet un algorithme de reconnaissance linéaire. En effet, une formule FNC ϕ est par définition NÉG-renommable si et seulement si on peut la renommer en une formule négative, c'est-à-dire en une formule ne contenant que des littéraux négatifs. Il suffit donc, pour décider si ϕ est NÉG-renommable, de la parcourir une fois et de décider si chacune de ses variables apparaît toujours avec le même signe. Si c'est le cas, ϕ est bien NÉG-renommable, et il suffit de renommer toutes les variables qui y apparaissent toujours positivement pour obtenir une formule négative. Dans le cas contraire, ϕ n'est pas monotone.

Ainsi, on peut voir que la formule ϕ_C de l'exemple du carrefour n'est pas monotone ; en effet, les littéraux positif *et* négatif formés sur la variable x_{VP}, par exemple, y apparaissent.

Proposition 4.8 (NÉG-renommabilité) *On peut décider si une formule FNC donnée ϕ est* NÉG*-renommable (monotone) en temps linéaire, en calculant un renommage de ϕ dans* NÉG *le cas échéant.*

Corollaire 4.2 (SAT[MONOT]) *On peut décider si une formule monotone donnée est satisfaisable en temps linéaire, en en exhibant un modèle le cas échéant.*

De façon générale, nous utilisons le terme *monotone*, plutôt que NÉG*-renommable*, car c'est le plus courant[1]. Notons que la classe MONOT est stable par propagation d'affectations, et que la formule FNC insatisfaisable (()) est monotone.

Encore une fois, les propriétés sémantiques connues de cette classe ne sont pas nombreuses, mais nous pouvons formuler la suivante, toujours similaire à celle de la classe HORN-REN, et donc la preuve découle directement du fait qu'une formule négative ϕ satisfaisable admet nécessairement l'affectation $\overline{0_{Var(\phi)}}$ pour modèle.

Proposition 4.9 (NÉG-renommages) *Soit $M \neq \emptyset$ un ensemble d'affectations. Alors M est descriptible dans* MONOT *si et seulement s'il existe un renommage $m \in M$ tel que l'ensemble d'affectations $M_{\oplus m}$ est descriptible dans* NÉG.

Encore une fois, nous pouvons donc vérifier avec cette proposition que l'ensemble d'affectations M_C de l'exemple du carrefour n'est pas descriptible dans MONOT.

Enfin, remarquons que cette classe, à l'instar de la classe NÉG, vérifie une propriété particulière. En effet, il est facile de voir, en copiant la preuve de la proposition 2.11 du chapitre 2, qu'il existe une unique formule FNC première décrivant un ensemble d'affectations donné descriptible dans MONOT ; cette formule est monotone et contient tous ses impliqués premiers.

4.3 Formules FND

Nous présentons enfin les classes de formules FND que nous étudions dans ce mémoire. Comme nous l'avons annoncé dans le chapitre 1, nous nous intéressons aux classes de formules FND définies en fonction d'une des classes $\mathcal{C}$ de formules FNC présentées dans les chapitres 2 et

[1] Certains auteurs utilisent toutefois ce terme pour désigner les formules *positives* [Bsh95, DMP99] ; si on ne veut vraiment pas appeler ces dernières «formules positives», une meilleure solution serait peut-être de les qualifier de «croissantes». Hammer et Kogan, par exemple, utilisent la même terminologie que nous [HK93, HK95].

3 et dans le paragraphe 4.2 de ce chapitre, car nous considérons l'utilité des formules FND dans la représentation implicite de la négation de bases de connaissances.

Rappelons tout d'abord que la classe entière des formules FND admet un algorithme polynomial pour SAT ; le problème est même trivial, puisque la seule formule FND insatisfaisable est celle ne contenant aucun terme (voir les préliminaires généraux).

Proposition 4.10 (SAT[FND]) *On peut décider si une formule FND donnée est satisfaisable en temps constant, et on peut exhiber un de ses modèles, le cas échéant, en temps linéaire.*

Ainsi, la formule FND de l'exemple du carrefour :

$$\phi'_C = (\neg x_{PV} \wedge \neg x_{VP} \wedge \neg x_{DT}) \vee (\neg x_{VP} \wedge x_{VV} \wedge \neg x_{DT}) \vee (x_{VP} \wedge \neg x_{VV} \wedge x_{DT}) \vee (\neg x_{VP} \wedge \neg x_{VV} \wedge \neg x_{DT})$$

est satisfaisable, et pour calculer un de ses modèles, il suffit de considérer l'un quelconque de ses termes, par exemple le second, de calculer son unique modèle, ici l'affectation 010 à l'ensemble de variables $\{x_{VP}, x_{VV}, x_{DT}\}$), puis de le prolonger à $Var(\phi'_C)$ en affectant une valeur quelconque aux autres variables. On obtient ainsi, par exemple, l'affectation 0010 à $V_C = \{x_{PV}, x_{VP}, x_{VV}, x_{DT}\}$.

En fait, pour les formules FND le problème correspondant au problème SAT des formules FNC est le problème TAUT, qui consiste à décider si une formule donnée ϕ est tautologique, autrement dit si l'on a $\mathcal{M}(\phi) = \{0,1\}^{Var(\phi)}$. En effet, on voit aisément qu'une formule est tautologique si et seulement si sa négation est insatisfaisable ; or, on peut passer d'une formule FND ϕ à une formule FNC logiquement équivalente à sa négation en temps linéaire via la transformation $\phi \mapsto Neg(\phi)$, et de fait le problème TAUT est coNP-complet pour la classe des formules FND (et trivial pour la classe des formules FNC).

Les classes de formules FND de la forme $\mathcal{C}$-FND, pour $\mathcal{C}$ l'une des classes de formules FNC présentées précédemment, admettront donc par construction un algorithme polynomial pour TAUT. Nous définissons tout d'abord formellement les telles classes, puis nous les listons ainsi que leurs principales propriétés. Notons que nous ne définissons pas la classe $\mathcal{C}$-FND comme la «négation» de la classe $\mathcal{C}$, mais comme la négation de la classe *duale* de $\mathcal{C}$; nous reviendrons sur ce point plus loin.

Définition 4.3 ($\mathcal{C}$-FND) *Soit $\mathcal{C}$ une classe de formules FNC. La classe des formules $\mathcal{C}$-FND est la classe des formules FND ϕ telles que la formule FNC obtenue de ϕ en inversant la conjonction et la disjonction est dans $\mathcal{C}$.*

Nous considérons ainsi les classes de formules FND suivantes :

- la classe Horn-FND des formules FND de Horn, dont chaque terme contient au plus un littéral positif, et sa classe duale Anti-Horn-FND
- la classe HDP-FND des formules FND Horn définies positives, dont chaque terme contient exactement un littéral positif, et sa classe duale Anti-HDP-FND
- la classe Nég-FND des formules FND négatives, dont chaque terme ne contient que des littéraux négatifs, et sa classe duale Pos-FND
- la classe Bij-FND des formules FND bijonctives, ou 2FND, dont chaque terme contient au plus deux littéraux
- la classe Horn-Ren-FND des formules FND renommables dans la classe Horn-FND
- la classe HDP-Ren-FND des formules FND renommables dans la classe HDP-FND
- la classe Monot-FND des formules FND renommables dans la classe Nég-FND.

Enfin, nous noterons DISJAFFINE la classe des disjonctions d'équations affines, pendant des formules affines ; notons que ces dernières ne sont pas des formules FND.

Ainsi, si l'on considère à nouveau la formule en FNC ϕ_C de l'exemple du carrefour, la formule $Neg(\phi_C)$ est la formule suivante :

$$(x_{DT} \wedge \neg x_{VP}) \vee (x_{VP} \wedge \neg x_{DT}) \vee (x_{PV} \wedge x_{VV} \wedge x_{DT}) \vee (x_{VP} \wedge x_{VV})$$

qui est une formule de la classe ANTI-HORN-FND ainsi que de la classe HDP-REN-FND (via le renommage de x_{PV} et x_{VV}), mais pas, par exemple, de la classe BIJ-FND.

Signalons que les définitions diffèrent dans la littérature ; ainsi, certains auteurs qualifient de «FND de Horn» une formule FND dont chaque terme contient au plus un littéral *négatif*, signifiant ainsi qu'une formule FND ϕ est de Horn si et seulement si la formule FNC $Neg(\phi)$ est de Horn [MHI99] ; nous ne suivons pas cette nomenclature, principalement parce qu'elle est peu intuitive si on la copie pour les formules négatives : une formule FND négative serait alors une formule FND dont chaque terme ne contient que des littéraux positifs ! Notons néanmoins que ces classes sont bien celles correspondant aux négations des classes de formules FNC présentées, puisque la classe des négations des formules FNC de Horn, par exemple, est la classe ANTI-HORN-FND, et ainsi de suite.

Pour la plupart des problèmes traités dans la partie II, consacrée à l'acquisition de connaissances, nous ne nous attarderons pas sur les classes de formules FND, puisque nous n'envisageons pas de *représenter* explicitement des connaissances en FND, mais plutôt de *raisonner* avec des formules en FND, lorsqu'un problème doit être résolu en fonction de la négation d'une base de connaissances. Nous mentionnerons tout de même quelques résultats intéressants, mais nous reviendrons principalement sur ces classes de formules dans la partie III, consacrée au raisonnement.

Chapitre 5

Les classes dans leur globalité

Sommaire

5.1 Classes de Schaefer 63
5.1.1 Dichotomie du problème SAT-généralisé 65
5.1.2 Dichotomie du problème Inverse-SAT 66
5.2 Intersections des classes 67
5.3 Résumé des propriétés des classes 69

Nous terminons cette partie en considérant les classes de formules présentées dans les chapitres précédents par rapport à l'ensemble des classes de formules que l'on peut définir. A la lumière des définitions des classes et de deux résultats de dichotomie très importants, nous renforçons ainsi nos motivations pour étudier ces classes en particulier. Nous étudions également les intersections de ces classes, afin de les situer les unes par rapport aux autres, et enfin nous résumons leurs principales propriétés.

5.1 Classes de Schaefer

Les deux résultats de dichotomie qui motivent notre choix de classes de formules, ainsi d'ailleurs que, souvent, leur étude dans la littérature, proviennent de [Sch78] et [KS98]. Pour les présenter, nous avons besoin de la notion de *S-formule*, introduite par Schaefer [Sch78] et très proche de la notion de réseau de contraintes présentée brièvement dans le chapitre 1. Plus précisément, les S-formules peuvent être vues comme des *Problèmes de Satisfaction de Contraintes booléens* (où les contraintes sont choisies dans l'ensemble S). De façon générale, nous renvoyons le lecteur à l'ouvrage de Creignou *et al.* [CKS01] pour plus de détails sur les résultats de dichotomie à propos des Problèmes de Satisfaction de Contraintes booléens.

Etant donné un entier $k \in \mathbb{N}$, on appelle *relation d'arité k* un ensemble de k-uplets $R \subseteq \{0,1\}^k$; intuitivement, un tel ensemble représente le support d'une fonction booléenne de $\{0,1\}^k$ dans $\{0,1\}$, c'est-à-dire l'ensemble des valeurs en lesquelles elle vaut 1 : par exemple, la relation $R_{eg} = \{00, 11\}$, d'arité 2, représente la fonction «égalité». Etant donnés une relation R d'arité k et un ensemble ordonné $V = (x_{i_1}, x_{i_2}, \ldots, x_{i_k})$ de k variables toutes différentes, le couple (R, V) est appelé une *contrainte* ; intuitivement, c'est un ensemble d'affectations à V : par exemple, la contrainte $(R_{eg}, (x_{VP}, x_{DT}))$ correspond à l'ensemble d'affectations $\{00, 11\}$ aux variables (x_{VP}, x_{DT}) ; notons que l'ordre des variables est important si la relation n'est pas symétrique (par exemple, la relation $\{000, 010\}$). Enfin, étant donné un ensemble S *fini* de relations,

d'arités non nécessairement égales, et un ensemble fini de variables V, on appelle *S-formule formée sur V* tout ensemble ϕ de contraintes de la forme (R_i, V_i) où pour tout i, $R_i \in S$ est une relation d'arité k_i et V_i, un sous-ensemble ordonné de V de cardinal k_i. Par exemple, en notant $R_{3,neg}$ la relation d'arité 3 :

$$\{000, 001, 010, 011, 100, 101, 110\}$$

la formule :

$$\phi'''_C = \{(R_{eg}, (x_{DT}, x_{VP})), (R_{3,neg}, (x_{PV}, x_{VV}, x_{DT})), (\{00, 01, 10\}, (x_{VP}, x_{VV}))\}$$

est une S_C-formule formée sur l'ensemble de variables $V_C = \{x_{PV}, x_{VP}, x_{VV}, x_{DT}\}$, avec $S_C = \{R_{eg}, R_{3,neg}, \{00, 01, 10\}\}$.

Etant donnés un ensemble S de relations et un ensemble V de variables, on dit alors qu'une affectation m à V *satisfait* une S-formule $\phi = \{(R_i, V_i) \mid i = 1, 2, \ldots, |\phi|\}$ formée sur V si, pour tout $i = 1, 2, \ldots, |\phi|$ il existe un k_i-uplet $m_i \in R_i$, vu comme une affectation à V_i, dont m est un prolongement sur V. Par exemple, l'ensemble des affectations satisfaisant la S-formule ϕ'''_C ci-dessus est exactement l'ensemble d'affectations M_C de l'exemple du carrefour.

Comme on peut le voir, les S-formules donnent donc une manière de coder, en particulier, les classes de formules FNC définies par la nature de leurs clauses, lorsque les «natures» autorisées sont en nombre fini. Ainsi, la contrainte $(R_{3,neg}, (x_{PV}, x_{VV}, x_{DT}))$ code la clause $(\neg x_{PV} \vee \neg x_{VV} \vee \neg x_{DT})$, en ce sens que les affectations à $\{x_{PV}, x_{VV}, x_{DT}\}$ satisfaisant l'une et l'autre sont les mêmes. Similairement, la formule ϕ'''_C ci-dessus code dans ce formalisme la formule ϕ_C de l'exemple du carrefour.

Modulo cet encodage, la classe des formules FNC de Horn dont les clauses sont de cardinal au plus k, pour un entier k fixé, est la classe des $S_{k\text{HORN}}$-formules, où $S_{k\text{HORN}}$ est par exemple l'ensemble :

$$\{\emptyset, R_{1,hdp} = \{1\}, R_{1,neg} = \{0\}, R_{2,hdp} = \{00, 01, 11\}, R_{2,neg} = \{00, 01, 10\}, \ldots, R_{k,hdp}, R_{k,neg}\}$$

contenant la relation vide (qui code la clause vide) et, pour tout $i \in \{1, 2, \ldots, k\}$ non nul, deux relations d'arité i servant de «modèles», l'une pour les clauses de Horn définies positives contenant i littéraux $(R_{i,hdp})$, et l'autre pour les clauses négatives de même cardinal $(R_{i,neg})$. La S_C formule ϕ'''_C ci-dessus admet alors les mêmes affectations (à V_C) satisfaisantes que la $S_{3\text{HORN}}$-formule :

$$\{(R_{2,hdp}, (x_{DT}, x_{VP})), (R_{2,hdp}, (x_{VP}, x_{DT})), (R_{3,neg}, (x_{PV}, x_{VV}, x_{DT})), (R_{2,neg}, (x_{VP}, x_{VV}))\}$$

Similairement, la classe BIJ est la classe des S_{BIJ}-formules, où S_{BIJ} est par exemple l'ensemble de relations :

$$S_{\text{BIJ}} = \{\emptyset, \{0\}, \{1\}, \{00, 01, 10\}, \{00, 01, 11\}, \{00, 10, 11\}, \{01, 10, 11\}\}$$

Remarquons que le formalisme des S-formules ne permet pas d'encoder les formules $\mathcal{C}$-renommables pour une classe donnée $\mathcal{C}$, par exemple ; mais puisque nous nous intéressons aux formules $\mathcal{C}$-renommables parce qu'elles possèdent des propriétés proches de celles de la classe $\mathcal{C}$, nous étudions l'intérêt de la classe des formules $\mathcal{C}$-renommables en fonction de l'intérêt de la classe $\mathcal{C}$.

5.1.1 Dichotomie du problème SAT-généralisé

Nous pouvons maintenant rappeler les deux résultats de dichotomie annoncés. Nous commençons par le premier dans l'ordre chronologique, qui motive notamment notre choix de classes vis-à-vis du *raisonnement*.

Ce résultat concerne le problème SAT dans le formalisme des S-formules, nommé *problème SAT-généralisé* par Schaefer. Pour S un ensemble (toujours *fini*) de relations, le problème SAT-Gen[S] est le suivant.

Problème 5.1 (SAT-Gen[S])

Donnée : *Une S-formule ϕ formée sur un ensemble de variables V*
Question : *Existe-t-il une affectation à V satisfaisant ϕ ?*

Notons que cette définition fournit un problème différent pour chaque ensemble S de relations. En conséquence, la taille de l'ensemble S est considérée comme constante lorsque l'on étudie la complexité du problème SAT-Gen[S] ; cette complexité est donc mesurée en fonction de la taille de la S-formule donnée ϕ.

Le résultat de dichotomie de Schaefer est alors le suivant ; pour R une relation, nous disons que R est *de Horn* si toute contrainte formée avec R, vue comme un ensemble d'affectations, est descriptible dans HORN, et de même pour les autres classes de formules ; par exemple, la relation $\{000, 010, 100, 101, 111\}$ est de Horn, car la contrainte qu'elle permet de former sur un ensemble ordonné (x, x', x''), quelconque, de variables est l'ensemble des modèles de la formule FNC de Horn $(x \vee \neg x'') \wedge (\neg x \vee \neg x' \vee x'')$, tandis que la relation $\{01, 10\}$ n'est pas de Horn, puisqu'une contrainte C formée avec elle, vue comme un ensemble d'affectations, ne vérifie pas $C = Cl_{\text{HORN}}(C)$ (cf. chapitre 2, proposition 2.2).

Proposition 5.1 ([Sch78]) *Soit S un ensemble fini de relations. Le problème* SAT-Gen[S] *est polynomial si :*

- *toutes les relations de S contiennent le vecteur tout à 0,* ou si
- *toutes les relations de S contiennent le vecteur tout à 1,* ou si
- *toutes les relations de S sont de Horn,* ou si
- *toutes les relations de S sont anti-Horn,* ou si
- *toutes les relations de S sont 2FNC,* ou si
- *toutes les relations de S sont affines.*

Dans tous les autres cas, le problème SAT-Gen[S] *est NP-complet.*

Cette première proposition est notre principale motivation pour considérer que les classes de formules FNC définies par la nature de leurs clauses et présentées dans les chapitres précédents sont les seules réellement intéressantes pour le *raisonnement*. En effet, la traitabilité du problème SAT pour une classe de formules donnée $\mathcal{C}$ est, comme nous l'avons évoqué dans le chapitre 1, une condition nécessaire à la traitabilité de la plupart des problèmes de raisonnement pour $\mathcal{C}$; nous reviendrons longuement sur ce fait dans la partie III.

Notons également que les deux premières classes de S-formules de la proposition 5.1 sont triviales : on sait que dans ces deux cas, une S-formule donnée, formée sur un ensemble de variables V, est satisfaisable parce que l'affectation $\overline{0_V}$ (resp. $\overline{1_V}$) la satisfait ; nous n'avons pas d'autre indication sur la structure des formules de ces classes. Ainsi, un récent résultat de dichotomie pour

le problème consistant à décider l'*équivalence logique* entre deux S-formules [BHRV02] montre que ce problème est coNP-complet pour tout ensemble S fini, sauf pour les cas où les relations de S sont toutes de Horn, toutes anti-Horn, toutes 2FNC ou toutes affines. Les deux premiers cas du résultat de Schaefer ne sont plus polynomiaux en général ; nous verrons de même qu'ils ne sont pas polynomiaux en général pour le problème `Inv-SAT[`S`]` (paragraphe 5.1.2).

En revanche, concernant les autres classes, la proposition 5.1 nous dit essentiellement que les classes de formules définies par la nature de leurs clauses, traitables pour `SAT` et *maximales* (au sens de l'inclusion de classes) sont les classes HORN (incluant les classes HDP et NÉG), ANTI-HORN, BIJ et AFFINE.

Bien entendu, la proposition n'est formulée que pour des ensembles *finis* de relations, et ne concerne donc pas, *a priori*, toutes les classes de formules FNC définies en nature. Néammoins, nous savons déjà que les classes de formules HORN, ANTI-HORN, BIJ et AFFINE admettent des algorithmes polynomiaux pour `SAT` ; il en va de même pour les classes des formules dont chaque clause contient au moins un littéral positif (resp. négatif), et bien entendu pour les sous-classes de toutes ces classes de formules, puisque étant donnée une formule d'une telle sous-classe, on peut reconnaître efficacement qu'elle fait partie de la sur-classe correspondante (toutes étant reconnaissables) et utiliser un algorithme efficace pour cette dernière. Notons plus généralement que les cas polynomiaux de la proposition de Schaefer s'*uniformisent*, c'est-à-dire qu'ils restent polynomiaux si S n'est plus un *paramètre* du problème, mais fait partie de la donnée [KV00] ; cela provient principalement du fait que l'on peut *décrire* efficacement une relation de Horn donnée en une formule de Horn, et de même pour les autres classes (nous reviendrons sur ce point dans le paragraphe 5.1.2, mais surtout dans le chapitre 7).

A l'inverse, soit $\mathcal{C}$ une classe de formules définie en nature et non incluse dans l'une des classes précédentes. Alors, en restreignant les «modèles» de clauses autorisés par $\mathcal{C}$ à un nombre fini d'entre eux, on peut obtenir un ensemble S de relations (les ensembles de k-uplets «satisfaisant» ces modèles de clauses) pour lequel le problème `SAT-Gen[`S`]` est NP-complet. Le problème `SAT` pour la classe $\mathcal{C}$ étant par construction d'autant plus difficile, il est également NP-complet, et finalement on voit que la proposition de Schaefer est valable dans le formalisme des formules FNC.

5.1.2 Dichotomie du problème Inverse-SAT

Le deuxième résultat de dichotomie [KS98] que nous rappelons fournit quant à lui notre principale motivation pour considérer que les classes de formules FNC, définies par la nature de leurs clauses, présentées auparavant sont les seules réellement intéressantes pour l'*acquisition de connaissances*.

Cette proposition concerne en effet la dichotomie du problème *Inverse-SAT* ; pour S un ensemble fini de relations, le problème `Inv-SAT[`S`]` est le suivant.

Problème 5.2 (`Inv-SAT[`S`]`)

Donnée : *Un ensemble M d'affectations à un ensemble de variables V*
Question : *Existe-t-il une S-formule dont M soit exactement l'ensemble d'affectations satisfaisantes ?*

Encore une fois, cette définition fournit un problème différent pour chaque ensemble de relations, et sa complexité sera mesurée en fonction de la taille de la donnée M, en considérant la taille de

S comme constante. Le résultat de dichotomie de Kavvadias et Sideri est alors le suivant.

Proposition 5.2 ([KS98]) *Soit S un ensemble fini de relations. Le problème* `Inv-SAT[`S`]` *est polynomial si :*
- *toutes les relations de S sont de Horn,* ou si
- *toutes les relations de S sont anti-Horn,* ou si
- *toutes les relations de S sont $2FNC$,* ou si
- *toutes les relations de S sont affines.*

Dans tous les autres cas, il est coNP-complet.

Cette proposition détermine donc les seules classes de S-formules dans lesquelles on peut décrire efficacement un ensemble d'affectations. Même si cette problématique de description ne couvre pas tous les problèmes étudiés dans la partie II, elle nous paraît centrale, puisqu'elle correspond à la tâche la plus naturelle de l'acquisition de connaissances dans un langage de représentation donné, celle consistant à acquérir des connaissances *exactement*.

En revanche, contrairement à la proposition 5.1, la proposition 5.2 n'est plus forcément valide dans le formalisme des classes de formules définies en nature. En effet, si S et S' sont deux ensembles de relations avec $S \subseteq S'$, le problème `Inv-SAT[`S'`]` n'est pas plus général que le problème `Inv-SAT[`S`]`, puisque S n'impose pas de restriction sur la *donnée* du problème, mais paramètre sa *sortie* ; ceci empêche d'utiliser pour une classe $\mathcal{C}$ un algorithme pour une sur-classe de $\mathcal{C}$. Nous pensons néanmoins que le résultat pour les S-formules motive l'étude des classes présentées précédemment, et *seulement* de celles-là. De plus, de façon générale il est naturel de considérer qu'un bon langage de représentation de connaissances est un langage permettant *et* un raisonnement *et* une acquisition de connaissances efficaces.

5.2 Intersections des classes

Nous donnons enfin plusieurs caractérisations des intersections entre les classes de formules qui nous intéressent. Ces caractérisations permettent en particulier de mesurer ce qu'apporte chaque classe de formules par rapport aux autres, autrement dit l'ensemble des connaissances qu'elle seule permet de représenter.

La proposition suivante est classique, et montre que la classe des formules FNC Horn-renommables généralise toutes les autres classes de formules FNC qui nous intéressent ; rappelons néanmoins que nous étudions malgré tout certaines de ses sous-classes car elles possèdent des propriétés que ne possède pas la classe Horn-Ren dans sa globalité.

Proposition 5.3 (Bij vs. Horn-Ren [Asp80, corollaire 1]) *Soit M un ensemble d'affectations à un ensemble de variables V. Si M est descriptible dans* Bij*, alors M est descriptible dans* Horn-Ren.

Preuve Si M est vide le résultat est trivial. Supposons donc $M \neq \emptyset$, et soient ϕ une description de M dans Bij et $m \in M$. Alors m satisfait au moins un littéral par clause de ϕ. Considérons m comme un renommage de V, et soit ℓ un littéral de ϕ satisfait par m ; alors par construction m renomme ℓ si et seulement si ℓ est positif, donc la formule $\phi_{\oplus m}$ contient au moins un littéral négatif par clause. Puisque toutes les clauses de ϕ contiennent au plus deux littéraux, ϕ contient au plus un littéral positif par clause et est donc de Horn. □

Notons que, comme le montre la preuve de cette proposition, cette relation est même plus précise, puisque toute *formule* 2FNC satisfaisable est HORN-renommable. Nous ne nous attardons cependant pas sur ce fait, puisque nous nous intéressons dans ce paragraphe à la *sémantique* des classes, plusieurs formules pouvant représenter la même connaissance.

Donc, toutes les classes de formules FNC qui nous intéressent sont des sous-classes de la classe HORN-REN (c'est vrai pour les autres classes par définition) ; seule la classe AFFINE, parmi les classes qui nous intéressent pour représenter *explicitement* des connaissances, n'est pas concernée par cette proposition. Nous montrons maintenant que cette classe permet en effet de représenter des connaissances que la classe HORN-REN ne permet pas de représenter.

Proposition 5.4 (AFFINE ∩ HORN-REN) *Soit M un ensemble d'affectations à un ensemble de variables V. Si M est descriptible dans* HORN-REN *et dans* AFFINE, *alors M est également descriptible dans* BIJ.

Preuve Soit ρ un renommage de V tel que l'ensemble d'affectations $\rho_{\oplus M}$ soit descriptible dans HORN. Comme la classe des formules affines est invariante par renommage, cet ensemble est également descriptible dans AFFINE. En vertu des propositions 2.2 (chapitre 2) et 3.6 (chapitre 3), on a donc :

$$\forall m, m' \in M, m \wedge m' \in M \text{ et } \forall m, m', m'' \in M, m \oplus m' \oplus m'' \in M$$

En combinant ces deux propriétés, on obtient :

$$\forall m, m', m'' \in M, (m \wedge m') \oplus (m \wedge m'') \oplus (m' \wedge m'') \in M$$

Or il est facile de voir que l'on a :

$$\forall b, b', b'' \in \{0,1\}, (b \wedge b') \oplus (b \wedge b'') \oplus (b' \wedge b'') = maj(b, b', b'')$$

et la proposition 2.13 (chapitre 2) conclut. □

Cette proposition nous apprend donc que toute connaissance représentable par une formule affine n'est représentable *que* par des formules affines (parmi les classes de formules qui nous intéressent), sauf si elle est représentable par une formule 2FNC. Or, il est facile de trouver des formules affines auxquelles il n'existe aucune formule 2FNC logiquement équivalente. Il suffit par exemple de considérer les formules réduites à une équation linéaire impliquant au moins trois variables, celles réduites à deux équations sur trois variables chacune et impliquant des ensembles de variables disjoints, ou possédant une seule variable en commun, et ainsi de suite. Plus anecdotiquement, nous avons vu (chapitre 4, paragraphe 4.2.1) que l'ensemble d'affectations M_t de l'exemple de la thèse n'était pas descriptible dans HORN-REN.

Nous terminons ce paragraphe en discutant le lien entre la classe des formules de Horn et celle des formules de Horn définies positives. Nous avons en effet remarqué au chapitre 2 que les classes HDP et NÉG étaient des sous-classes de la classe HORN, mais dans le cas de la classe HDP nous pouvons préciser cette inclusion.

Proposition 5.5 (HORN vs. HDP) *Pour tout ensemble de variables V, les ensembles d'affectations à V qui sont descriptibles dans* HDP *sont en bijection avec ceux qui sont descriptibles dans* HORN\HDP.

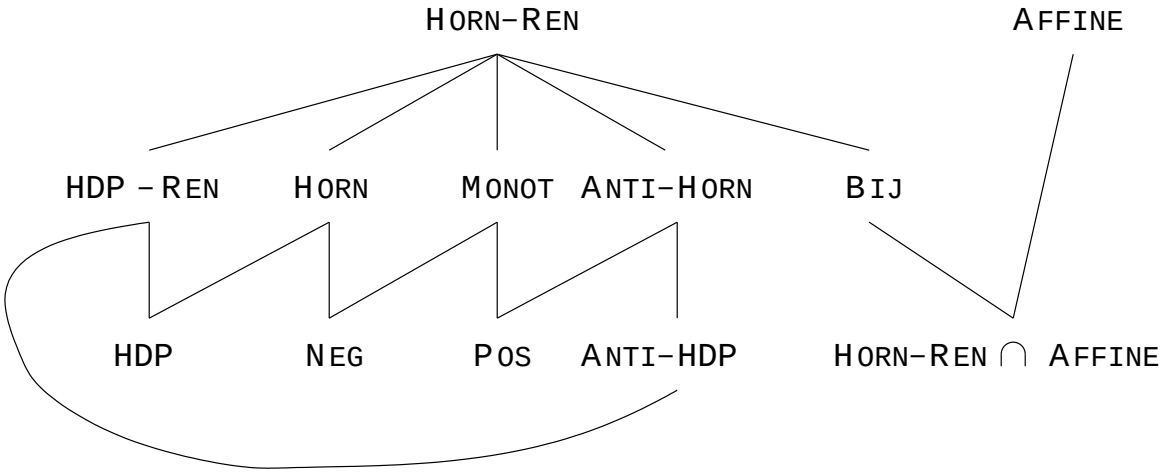

FIG. 5.1 – Les relations d'inclusion entre les classes d'ensembles d'affectations descriptibles dans les différentes classes de formules

Preuve La proposition 2.5 (chapitre 2) montre que la fonction qui associe à un ensemble d'affectations M descriptible dans HDP l'ensemble $M \backslash \{\overline{1_V}\}$ est bien une telle bijection. □

Rappelons enfin que pour un ensemble de variables V, seul l'ensemble d'affectations $\{0,1\}^V$ est descriptible à la fois dans HDP et dans NÉG, et donc que ces deux classes permettent de représenter des bases de connaissances différentes.

Pour résumer ce paragraphe, les relations d'inclusion entre les classes d'ensembles d'affectations descriptibles dans chacune des classes de formules FNC (et affines) qui nous intéressent sont représentées graphiquement sur la figure 5.1 (les arêtes reliant les classes liées par une relation d'inclusion). En d'autres termes, la figure 5.1 compare l'*expressivité* des différentes classes.

5.3 Résumé des propriétés des classes

Nous récapitulons les propriétés des classes de formules FNC présentées dans cette partie dans la table 5.1. Cette table résume la complexité du problème `SAT` pour chacune de ces classes $\mathcal{C}$, ainsi que les propriétés de clôture des ensembles de modèles. Rappelons également que toutes ces classes, sauf la classe HDP-REN, sont reconnaissables en temps linéaire, c'est-à-dire que l'on peut décider en temps linéaire si une formule donnée en fait partie. Notons même que, sauf pour la classe HORN-REN, cette reconnaissance nécessite un seul parcours de la formule. Enfin, rappelons que pour toutes les classes $\mathcal{C}$ de formules FNC présentées, une formule première est dans $\mathcal{C}$ si et seulement si l'ensemble de ses modèles y est descriptible. Nous utiliserons cette propriété à maintes reprises dans la suite du mémoire.

Nous ne résumons pas ici les propriétés des classes de formules FND présentées dans le chapitre 4. Rappelons simplement que le problème `SAT` est trivial pour la classe entière des formules FND ; pour les autres propriétés, toutes sont duales de celles des formules FNC de la classe correspondante. Ainsi, la classe HORN-FND a pour `TAUT` la même complexité (linéaire) que la classe de formules FNC ANTI-HORN a pour `SAT`, puisque à partir d'une formule $\phi \in$ HORN-FND on peut calculer en temps linéaire la formule $Neg(\phi) \in$ ANTI-HORN, logiquement

Classe $\mathcal{C}$	SAT[$\mathcal{C}$]	Clôture $Cl_{\mathcal{C}}(.)$
HORN	linéaire	$m, m' \in M \Rightarrow m \wedge m' \in M$
HDP	$\overline{1_{Var(\phi)}} \in \mathcal{M}(\phi)$	$\overline{1_V} \in M$ et $(m, m' \in M \Rightarrow m \wedge m' \in M)$
NÉG	un parcours	$(m \in M, m' \prec m) \Rightarrow m' \in M$
BIJ	linéaire	$m, m', m'' \in M \Rightarrow maj(m, m', m'') \in M$
ANTI-HORN	linéaire	$m, m' \in M \Rightarrow m \vee m' \in M$
ANTI-HDP	$\overline{0_{Var(\phi)}} \in \mathcal{M}(\phi)$	$\overline{0_V} \in M$ et $(m, m' \in M \Rightarrow m \vee m' \in M)$
POS	un parcours	$(m \in M, m' \succ m) \Rightarrow m' \in M$
AFFINE	$O(\lvert\phi\rvert^2 \lvert Var(\phi)\rvert)$	$m, m', m'' \in M \Rightarrow m \oplus m' \oplus m'' \in M$
HORN-REN	linéaire	$\exists m \in M, M_{\oplus m} = Cl_{\text{HORN}}(M_{\oplus m})$
HDP-REN	au moins un modèle	$\exists m \in M, M_{\oplus(m \oplus \overline{1_V})} = Cl_{\text{HDP}}(M_{\oplus(m \oplus \overline{1_V})})$
MONOT	un parcours	$\exists m \in M, M_{\oplus m} = Cl_{\text{NÉG}}(M_{\oplus m})$

TAB. 5.1 – Resumé des propriétés des classes de formules FNC et affines

équivalente à la négation de ϕ, et réciproquement. De même, cette classe est caractérisée par la propriété de clôture «duale» de celle qui caractérise la classe ANTI-HORN, autrement dit un ensemble M d'affectations est descriptible dans HORN-FND si et seulement si on a :

$$m, m' \notin M \Rightarrow m \vee m' \notin M$$

ce qu'on voit aisément en considérant le complémentaire de M dans $\{0,1\}^V$. Les résultats correspondants pour les autres classes de formules FND ainsi que pour la classe DISJAFFINE se déduisent de même. Enfin, notons qu'en considérant une description en FNC du complémentaire de M, on voit facilement qu'une formule FND *première* est dans l'une des classes de formules FND présentées si et seulement si l'ensemble de ses modèles y est descriptible.

Deuxième partie

Acquisition de connaissances

Chapitre 6

Introduction

Sommaire

6.1 Présentation et motivations 73
6.2 Problématiques étudiées 74

Nous nous intéressons dans cette seconde partie aux problématiques d'*acquisition de connaissances* dans les classes de formules présentées dans la partie I. Les problématiques précises que nous étudions sont définies dans chaque chapitre, mais nous en donnons ici un aperçu global. Nous motivons tout d'abord leur étude, puis donnons une définition informelle de chacune.

6.1 Présentation et motivations

D'une façon générale, les problématiques que nous traitons dans cette partie sont relatives à l'acquisition de connaissances *à partir d'exemples* [RK91, chapitre 17], [RN95, chapitre 18]. Nous serons donc toujours dans une situation où la donnée est un ensemble d'affectations à un même ensemble de variables, diversement présenté ou accessible, ayant également des significations diverses, mais formalisant des exemples d'un concept ou des situations envisageables dans un petit monde donné. Nous chercherons alors à intégrer ces exemples à une base de connaissances, sous la forme d'une formule d'une certaine classe ; cette formule formalisera donc la définition du concept concerné, ou la représentation en intension de l'ensemble des situations envisageables dans le petit monde, et admettra de fait l'ensemble d'affectations donné, ou un ensemble proche, comme ensemble de modèles. Nous étudierons les problèmes pour chacune des classes de formules qui nous intéressent.

L'acquisition de connaissances à partir d'exemples est le cadre le plus courant des problématiques d'*apprentissage* : c'est celui de la forme la plus «pure» du *PAC-apprentissage* de Valiant [Val84], celui de la *Programmation Logique Inductive* [DL01], celui encore des problèmes plus généraux de *classification* (on parle également d'*apprentissage supervisé*) ou de *clustering* (apprentissage *non supervisé*) [HTF01]. La présentation de connaissances sous forme d'exemples survient en effet toujours naturellement. Dans le cadre de la classification par exemple, où il s'agit de calculer un *classifieur* (une fonction) attribuant une certaine classe à des affectations, on suppose qu'un expert du domaine a *étiqueté* un certain nombre d'objets, en leur attribuant leur vraie classe. Mais on peut aussi voir les exemples comme des situations envisageables *observées* dans un petit monde, par le biais de capteurs par exemple. On peut encore les voir comme les enregistrements d'une *base de données* (booléenne) dont les attributs sont les variables décrivant le petit monde de référence.

Si l'on considère à nouveau l'exemple du carrefour (préliminaires généraux, paragraphe 3.2), l'acquisition de connaissances est la problématique qui s'y dessine naturellement si l'on considère un extra-terrestre arrivant sur Terre, et voulant apprendre le fonctionnement de ce petit monde en l'observant. Au travers de capteurs par exemple, lui indiquant les couleurs des feux, la présence de véhicules à leur abord et décelant les coups de sifflet de gendarmes à l'intention de piétons, il peut ainsi découvrir un certain nombre de situations envisageables. Les processus que nous étudions dans cette partie prennent alors place lorsqu'il veut convertir cette liste de situations envisageables en des connaissances à propos de ce petit monde, dans le formalisme qu'accepte son système de gestion de connaissances. Rappelons (voir la discussion du paragraphe 3.1 des préliminaires généraux) que nous n'étudions pas la pertinence des variables pour décrire l'ensemble de situations en question, ni les processus de sélection des observations pertinentes. Nous supposons simplement donné un ensemble d'observations sur un certain ensemble de variables.

Pour ce qui est du petit monde de la thèse (paragraphe 3.3 des préliminaires généraux), les situations envisageables peuvent par exemple être découvertes en recoupant les souvenirs de vieux chercheurs et les informations d'une base de données sommaire des thèses qu'ils ont rapportées, et les connaissances acquises à partir de ces exemples pourraient ainsi servir à un jeune doctorant...

6.2 Problématiques étudiées

Comme nous l'avons dit, le problème général dont nous étudions l'algorithmique est le suivant :

> Etant donné un ensemble d'affectations M et une classe de formules $\mathcal{C}$, calculer une formule ϕ de la classe $\mathcal{C}$ représentant M.

Bien entendu, le terme «représenter» sera entendu différemment pour chaque problématique, et l'ensemble d'affectations M sera également diversement accessible. Notons que pour les problèmes algorithmiques de ce type, la classe $\mathcal{C}$ paramètre la *sortie* des algorithmes ; en conséquence, ces problèmes ne seront en général pas plus faciles pour une classe $\mathcal{C}$ que pour une *sur-classe* $\mathcal{C}'$ de $\mathcal{C}$.

Description et identification Les premières problématiques précises auxquelles nous nous intéressons (chapitre 7) sont celle de la *description* dans une classe $\mathcal{C}$ et celle de l'*identification* d'une telle classe [DP92]. Dans les deux cas, il s'agit de représenter *exactement* l'ensemble d'exemples donné par une formule de $\mathcal{C}$: en supposant que c'est possible pour la problématique de la description, et en ajoutant au problème la vérification que c'est possible pour l'identification. Ces deux problèmes représentent donc une acquisition de connaissances sans généralisation (*induction*). Il s'agit plutôt de *reformuler* des connaissances dans la forme qui nous intéresse, ou encore d'essayer de détecter sous quelle forme un ensemble d'exemples peut être représenté.

Ces problématiques correspondent par exemple au cas où l'extra-terrestre du paragraphe 6.1 observe le petit monde du carrefour pendant longtemps, et conclut qu'il y a recueilli toutes les situations envisageables. Il a donc la connaissance *exacte* de son fonctionnement, et n'a plus qu'à reformuler cette connaissance dans le langage que comprend son système de gestion de connaissances, afin de pouvoir la réutiliser par la suite, d'une part, et de pouvoir l'intégrer à ce système, où préexistent éventuellement d'autres connaissances, d'autre part.

Approximation Nous étudions ensuite le problème consistant à *approximer* un ensemble d'exemples par une formule d'une classe $\mathcal{C}$ donnée (chapitre 8), avec des notions d'approxi-

mation que nous préciserons [SK96]. Il s'agit donc d'intégrer des connaissances dans un langage précis, par exemple parce que c'est le seul que comprend notre système de gestion de connaissances, ou encore afin de pouvoir raisonner efficacement avec ces connaissances par la suite ; tout cela, au besoin en s'autorisant une perte d'information, l'ensemble initial d'exemples n'étant plus nécessairement l'ensemble des modèles de la formule calculée.

Cette situation correspond par exemple au cas où notre extra-terrestre ne peut retenir que des connaissances représentées par des formules affines. Après avoir détecté (problématique de l'*identification*) que ses observations ne peuvent pas être représentées par une telle formule (en l'occurrence, car le nombre de ces observations n'est pas une puissance de 2), l'extra-terrestre peut alors procéder à l'*approximation* de cet ensemble d'observations, afin d'intégrer à sa base de connaissances une formule du langage adéquat aussi proche que possible des observations.

PAC-apprentissage Enfin, nous étudierons le problème du *PAC-apprentissage* (chapitre 9), proche de celui de l'approximation mais avec un accès limité à l'ensemble d'exemples [Val84], le but étant d'être capable d'apprendre une bonne approximation d'un ensemble d'exemples en prenant connaissance du moins possible d'entre eux, puis de mettre à jour cette connaissance au fur et à mesure que sont disponibles de nouveaux exemples. Cette problématique est la plus proche de la notion d'apprentissage (ou *induction*) la plus couramment considérée et la plus naturelle. C'est en fait une formalisation pour la logique propositionnelle du problème de la *classification* lorsque le nombre de classes est limité à deux : la classe des *exemples* et celle des *contre-exemples*. Cette problématique fait également entrer en scène des notions de *probabilité*, qui formalisent les chances que l'on a d'observer tel ou tel exemple.

Cette problématique permet de formaliser, par exemple, le cas où notre extra-terrestre ne peut observer le carrefour que de façon sporadique, puisqu'il a mille autres choses à apprendre sur la Terre et que chaque venue à ce carrefour gaspille une partie précieuse de son temps. Il peut se passer longtemps entre deux observations, et dès les premières il veut acquérir un maximum de connaissances, qu'il mettra à jour lorsqu'il reviendra, pour tendre finalement vers une connaissance exacte.

Nous étudions donc dans cette partie l'algorithmique de ces quatre processus : description, identification, approximation et PAC-apprentissage, lorsque le langage de représentation de connaissances est l'un de ceux présentés dans la partie I. Dans tous les cas, nous nous intéressons également à la *taille* des formules calculées, cherchant à la minimiser ; en effet, d'une part cette taille correspond à l'espace requis pour stocker la nouvelle connaissance, et d'autre part, plus elle est petite, plus le raisonnement sera rapide avec de telles formules (car elles constituent les *données* des algorithmes de raisonnement). Rappelons également du chapitre 4, paragraphe 4.3 que nous n'envisageons pas les classes de formules FND comme des langages de représentation de connaissances à proprement parler ; nous ne les étudions donc que ponctuellement dans cette partie. Remarquons enfin que la difficulté des problèmes que nous étudions va crescendo avec l'ordre ci-dessus, car l'ensemble d'exemples ou d'observations donné représente la connaissance moins exactement pour l'approximation que pour la description ou l'identification, et parce qu'il la représente tout aussi exactement, mais est moins accessible pour le PAC-apprentissage que pour l'approximation.

A quelques exceptions près, toutes signalées, les résultats donnés dans le corps des chapitres sont des contributions originales de cette thèse. Nous rappelons de plus, dans le premier pa-

ragraphe de chaque chapitre, les travaux précédents sur le même sujet ; toutefois, nous nous limitons principalement aux résultats concernant les classes de formules présentées dans la première partie, car ces problèmes ont en général été étudiés pour un trop large spectre de langages pour que tous les résultats puissent être évoqués.

Chapitre 7

Description et identification

Sommaire

7.1 Présentation et travaux précédents 77
7.1.1 Formalisation des problématiques 77
7.1.2 Motivations 79
7.1.3 Travaux précédents 79
7.2 Description en FNC 81
7.3 Identification 86
7.3.1 Description en FNC première 86
7.3.2 Identification avec les impliqués premiers 89
7.4 Description et identification optimisées 91
7.4.1 Description en FNC négative ou monotone 92
7.4.2 Description en FNC de Horn 93
7.5 Autres classes de formules et résumé des résultats connus . . . 98

Nous étudions tout d'abord les problématiques de *description* et d'*identification* [DP92], c'est-à-dire à la conversion d'un ensemble d'observations en une formule sans perte d'information. Nous donnons tout d'abord un algorithme polynomial pour la description d'un ensemble d'affectations donné par une formule en FNC, puis un algorithme nouveau, général et efficace, pour la description d'un tel ensemble par une formule en FNC *première* et pour l'identification de toutes les classes de formules qui nous intéressent. Nous donnons enfin deux algorithmes, nouveaux et très efficaces, pour la description d'un ensemble d'affectations par une formule FNC monotone ou de Horn, lorsque c'est possible, et pour l'identification de ces classes. Enfin, nous concluons par la présentation rapide d'algorithmes similaires pour la description par des formules en FND.

7.1 Présentation et travaux précédents

Nous présentons dans ce paragraphe les formalisations que nous adoptons pour les problématiques de *description* et d'*identification*, puis nos motivations pour les étudier, et enfin un bref état de l'art sur le sujet.

7.1.1 Formalisation des problématiques

Description Nous commençons par présenter formellement la problématique de la *description*. Ce problème est paramétré par la classe de formules dans laquelle on veut décrire, et de fait chaque

telle classe définit un problème différent. Soit donc $\mathcal{C}$ une classe de formules propositionnelles ; le problème de la *description dans* $\mathcal{C}$ est le problème avec sortie suivant.

Problème 7.1 (`Description[`$\mathcal{C}$`]`)

Donnée : *Un ensemble d'affectations M, descriptible dans $\mathcal{C}$, à un ensemble de variables V*
Sortie : *Une formule $\phi \in \mathcal{C}$ décrivant M.*

Ainsi, la formule ϕ_C de l'exemple du carrefour est une description de l'ensemble d'affectations M_C dans la classe HORN, mais pas dans la classe BIJ, même s'il existe une formule de cette dernière classe logiquement équivalente à ϕ_C (voir le chapitre 2, paragraphe 2.4). La formule ϕ_C est donc une réponse correcte au problème `Description[HORN]` pour la donnée M_C, mais pas au problème `Description[BIJ]` pour la même donnée. Enfin, M_C n'est pas une donnée acceptable pour le problème `Description[AFFINE]`, puisqu'il n'existe pas de description affine de cet ensemble (son cardinal n'étant pas une puissance de 2).

De façon générale, la donnée du problème étant un ensemble d'affectations M (et implicitement l'ensemble de variables V), la complexité d'un algorithme pour ce problème sera mesurée en fonction du nombre $|M|$ d'affectations et du nombre $|V|$ de variables. Pour toutes les classes de formules $\mathcal{C}$ qui nous intéressent et pour tout ensemble d'affectations M, lorsqu'il existe au moins une description de M dans C il en existe toujours une de taille polynomiale en $|M|$ et $|V|$, comme nous le verrons plus loin, et de fait nous n'avons pas besoin des notions de complexité qui prennent en compte la taille de la sortie de l'algorithme évoquées dans les préliminaires généraux (paragraphe 2.3). Remarquons cependant que ce pourrait être le cas pour d'autres classes de formules, comme par exemple la classe des formules ϕ dont chaque clause contient exactement $|Var(\phi)|$ littéraux, comme nous le verrons plus loin.

Identification Pour une classe de formules $\mathcal{C}$, les données du problème `Description[`$\mathcal{C}$`]` sont supposées descriptibles dans $\mathcal{C}$; en revanche, celles du problème `Identification[`C`]` peuvent être quelconques. Ce problème ajoute donc à celui de la description la *reconnaissance* des ensembles d'affectations M descriptibles dans $\mathcal{C}$. Formellement, ce problème peut être énoncé comme suit ; soit $\mathcal{C}$ une classe de formules propositionnelles (comme pour la description, chaque classe $\mathcal{C}$ définit un problème différent).

Problème 7.2 (`Identification[`$\mathcal{C}$`]`)

Donnée : *Un ensemble d'affectations M à un ensemble de variables V*
Sortie : *«Non» si M n'est pas descriptible dans $\mathcal{C}$, et sinon une description de M dans $\mathcal{C}$.*

Ce problème est donc plus difficile que celui de la description dans la même classe de formules. Mentionnons par exemple que si l'on sait qu'un ensemble d'affectations M à un ensemble de variables V est descriptible par une formule FNC contenant trois littéraux ou moins par clause, il est facile de calculer une telle formule en temps polynomial : il suffit de «projeter» M sur tous les triplets de variables de V et de calculer les clauses correspondantes [DP92] ; en revanche, le problème consistant à décider si un tel ensemble M peut être décrit par une telle formule est coNP-complet [KS98].

La formule ϕ_C de l'exemple du carrefour est donc une réponse correcte au problème `Identi-`

fication[HORN] pour la donnée M_C. Pour ce qui est du problème Identification[BIJ] sur la même donnée, la formule ϕ_C n'est pas une réponse correcte, mais la formule 2FNC logiquement équivalente du chapitre 2, paragraphe 2.4 en est une, et enfin, la seule réponse correcte au problème Identification[AFFINE] sur la donnée M_C est «Non».

7.1.2 Motivations

Comme nous l'avons vu dans le chapitre 6, le processus consistant à décrire un ensemble d'affectations par une formule d'une classe $\mathcal{C}$ formalise par exemple la conversion *exacte* d'un ensemble d'observations, ou d'exemples, en une connaissance exprimée dans un langage requis. Ce processus peut également être vu comme un processus de *compilation d'une base d'exemples*, où la compilation [CD97, Lib98, DM02] a pour but de réduire la taille de la représentation, une liste d'exemples étant souvent une représentation coûteuse en espace, tout en calculant une représentation exploitable efficacement, comme le sont, pour les principales tâches de raisonnement, les formules des classes qui nous intéressent. On peut aussi voir ce processus comme le calcul d'une représentation *en intension* d'une connaissance donnée en *extension* : par exemple, à des fins de communication avec un être humain, lorsqu'il s'agit de l'informer du fonctionnement d'un petit monde observé ou de lui définir un concept (voir la discussion sur la lisibilité des représentations dans le chapitre 1, paragraphe 1.1). Encore une fois, lorsque l'on parle de formules FNC ou affines définies par la nature de leurs clauses, ce processus permet également d'obtenir une représentation d'une nouvelle connaissance qui pourra être ajoutée par conjonction à une base préexistante.

Pour ce qui est de la problématique de l'identification, son intérêt est plus général que celui de la description. Les utilisations d'un algorithme efficace résolvant ce problème pour une classe de formules $\mathcal{C}$ sont en effet les mêmes que celle d'un algorithme pour la *description* dans $\mathcal{C}$, mais un tel algorithme permet de plus de vérifier que l'on peut effectivement décrire l'ensemble d'observations ou d'exemples considéré dans $\mathcal{C}$. L'algorithme que nous présentons dans le paragraphe 7.3 permet de plus de déterminer en résolvant le problème une seule fois toutes les classes de formules, parmi celles qui nous intéressent, dans lesquelles l'ensemble d'affectations donné peut être décrit.

Enfin, comme nous l'avons vu dans les préliminaires généraux (paragraphe 3.1), une base de connaissances définit des liens entre les variables du petit monde ou du concept considéré : par exemple, des implications, ou encore des ensembles interdits. On voit qu'une représentation d'une telle base de connaissances par l'ensemble des exemples ou des situations envisageables décrit ce type de liens de façon *implicite*, tandis qu'une représentation par une formule (FNC ou affine en particulier) en représente une partie de manière *explicite*. On peut même voir une telle formule comme une *couverture* (non nécessairement minimale) de tous les liens de ce type entre les variables considérées, en ce sens que tous peuvent être engendrés (par résolution de clauses, ou par addition membre à membre d'équations) à partir de cette couverture. Alors, même s'il n'y a pas à proprement parler de découverte de nouvelles connaissances, on peut tout de même voir la description comme un processus permettant d'exhiber des liens entre variables.

7.1.3 Travaux précédents

Origine des problématiques Le problème de la *description* est un cas particulier du problème de la *conversion* d'une représentation des fonctions booléennes en une autre, et est à ce titre un problème classique. Néanmoins, le problème de l'identification, et implicitement celui de la description, ont été formalisés comme des problèmes algorithmiques d'Intelligence Artificielle par Dechter et Pearl [DP92]. Les auteurs voient ce processus comme un processus de *découverte*

d'une structure («structure identification») dans une base de données relationnelles donnée, ou encore comme un processus de *compilation de connaissances*. Notons que leur formalisation est plus générale que la nôtre, pour deux raisons : d'une part, elle prend en compte une classe de formules $\mathcal{C}'$ dans laquelle l'ensemble d'affectations donné est supposé descriptible, définissant ainsi un problème différent pour chaque paire de classes $(\mathcal{C}, \mathcal{C}')$; dans notre formalisation, la classe $\mathcal{C}'$ est toujours la classe de toutes les formules FNC, dans laquelle tout ensemble d'affectations est descriptible ; d'autre part, leur formalisation se place dans le cadre où le domaine des variables n'est pas nécessairement $\{0,1\}$.

Classes de formules Pour ce qui est des résultats correspondant à notre formalisation du problème, Dechter et Pearl montrent tout d'abord que les problèmes `Description[HORN]` et `Identification[HORN]` peuvent être résolus en temps $O(|M|^2|V|^2)$ pour une donnée (M,V), en calculant le cas échéant une formule contenant $O(|M||V|^2)$ clauses. Leur algorithme est basé sur la notion d'*enveloppe*, et mentionné également par Kavvadias, Papadimitriou et Sideri [KPS93]. Dechter et Pearl montrent également, de façon générale, que pour une constante $k \in \mathbb{N}$ le problème de la description dans la classe des formules kFNC (c'est-à-dire les formules FNC dont chaque clause contient au plus k littéraux) peut être résolu en temps $O(|M||V|^k)$, en calculant une formule contenant $O(|M||V|^k)$ clauses ; en revanche, le problème d'identification correspondant n'y est montré polynomial que dans le cas $k \leq 2$ (avec la même complexité), et le problème de décision associé y est conjecturé NP-difficile pour $k \geq 3$; cette conjecture est prouvée dans [KS98]. Cependant, la classe des formules kHorn, c'est-à-dire les formules FNC de Horn contenant au plus k littéraux par clauses, admet un algorithme de description de complexité $O(|M||V|^{k+1})$, calculant une formule contenant $O(|M||V|^{k+1})$ clauses, et peut être identifiée avec la même complexité. Enfin, Dechter et Pearl montrent que le problème consistant à décrire dans les classes des formules FNC ayant une structure d'arbre (voir aussi l'article plus ancien de Dechter [Dec90]) ou d'étoile, ou à identifier ces classes, est polynomial (nous renvoyons aux articles pour les définitions, mais notons que ces classes sont définies par la structure de leurs formules).

Un résultat précède néanmoins cet article. Angluin *et al.* [AFP92] montrent dans un formalisme différent (voir le chapitre 9) que le problème `Description[HORN]` peut être résolu en temps $O(\mu^2|M||V|^2)$, où μ est le nombre minimal de clauses dans une description de M, la formule calculée contenant au plus $\mu(|V|+1)$ clauses ; puisque nous montrons (paragraphe 7.3) que μ vaut toujours au plus $|M||V|$, la complexité de leur algorithme est également en $O(|M|^3|V|^2)$; notons que le fait que cet algorithme, bien qu'exhibé dans un formalisme différent, résout bien le problème `Identification[HORN]` est remarqué par Dechter et Pearl dans leur article.

Les problèmes `Description[AFFINE]` et `Identification[AFFINE]`, quant à eux, admettent des algorithmes polynomiaux, qui à notre connaissance ne sont pas mentionnés tels quels dans la littérature, mais découlent de résultats extrêmement classiques d'algèbre linéaire. Ainsi, la description polynomiale dans cette classe est assurée par le fait que l'on peut décrire un espace vectoriel par une formule affine étant donnée simplement une de ses bases (cf. chapitre 3, proposition 3.5), et qu'une base d'un ensemble d'affectations donné peut être calculée efficacement avec un algorithme glouton (de la même manière que pour l'approximation, voir la proposition 8.5 du chapitre 8).

Résultats généraux Si les résultats cités concernant la description exploitent la plupart du temps des propriétés particulières de chaque classe de formules, les résultats concernant l'identification utilisent tous un algorithme de description polynomiale, d'une part, et l'une ou l'autre de deux méthodes pour décider la descriptibilité de l'ensemble d'affectations M donné, d'autre

part : la première méthode consiste à décider si M est clos par l'opération correspondant à la classe, lorsqu'elle existe (voir résumé de ces opérations dans le chapitre 5), toutes étant calculables en temps polynomial de façon évidente ; la deuxième méthode, quant à elle, consiste à calculer une formule de la classe susceptible de décrire M, puis de tester qu'elle le décrit bien, en générant ses modèles. Cette deuxième méthode est notamment celle utilisée par Dechter et Pearl pour la classe BIJ [DP92].

Rappelons également que Kavvadias et Sideri [KS98] montrent un résultat de dichotomie pour le problème de l'identification dans le formalisme des classes de S-formules (problème `Inv-SAT[`S`]`), résultat que nous redonnons dans le chapitre 5. Enfin, à notre connaissance ni la description ni l'identification n'ont été étudiées dans la littérature pour les classes de formules renommables ou pour les classes de formules en FND.

Notre apport Notre contribution à cet état de l'art est la suivante. Nous unifions les résultats concernant la description et l'identification pour toutes les classes de formules FNC qui nous intéressent (que ce soit celles définies en nature ou les classes de formules renommables) et pour la classe des formules AFFINE, via un algorithme efficace basé sur la notion d'impliqué premier [ZH02]. Nous donnons ensuite des algorithmes plus efficaces que ceux connus jusqu'alors pour la description dans les classes HORN, NÉG et MONOT, et pour l'identification de ces mêmes classes [HZ03]. Enfin, nous montrons que tous ces algorithmes peuvent être adaptés, avec la même complexité, pour les classes de formules FND qui nous intéressent, sauf pour les classes HDP-REN-FND et HORN-FND.

7.2 Description en FNC

Nous présentons tout d'abord un algorithme polynomial pour décrire un ensemble d'affectations M donné en une formule FNC ; cet algorithme servira de base aux autres algorithmes présentés dans ce chapitre. Notons que comme les formules FNC peuvent décrire tous les ensembles d'affectations à un même ensemble de variables, nous n'avons pas d'hypothèse particulière sur la donnée M. Néanmoins, puisque la formule FNC réduite à la clause vide, $(())$, est insatisfaisable, nous pouvons écarter le cas $M = \emptyset$ comme trivial. Nous supposons donc, pour tout le chapitre, que les ensembles d'affectations sont non vides.

Représentation par affectations partielles Si M est un ensemble d'affectations à un ensemble de variables V, on peut calculer une formule FNC décrivant M en introduisant une clause pour chaque affectation m du complémentaire de M dans $\{0,1\}^V$, de sorte que m soit la seule affectation à V ne satisfaisant pas cette clause : il suffit de choisir la clause $\{x \mid x \in V \text{ et } m[x] = 0\} \cup \{\neg x \mid x \in V \text{ et } m[x] = 1\}$ (Dechter et Pearl [DP92] appellent cette représentation la «représentation canonique» de M). Par exemple, pour décrire l'ensemble d'affectations

$$M_C = \{0000, 0010, 0101, 1000, 1010, 1101\}$$

à l'ensemble de variables $V_C = \{x_{PV}, x_{VP}, x_{VV}, x_{DT}\}$ en FNC, on peut calculer la formule contenant les dix clauses suivantes, correspondant aux dix affectations $0001, 0011, \ldots, 1111$ de $\{0,1\}^{V_C} \setminus M_C$:

$$(x_{PV} \vee x_{VP} \vee x_{VV} \vee \neg x_{DT}), (x_{PV} \vee x_{VP} \vee \neg x_{VV} \vee \neg x_{DT}), \ldots, (\neg x_{PV} \vee \neg x_{VP} \vee \neg x_{VV} \vee \neg x_{DT})$$

Mais cet algorithme introduit un nombre exponentiel de clauses dans le pire des cas, le nombre total d'affectations à V étant $|\{0,1\}^V| = 2^{|V|}$. Pour éviter cette explosion, nous utilisons des

affectations partielles à V pour représenter de façon compacte les éléments du complémentaire de M dans $\{0,1\}^V$. Rappelons qu'une affectation partielle à V est une fonction de V dans $\{0,1,?\}$ (voir les préliminaires généraux). Celles que nous utilisons correspondent intuitivement à des clauses, et ce via la définition suivante.

Définition 7.1 (clause/FNC interdisant p, P) *Soient V un ensemble de variables, et p une affectation partielle à V. On appelle* clause interdisant *p, et on note $cl(p)$, la clause*

$$\{x \mid x \in V \text{ et } p[x] = 0\} \cup \{\neg x \mid x \in V \text{ et } p[x] = 1\}$$

Si P est un ensemble *d'affectations partielles à V, on appelle* formule FNC interdisant *P, et on note $fnc(P)$, la formule FNC $\bigcup_{p \in P} cl(p)$.*

Considérons à nouveau l'ensemble de variables $V_C = \{x_{PV}, x_{VP}, x_{VV}, x_{DT}\}$ de l'exemple du carrefour. L'ensemble $P_C = \{?0?1, ?1?0, 1?11, ?11?\}$ est un ensemble d'affectations partielles à V_C, son extension $ext(P_C)$ est l'ensemble d'affectations à V_C :

$$\{0,1\}^{V_C} \setminus M_C = \{0001, 0011, 0100, 0110, 0111, 1001, 1011, 1100, 1110, 1111\}$$

et la formule FNC interdisant P_C est la formule :

$$\phi_C = (x_{VP} \vee \neg x_{DT}) \wedge (\neg x_{VP} \vee x_{DT}) \wedge (\neg x_{PV} \vee \neg x_{VV} \vee \neg x_{DT}) \wedge (\neg x_{VP} \vee \neg x_{VV})$$

Pour p une affectation partielle à V, on voit facilement que les affectations à V satisfaisant la clause $cl(p)$ sont exactement les affectations de $\{0,1\}^V \setminus ext(p)$; similairement, pour un ensemble P de telles affectations, on voit que les affectations à V satisfaisant la formule FNC $fnc(P)$ sont exactement les affectations de $\{0,1\}^V \setminus ext(P)$. On a donc le lemme suivant.

Lemme 7.1 *Soient V un ensemble de variables et M (resp. P) un ensemble d'affectations (resp. d'affectations partielles) à V. Si l'extension de P est le complémentaire de M dans $\{0,1\}^V$, alors la formule FNC interdisant P, $fnc(P)$, décrit M.*

La traduction d'un ensemble d'affectations partielles P en la formule FNC l'interdisant, $fnc(P)$, nécessite simplement une lecture de P ; étant donné un ensemble d'affectations M à décrire en FNC, nous cherchons donc à définir un ensemble $P(M)$ d'affectations partielles à V dont l'extension soit le complémentaire de M dans $\{0,1\}^V$, et pouvant être construit en temps polynomial en $|M|$ et $|V|$. Rappelons que nous supposons $M \neq \emptyset$.

Construction La construction que nous proposons peut être vue comme le calcul du «complémentaire» d'un arbre de décision représentant M (voir préliminaires généraux, paragraphe 2.4). Nous donnons une intuition de cette construction sur la figure 7.1, dont la partie gauche représente l'arbre de décision associé à l'ensemble d'affectations $M_C = \{0000, 0010, 0101, 1000, 1010, 1101\}$ de l'exemple du carrefour.

On voit que l'ensemble des branches non présentes dans un tel arbre représente le complémentaire de M dans $\{0,1\}^V$. Pour représenter ces branches de façon compacte, nous distinguons les noeuds internes de l'arbre qui ne possèdent qu'un fils, et considérons les racines des fils manquants ; ces racines sont marquées par un cercle sur la figure 7.1 (arbre de droite). Il est alors aisé de voir que les chemins de la racine de l'arbre représentant M à ces racines codent le complémentaire de M dans $\{0,1\}^V$ sous forme d'affectations partielles à V, comme indiqué sur l'arbre de droite de la figure 7.1.

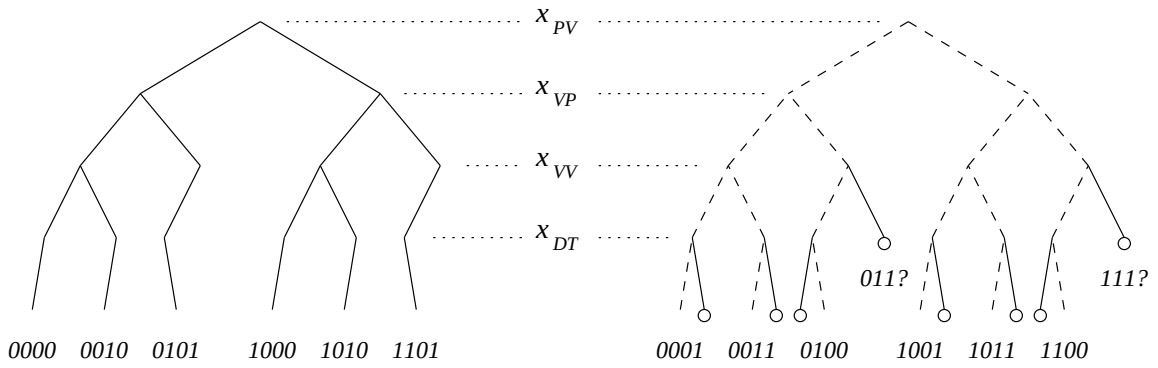

FIG. 7.1 – L'arbre de décision associé à M_C et son «complémentaire»

Formalisation C'est cette construction de l'ensemble d'affectations partielles $P(M)$ que nous formalisons maintenant, en fonction des éléments de M et de V ; nous prouvons ensuite que l'extension $ext(P(M))$ de l'ensemble ainsi construit est bien le complémentaire de M dans $\{0,1\}^V$, et concluons avec le lemme 7.1.

Rappelons que nous supposons toujours un ordre total $<$ sur les éléments de V ; nous notons donc ces éléments $x_1, x_2, \ldots, x_{|V|}$ de sorte que l'on ait $x_1 < x_2 < \ldots < x_{|V|}$, sans perte de généralité. Soit alors $(m_1, m_2, \ldots, m_{|M|})$ l'ensemble des éléments de M ordonnés par ordre lexicographique croissant, c'est-à-dire :

$$m' < m \iff \exists i \in \{1,2,\ldots,|V|\}, m[x_i] = 1, m'[x_i] = 0 \text{ et } \forall i' < i, m[x_{i'}] = m'[x_{i'}]$$

Intuitivement, l'ordre lexicographique croissant sur les affectations de M est donc l'ordre dans lequel on lit ces affectations, de gauche à droite, dans l'arbre de décision associé à M (selon le même ordre sur les variables). Ainsi, pour l'ensemble d'affectations M_C de l'exemple, l'ordre lexicographique est $(0000, 0010, 0101, 1000, 1010, 1101)$. Pour un élément m_k de M ($k \geq 2$), soit $prec(m_k)$ le (seul) entier de $\{1,2,\ldots,|V|\}$ tel que l'on ait :

$$\begin{cases} \forall i \in \{1,2,\ldots,|V|\}, i < prec(m_k) \Rightarrow m_{k-1}[x_i] = m_k[x_i] \\ m_{k-1}[x_{prec(m_k)}] = 0 \text{ et } m_k[x_{prec(m_k)}] = 1 \end{cases}$$

On convient de plus que $prec(m_1)$ vaut 0 (dans tous les cas, nous omettons la référence à M pour alléger les notations) ; intuitivement, $prec(m_k)$ est donc la profondeur à laquelle la branche représentant m_k dans l'arbre associé à M se «sépare» de la branche immédiatement à sa gauche (1 pour la racine). Ainsi, sur l'arbre de gauche de la figure 7.1 on voit que l'on a $prec(0010) = 3$, $prec(1000) = 1$ et $prec(0000) = 0$. Symétriquement, pour un élément m_k de M ($k \leq |M| - 1$), soit $succ(m_k)$ le (seul) entier de $\{1,2,\ldots,|V|\}$ tel que l'on ait :

$$\begin{cases} \forall i \in \{1,2,\ldots,|V|\}, i < succ(m_k) \Rightarrow m_k[x_i] = m_{k+1}[x_i] \\ m_k[x_{succ(m_k)}] = 0 \text{ et } m_{k+1}[x_{succ(m_k)}] = 1 \end{cases}$$

On convient $succ(m_{|M|}) = 0$. Ainsi, on voit sur la figure 7.1 que l'on a $succ(0101) = 1$, $succ(1010) = 2$ et $succ(1101) = 0$. Les valeurs $prec(m)$ et $succ(m)$ nous serviront à déterminer, intuitivement, la profondeur à partir de laquelle on peut trouver des noeuds internes avec un seul fils sur la branche représentant m_k dans l'arbre. On voit en effet aisément que si l'on a

Entrée : Un ensemble d'affectations $M \neq \emptyset$ à un ensemble de variables $V = \{x_1, x_2, \ldots, x_{|V|}\}$
Sortie : Une formule FNC ϕ décrivant M.

Début
  Trier M par ordre lexicographique ; /* $M = (m_1, \ldots, m_{|M|})$ */
  $P(M) \leftarrow \emptyset$;
  Pour $k = 1, \ldots, |M|$ **Faire**
    calculer $prec(m_k)$ et $succ(m_k)$;
    Pour $i = 1, \ldots, |V|$ **Faire**
      Si on a soit $m_k[x_i] = 1$ et $i > prec(m_k)$, soit $m_k[x_i] = 0$ et $i > succ(m_k)$ **Alors**
        $p(m_k, i)[x'_i] \leftarrow m_k[x_i]$ pour tout $i' < i$;
        $p(m_k, i)[x_i] \leftarrow m_k[x_i] \oplus 1$;
        $p(m_k, i)[x'_i] \leftarrow ?$ pour tout $i' > i$;
        $P(M) \leftarrow P(M) \cup \{p(m_k, i)\}$;
      Fin Si ;
    Fin Pour ;
  Fin Pour ;
  retourner $fnc(P(M))$;
Fin

FIG. 7.2 – Algorithme `DecrFNC`

$i > prec(m_k)$ et $m_k[i] = 1$ (resp. $i > succ(m_k)$ et $m_k[i] = 0$), alors le noeud représentant x_i sur la branche représentant m_k n'a qu'un fils.

Pour un tel m_k et un tel i, nous introduisons donc l'affectation partielle à V correspondant au fils manquant du noeud représentant x_i, notée $p(m_k, i)$ (encore une fois nous omettons la référence à M pour alléger les notations) ; cette affectation est définie par :

$$\begin{cases} p(m_k, i)[x_j] = m_k[x_j] & \text{pour } x_j \in V, j < i \\ p(m_k, i)[x_i] = m_k[x_i] \oplus 1 & \\ p(m_k, i)[x_j] = ? & \text{pour } x_j \in V, j > i \end{cases}$$

Considérons ainsi, par exemple, l'affectation $0101 \in M_C$; nous avons vu que $succ(0101)$ vaut 1, donc on a $3 > succ(0101)$, et comme $0101[3]$ vaut 0 on introduit l'affectation partielle à V_C $p(0101, 3) = 011?$.

Nous définissons enfin l'ensemble d'affectations partielles à V noté $P(M)$, et égal à l'ensemble :

$$\{p(m, i) \mid m \in M, 1 \leq i \leq |V| \text{ et } (i > prec(m) \text{ et } m[x_i] = 1) \text{ ou } (i > succ(m) \text{ et } m[x_i] = 0)\}$$

dont l'extension est égale au complémentaire de M dans $\{0,1\}^V$. Ainsi, pour l'ensemble d'affectations M_C on a :

$$P(M_C) = \{0001, 0011, 0100, 011?, 1001, 1011, 1100, 111?\}$$

comme reporté sur la figure 7.1.

Nous sommes finalement en mesure de prouver que l'ensemble $P(M)$ ainsi défini convient bien, puis que l'on peut décrire un ensemble affectations M en une formule FNC en temps polynomial ; l'algorithme est résumé sur la figure 7.2.

Lemme 7.2 *Soient V un ensemble de variables et $M \neq \emptyset$ un ensemble d'affectations à V. L'ensemble $P(M)$ d'affectations partielles à V vérifie $ext(P(M)) = \{0,1\}^V \backslash M$.*

Preuve Nous montrons tout d'abord que pour tout $m \in M$, on a $m \notin ext(P(M))$. Pour tout $i \in \{1, 2, \ldots, |V|\}$ tel que $p(m,i)$ est défini on a $m \notin ext(p(m,i))$, puisque par construction on a $p(m,i)[x_i] \neq m[x_i]$. Soit maintenant $m' \in M$ avec $m' \neq m$, et soit $p(m',i)$, pour un certain $i \in \{1, 2, \ldots, |V|\}$, un élément de $P(M)$. Par symétrie, supposons que m est plus petit que m' dans l'ordre lexicographique, et soit i_0 le plus petit indice avec $m[x_{i_0}] = 0$ et $m'[x_{i_0}] = 1$. Si on a $i > i_0$ alors on a $p(m',i)[x_{i_0}] = m'[x_{i_0}] = 1$, donc $m \notin ext(p(m',i))$. On ne peut avoir $i = i_0$ car, du fait de la présence de m dans M, on a $prec(m') \geq i_0$. Reste donc le cas $i < i_0$. Comme nous l'avons remarqué on a $prec(m') \geq i_0$, donc $p(m',i)$ est dans le cas $i > succ(m')$, et donc $m'[x_i] = 0$ et $p(m',i)[x_i] = 1$. Or, par minimalité de i_0 on a $m[x_i] = m'[x_i] = 0$, et finalement on a $m \notin ext(p(m',i))$.

Nous montrons maintenant que pour tout $m \in \{0,1\}^V \backslash M$, on a $m \in ext(P(M))$. Nous notons encore $(m_1, m_2, \ldots, m_{|M|})$ les éléments de M triés par ordre lexicographique croissant. Si m est plus petit que m_1 pour cet ordre, soit i_0 le plus petit indice avec $m[x_{i_0}] = 0$ et $m_1[x_{i_0}] = 1$; alors par construction on a $m \in ext(p(m_1, i_0))$. Le cas où m est plus grand que $m_{|M|}$ pour l'ordre lexicographique est symétrique. Supposons enfin que m est plus grand que m_j et plus petit que m_{j+1} pour un certain $j \in \{1, 2, \ldots, |M| - 1\}$, et notons $i(j), i(j+1)$ les plus petits indices tels que l'on ait $m_j[x_{i(j)}] = 0$, $m[x_{i(j)}] = 1$, $m[x_{i(j+1)}] = 0$ et $m_{j+1}[x_{i(j+1)}] = 1$. Par définition de $prec(m_{j+1})$ et de $succ(m_j)$ on a $succ(m_j) = prec(m_{j+1})$, et par définition de $i(j)$ et de $i(j+1)$ on a soit $i(j) > succ(m_j)$ (et $i(j+1) = prec(m_{j+1})$), soit $i(j+1) > prec(m_{j+1})$ (et $i(j) = succ(m_j)$). Dans le premier cas, on a $m \in ext(p(m_j, i(j)))$, et dans le second, $m \in ext(p(m_{j+1}, i(j+1)))$. Finalement, on a bien $m \in ext(P(M))$ dans tous les cas. □

Proposition 7.1 (`Description[FNC]`) *Soit V un ensemble de variables, et soit $M \neq \emptyset$ un ensemble d'affectations à V. L'algorithme* `DecrFNC` *de la figure 7.2 calcule une description de M en FNC en temps $O(|M||V|^2)$, et cette description contient au plus $|M||V|$ clauses.*

Preuve Clairement, l'algorithme calcule la formule $fnc(P(M))$. L'ensemble M peut être trié par ordre lexicographique en temps $O(|M||V|)$ en utilisant un arbre de décision ; en effet, cet arbre peut être construit en temps $O(|M||V|)$ (préliminaires généraux, paragraphe 2.4), et l'ensemble des branches peut être lu par un parcours en profondeur, en temps linéaire en la taille de l'arbre. La boucle principale de l'algorithme est ensuite exécutée $|M|$ fois. A chaque exécution, les valeurs $prec(m_k)$ et $succ(m_k)$ peuvent être calculées en temps $O(|V|)$ en lisant m_{k-1} et m_{k+1}. Puis la boucle interne est exécutée $|V|$ fois, avec à chaque fois au plus l'écriture d'une clause, en temps $O(|V|)$. On obtient donc une complexité en $O(|M||V| + |M|(|V| + |V||V|)) = O(|M||V|^2)$. Enfin, le nombre de clauses est borné par le nombre de couples (m_k, i) avec $m_k \in M$ et $i \in \{1, 2, \ldots, |V|\}$, soit au plus $|M||V|$ clauses. □

Remarque 7.1 (généralisation) *Ce calcul est également possible, avec la même complexité, si la donnée n'est pas une relation, mais une formule FND* associable à un arbre binaire *; nous entendons par là une formule FND ϕ sur les variables de laquelle on puisse trouver un ordre total, de sorte que chaque terme de ϕ contenant un littéral formé sur une variable x contienne également un littéral formé sur chacune des variables plus petites que x. Nous ne détaillons pas plus ce point ici, mais la construction à partir d'un arbre binaire donne l'intuition, puisqu'on peut représenter une telle formule FND par un arbre de la même manière qu'une relation. Notons*

qu'on peut reconnaître efficacement ces formules, et calculer un ordre adéquat, en procédant de façon gloutonne.

7.3 Identification

Nous nous intéressons maintenant à la problématique de l'*identification*. Rappelons qu'étant donnés une classe $\mathcal{C}$ de formules propositionnelles et un ensemble d'affectations M, il s'agit de décider si M a au moins une description dans $\mathcal{C}$ et, le cas échéant, d'en calculer une. Nous donnons ici un nouvel algorithme, basé sur l'algorithme `DecrFNC`, qui résout ce problème efficacement et de façon unifiée pour presque toutes les classes de formules FNC qui nous intéressent ; seule la classe des formules Horn définies positives renommables échappe à la règle, mais nous verrons cependant que le problème peut être contourné.

7.3.1 Description en FNC première

Idée de l'algorithme Notre algorithme est basé sur la notion d'*impliqué premier* ; nous avons vu (voir le paragraphe 5.3 page 69) que pour toutes les classes $\mathcal{C}$ de formules FNC qui nous intéressent, une formule FNC première est dans $\mathcal{C}$ si et seulement si l'ensemble de ses modèles est descriptible dans $\mathcal{C}$. L'idée de l'algorithme est donc, étant donné M, de calculer une formule FNC première ϕ décrivant M, puis de décider l'appartenance de ϕ à la classe $\mathcal{C}$; si ϕ est une formule de $\mathcal{C}$, on peut retourner ϕ, et sinon on peut conclure que M n'a pas de description dans $\mathcal{C}$, et donc retourner «Non». C'est la NP-complétude du test de renommabilité d'une formule FNC en une formule de Horn définie positive qui explique que cet algorithme n'est pas applicable aussi simplement pour la classe HDP-Ren ; nous verrons également que les formules affines, du fait qu'elles ne sont pas des formules FNC, nécessitent un traitement supplémentaire, qui n'altère cependant pas fondamentalement le principe de l'algorithme.

Pour calculer une formule FNC première décrivant M, nous rendons première chaque clause d'une description de M en FNC, indépendamment ; nous avons vu (préliminaires généraux, paragraphe 1.4) que cette opération préservait l'équivalence logique. Etant donnée une clause C d'une description de M, nous supprimons de C un nombre maximum de littéraux tout en maintenant la propriété $\forall m \in M, m \models C$. On peut le faire de façon gloutonne, en parcourant une fois la clause et en décidant pour chaque littéral $\ell \in C$ et pour chaque affectation $m \in M$ si m empêche de supprimer ℓ de C, c'est-à-dire si $m \not\models C\backslash\{\ell\}$; mais cela conduit à une complexité en $O(|M||V|)$ pour chaque littéral de C, et donc en $O(|M||V|^2)$ pour chaque clause C. Nous montrons qu'on peut obtenir une complexité en $O(|M||V|)$, en ne parcourant M qu'une fois pour une clause C donnée.

Formalisation Pour atteindre cette complexité, pour une clause C fixée nous résumons M en un tableau T indicé par les littéraux de C et les affectations $m \in M$. Pour un littéral $\ell \in C$ et une affectation $m \in M$, nous mettons $T[\ell, m]$ à 1 si $m \models \ell$, et à 0 dans le cas contraire.

Par exemple, la figure 7.3 donne le tableau T pour la clause $(x_{PV} \vee \neg x_{VP} \vee \neg x_{VV})$ calculée par l'algorithme `DecrFNC` pour l'ensemble d'affectations M_C (clause provenant de l'affectation partielle 011?).

Nous devons donc garder un ensemble minimal (pour l'inclusion) de littéraux de C tel que pour chaque $m \in M$, on ait gardé un littéral ℓ avec $T[\ell, m] = 1$. Nous utilisons un ordre total sur les littéraux, par exemple celui des variables sur lesquelles ils sont formés, et pour chaque $m \in M$ nous distinguons le plus grand littéral $\ell \in C$ avec $T[\ell, m] = 1$: nous notons $T[\ell, m] = 2$.

	0000	0010	0101	1000	1010	1101
x_{PV}	0	0	0	1	1	1
$\neg x_{VP}$	1	**1**	0	1	**1**	0
$\neg x_{VV}$	**1**	0	**1**	**1**	0	**1**

FIG. 7.3 – Le tableau T pour la clause $(x_{PV} \vee \neg x_{VP} \vee \neg x_{VV})$ et l'ensemble d'affectations M_C

Par exemple, pour l'exemple précédent nous remplaçons par 2 les valeurs en gras de la figure 7.3. Le tableau T se remplit en une passe, et nous utilisons finalement un algorithme glouton pour minimiser C sans relire M. Le processus est résumé sur la figure 7.4 ; nous supposons que l'ordre choisi sur les littéraux est celui de leurs variables, avec $x_1 < x_2 < \cdots < x_{|V|}$, et que la formule FNC que l'on rend première est la formule calculée par l'algorithme `DecrFNC`. Ainsi, pour la clause $(x_{PV} \vee \neg x_{VP} \vee \neg x_{VV})$ de l'exemple précédent, l'algorithme retire de la clause le littéral x_{PV}, car le tableau de la figure 7.3 indique que pour aucune affectation $m \in M_C$ ce n'est le dernier littéral l'assurant d'être un modèle de la clause ; la liste `modeles` reste vide. Ensuite, comme on a $T[\neg x_{VP}, 0010] = 2$ (**1** sur la figure), l'algorithme garde ce littéral dans la clause, et ajoute les affectations 0000, 0010, 1000 et 1010 à la liste `modeles` ; en effet, en gardant le littéral $\neg x_{VP}$ on est assuré que ces affectations seront des modèles de la clause. Enfin, l'algorithme considère le littéral $\neg x_{VV}$, et puisque on a $T[\neg x_{VV}, 0101] = 2$ et que 0101 n'est pas encore assuré d'être un modèle de la clause (il n'est pas dans la liste `modeles`), on garde également ce littéral. Finalement, l'algorithme a minimisé la clause $(x_{PV} \vee \neg x_{VP} \vee \neg x_{VV})$ en la clause $(\neg x_{VP} \vee \neg x_{VV})$. Si on fait complètement tourner l'algorithme sur l'ensemble d'affectations M_C, on obtient la formule FNC première décrivant M_C :

$$(x_{VP} \vee \neg x_{DT}) \wedge (\neg x_{VV} \vee \neg x_{DT}) \wedge (\neg x_{VP} \vee x_{DT}) \wedge (\neg x_{VP} \vee \neg x_{VV})$$

La preuve de la correction et de la complexité de l'algorithme `DecrIP` est simple, et énoncée dans la proposition suivante.

Proposition 7.2 (Description en FNC première) *Soient V un ensemble de variables, et $M \neq \emptyset$ un ensemble d'affectations à V. L'algorithme* `DecrIP` *calcule une description de M en FNC et première en temps $O(|M|^2|V|^2)$, et cette description contient au plus $|M||V|$ clauses.*

Preuve Le temps d'exécution et le nombre de clauses se lisent directement sur l'algorithme. Le seul point difficile est la correction de la procédure de minimisation des clauses. Soit donc C une clause de la formule `DecrFNC`(M), et notons C' la clause obtenue de C après minimisation par l'algorithme. Puisque `DecrFNC`(M) décrit M, on a $\forall m \in M, m \models C$, et puisque la procédure de minimisation garde au moins un littéral ℓ de C avec $T[\ell, m] = 1$ ou 2 pour tout $m \in M$, par construction toutes les affectations $m \in M$ satisfont bien la clause C'. Reste à montrer que C' est première. Soit ℓ un littéral de C' ; puisque la procédure de minimisation n'a pas retiré ℓ de C, on sait que lorsqu'elle a considéré ℓ il existait $m \in M$ avec $T[\ell, m] = 2$ et $m \notin$ `modeles`. On en déduit que m ne satisfait pas la clause $C' \backslash \{\ell\}$, et comme il existe un tel $m \in M$ pour chaque littéral $\ell \in C$, que C' est première. □

Remarque 7.2 (suite de la remarque 7.1) *Le passage à une formule première ne fonctionne plus tel quel avec une formule FND ϕ associable à un arbre binaire comme donnée. En effet, dans ce cas on ne peut pas maintenir la liste* `modeles` *de l'algorithme à jour en temps polynomial. Il*

Entrée : Un ensemble d'affectations $M \neq \emptyset$ à un ensemble de variables $V = \{x_1, x_2, \ldots, x_{|V|}\}$
Sortie : Une formule FNC première ϕ décrivant M.

Début
  $\phi \leftarrow$ `DecrFNC`(M) ;
  Pour chaque clause $C \in \phi$ **Faire**
    / Création et remplissage du tableau T : */*
    $T \leftarrow$ un tableau d'entiers, indicé par les éléments de C et de M ;
    Pour chaque $m \in M$ **Faire**
      `der` $\leftarrow 1$;
      Pour chaque $\ell \in C$ dans l'ordre décroissant **Faire**
        Si $m \models \ell$ et `der`$= 1$ **Alors** [$T[\ell, m] \leftarrow 2$; `der`$\leftarrow 0$;]
        Sinon Si $m \models \ell$ **Alors** $T[\ell, m] \leftarrow 1$;
        Sinon $T[\ell, m] \leftarrow 0$;
        Fin Si ;
      Fin Pour ;
    Fin Pour ;
    / Minimisation de C : */*
    `modeles` $\leftarrow \emptyset$;
    Pour chaque $\ell \in C$ dans l'ordre croissant **Faire**
      Si $\exists m \in M \backslash$`modeles` avec $T[\ell, m] = 2$ **Alors**
        / on garde ℓ dans C */*
        `modeles` $\leftarrow$ `modeles` $\cup \{m' \in M \mid T[\ell, m'] = 1$ ou $2\}$;
      Sinon
        $C \leftarrow C \backslash \{\ell\}$;
      Fin Si ;
    Fin Pour ;
  Fin Pour ;
  retourner ϕ ;
Fin

FIG. 7.4 – Algorithme `DecrIP`

faut donc employer pour chaque clause C un algorithme glouton, qui décide pour chaque littéral ℓ si on a $\forall m \in M, m \models C\backslash\{\ell\}$; ce test peut être réalisé en temps linéaire en la taille $|M||V|$ de M, mais la complexité totale du processus est alors en $O(|M||V|^2)$ pour chaque clause, et donc en $O(|M|^2|V|^3)$ au total.

Une autre remarque importante concernant cet algorithme est qu'il permet de détecter d'éventuels ensembles de variables indépendants dans M, c'est-à-dire une partition $\{V_1, V_2, \ldots, V_k\}$ de V telle que M, vue comme une base de données relationnelle sur l'ensemble d'attributs V, puisse être projetée sur les V_i sans perte d'information. Par exemple, l'ensemble d'affectations M_C de l'exemple du carrefour peut être décomposée sans perte d'information sur les sous-ensembles $\{x_{PV}\}$ et $\{x_{VP}, x_{VV}, x_{DT}\}$ de V_C. Notons qu'une telle partition maximalement fine est unique. Pour la calculer, la description ϕ de M calculée par l'algorithme `DecrIP` étant première, il suffit de calculer les composantes connexes de la relation binaire $\mathcal{R}$ sur $V \times V$ définie par : $x\mathcal{R}x'$ si et seulement s'il existe une clause de ϕ contenant un littéral formé sur x et un littéral formé sur x'.

Enfin, le fait que la description calculée soit première garantit que n'y apparaissent que les variables de V dont M dépend réellement, c'est-à-dire l'unique sous-ensemble minimal de variables $V' \subseteq V$ tel que M soit l'ensemble des prolongements des éléments de sa projection sur V à V'. Par exemple, l'ensemble d'affectations M_C de l'exemple du carrefour ne dépend réellement que des variables x_{VP}, x_{VV} et x_{DT}, et la variable x_{PV} n'apparaît pas dans la formule calculée par l'algorithme `DecrIP`.

Pour une présentation détaillée des notions d'indépendance, nous renvoyons le lecteur à l'article de Lang *et al.* [LLM02] (la notion du paragraphe précédent y est appelée «V-indépendance sémantique»).

7.3.2 Identification avec les impliqués premiers

Nous pouvons désormais énoncer nos résultats pour le problème de l'identification d'une classe $\mathcal{C}$. Rappelons qu'étant donné un ensemble d'affectations M, l'idée est de calculer une description ϕ de M, en FNC et première, puis de tester l'appartenance de ϕ à $\mathcal{C}$.

Proposition 7.3 (`Identification[`$\mathcal{C}$`]`) *Les classes de formules :*

HORN,ANTI-HORN,HDP,ANTI-HDP,NÉG,POS,BIJ,HORN-REN,MONOT

sont identifiables dans un ensemble d'affectations M donné à un ensemble de variables V en temps $O(|M|^2|V|^2)$; pour toutes les classes, l'éventuelle formule calculée contient au plus $|M||V|$ clauses.

Preuve L'algorithme `DecrIP` calcule une formule FNC première ϕ, contenant au plus $|M||V|$ clauses et décrivant M, en temps $O(|M|^2|V|^2)$ (proposition 7.2). Pour chaque classe de formules $\mathcal{C}$ de l'énoncé on peut tester si ϕ est dans $\mathcal{C}$ en temps linéaire en sa taille, c'est-à-dire en temps $O(|M||V|^2)$ (cf. chapitre 5, paragraphe 5.3). Or, pour chacune de ces classes $\mathcal{C}$, ϕ est dans $\mathcal{C}$ si et seulement si M y est descriptible (paragraphe 5.3 du chapitre 5), ce qui conclut la preuve. □

Ainsi, sur l'exemple du carrefour la formule calculée dans le paragraphe 7.3.1 est une réponse correcte aux problèmes :

`Identification[HORN]`, `Identification[BIJ]`, `Identification[HORN-REN]`

pour la donnée M_C. En revanche, on voit que la réponse aux problèmes :

`Identification[HDP]`, `Identification[NÉG]`, `Identification[ANTI-HORN]` etc.

pour la même donnée est «Non».

La classe des formules Horn définies positives renommables ne peut pas être montrée identifiable de la même manière, puisque décider si une formule FNC donnée appartient à cette classe est un problème NP-complet. Nous avons cependant vu (proposition 4.7 du chapitre 2) que les renommages candidats formaient un sous-ensemble de l'ensemble des renommés des modèles de la formule par l'affectation de 1 à toutes les variables. L'ensemble des modèles de la formule faisant partie de la donnée du problème, nous pouvons donc tout de même utiliser le résultat pour la classe HDP pour montrer que la classe HDP-REN est également identifiable.

Proposition 7.4 (`Identification[HDP-REN]`) *La classe* HDP-REN *est identifiable dans un ensemble d'affectations M à un ensemble de variables V en temps $O(|M|^2|V|^2)$; l'éventuelle formule calculée contient au plus $|M||V|$ clauses.*

Preuve Etant donné M, nous calculons tout d'abord une description ϕ de M, en FNC et première, avec l'algorithme `DecrIP` ; ce calcul a une complexité en $O(|M|^2|V|^2)$, et ϕ contient au plus $O(|M||V|)$ clauses (proposition 7.2). D'après la proposition 4.2 du chapitre 4 et la proposition 2.8 du chapitre 2, on sait que M est descriptible dans HDP-REN si et seulement si l'on a $\phi \in$ HDP-REN. Pour décider cette appartenance, nous utilisons la proposition 4.7 du chapitre 4, et décidons donc $\phi_{\oplus\rho} \in$ HDP pour tout $\rho \in M_{\oplus\overline{1_V}}$; par la proposition, on a $\phi \in$ HDP-REN si et seulement si l'un de ces renommés est dans HDP. Nous devons donc calculer au plus $|M|$ renommés de ϕ, chacun en temps linéaire en la taille de ϕ, soit en temps $O(|M||V|^2)$, et décider pour chacun s'il appartient à HDP, encore une fois en temps $O(|M||V|^2)$, et nous avons le résultat. □

Notons que cette idée peut également être exploitée pour les classes HORN-REN et MONOT, mais cela donne la même complexité qu'en utilisant notre procédure. Notons également que pour les classes HORN-REN, HDP-REN et MONOT, les algorithmes que nous proposons permettent de réaliser *mieux* que la tâche d'identification elle-même. Elles permettent également de calculer un renommage adéquat, et même de décider si l'ensemble M peut être décrit par une formule, disons HORN-renommable, *grâce au même renommage* qu'une formule donnée, par exemple une autre base de connaissances. Il suffit en effet de tester si la conjonction de cette base de connaissances et de la formule FNC première calculée est bien HORN-renommable.

Nous terminons ce paragraphe avec le problème `Identification[AFFINE]`. Cette classe ne peut pas non plus être montrée identifiable dans le cadre de la proposition 7.3, puisque la notion de formule affine ne concerne pas les formules FNC ; nous avons cependant vu (chapitre 3) que les formules FNC *premières* représentant des théories affines avaient une propriété purement syntaxique particulière. C'est cette propriété que nous utilisons pour montrer le résultat suivant dans notre cadre.

Proposition 7.5 (`Identification[AFFINE]`) *La classe* AFFINE *est identifiable dans un ensemble d'affectations M à un ensemble de variables V en temps $O(|M|^2|V|^2)$.*

Preuve Nous utilisons le lien syntaxique entre les formules affines et les formules FNC premières logiquement équivalentes (proposition 3.7 du chapitre 3). Nous calculons tout d'abord une description ϕ de M en FNC et première avec l'algorithme `DecrIP` (en temps $O(|M|^2|V|^2)$, ϕ contenant au plus $|M||V|$ clauses). Nous calculons ensuite la formule $aff(\phi)$ (voir chapitre 3), en temps linéaire en la taille de ϕ, soit en temps $O(|M||V|^2)$. On sait (voir la remarque après la proposition 3.7 du chapitre 3) que $aff(\phi)$ est logiquement plus forte que ϕ, et donc que les affectations à V qui la satisfont sont dans M. Il ne reste qu'à vérifier qu'à l'inverse, toutes les affectations de M la satisfont, en temps linéaire en sa taille pour chacune, soit en temps

$O(|M|^2|V|^2)$ au total. Si c'est le cas, alors $aff(\phi)$ est bien une description affine de M, et dans le cas contraire la proposition 3.7 du chapitre 3 nous dit que M n'est pas descriptible dans AFFINE. □

Ainsi, si l'on considère l'exemple du carrefour, on calcule tout d'abord, à partir de la formule FNC première calculée dans le paragraphe 7.3.1, la formule affine :

$$(x_{VP} \oplus x_{DT} = 0) \wedge (x_{VV} \oplus x_{DT} = 1) \wedge (x_{VP} \oplus x_{VV} = 1)$$

Or, on voit que l'affectation $0000 \in M_C$ ne satisfait pas la deuxième équation de cette formule, et on en déduit que M_C n'est pas descriptible dans AFFINE (remarquons cependant que lorsque, comme dans ce cas, le cardinal de M n'est pas une puissance de 2, on peut obtenir cette conclusion sans calculer une description de M). En revanche, pour l'exemple de la thèse, sur la donnée M_t l'algorithme `DecrIP` calcule la formule FNC première :

$$\phi_t = \begin{array}{ll} & (x_{qu} \vee x_{ba}) \wedge (\neg x_{qu} \vee \neg x_{ba}) \\ \wedge & (x_{me} \vee \neg x_{ba} \vee x_{in}) \wedge (x_{me} \vee x_{ba} \vee \neg x_{in}) \wedge (\neg x_{me} \vee x_{ba} \vee x_{in}) \\ \wedge & (\neg x_{me} \vee \neg x_{ba} \vee \neg x_{in}) \\ \wedge & (\neg x_{in} \vee x_{ch}) \wedge (x_{in} \vee \neg x_{ch}) \end{array}$$

La formule $A(\phi_t)$ est donc la formule :

$$\phi''_t = (x_{qu} \oplus x_{ba} = 1) \wedge (x_{me} \oplus x_{ba} \oplus x_{in} = 0) \wedge (x_{in} \oplus x_{ch} = 0)$$

qui est satisfaite par toutes les affectations de M_t ; on en déduit que M_t est descriptible dans AFFINE, et que ϕ''_t est une description convenable dans cette classe.

Comme nous l'avons annoncé, cet algorithme permet donc en une seule exécution :

- de repérer les variables indépendantes les unes des autres
- de repérer les variables dont M ne dépend pas réellement
- d'obtenir une description de M dans toutes les classes de formules (parmi celles qui nous intéressent) dans lesquelles il est descriptible
- dans tous les cas, y compris celui où M n'est descriptible dans aucune classe «intéressante», d'obtenir tout de même une formule première le décrivant et de taille comparable ou plus petite.

7.4 Description et identification optimisées

Nous revenons maintenant au problème de la description, et montrons que pour certaines classes, l'idée du paragraphe 7.3 peut bénéficier de l'hypothèse que la relation admet une description dans $\mathcal{C}$. L'idée générale reste la réduction des clauses d'une description en FNC de la relation initiale, mais guidée par la syntaxe de la classe $\mathcal{C}$. Pour ce qui est des classes NÉG et MONOT (paragraphe 7.4.1), à notre connaissance le problème n'avait jamais été étudié dans la littérature, et pour ce qui est de la classe HORN (paragraphe 7.4.2), déjà largement étudiée (voir paragraphe 7.1.3), l'algorithme que nous donnons est plus efficace que tous ceux connus jusqu'ici. Dans les deux cas, puisque l'on peut tester efficacement si un ensemble d'affectations donné est dans la classe, les optimisations décrites permettent également d'améliorer la complexité de l'*identification*.

7.4.1 Description en FNC négative ou monotone

Nous nous intéressons tout d'abord à la classe NÉG. On sait que la (seule) formule première décrivant un ensemble d'affectations descriptible dans cette classe est formée uniquement de clauses négatives.

Soient donc un ensemble de variables V et un ensemble d'affectations M à V, descriptible dans NÉG, et soit ϕ une formule FNC décrivant M. Si on retire des littéraux des clauses de ϕ, on obtient une formule logiquement plus forte, c'est-à-dire une formule FNC ϕ' avec $\phi' \models \phi$. Mais on sait également qu'il existe une formule FNC première ϕ'' logiquement équivalente à ϕ et que l'on peut obtenir à partir de ϕ en minimisant ses clauses, par exemple avec la procédure du paragraphe précédent ; ϕ'' étant première, on sait qu'on a retiré *au moins* tous les littéraux positifs de ϕ pour obtenir ϕ''. Supposons alors qu'on retire *exactement* les littéraux positifs de ϕ pour obtenir ϕ', on voit que $\phi'' \models \phi'$. On obtient donc $\phi'' \models \phi' \models \phi$, et comme on a $\phi'' \equiv \phi$, on a bien l'équivalence $\phi' \equiv \phi$, ϕ' étant négative. On obtient donc le résultat suivant ; pour résumer, l'hypothèse que M admet une description dans NÉG nous permet de réduire les clauses d'une description de M en FNC sans consulter M à nouveau.

Par exemple, si V_{tennis} est l'ensemble de variables $\{x_1, x_2, x_3\}$ du chapitre 2 (paragraphe 2.3), et M_{tennis} l'ensemble d'affectations à V_{tennis} :

$$\{000, 001, 010, 100\}$$

représentant tous les matches perdus, l'algorithme `DecrFNC` calcule sur la donnée M_{tennis} la formule FNC :

$$(x_1 \vee \neg x_2 \vee \neg x_3) \wedge (\neg x_1 \vee x_2 \vee \neg x_3) \wedge (\neg x_1 \vee \neg x_2)$$

Puisque l'ensemble M_{tennis} est descriptible dans NÉG (voir à nouveau le chapitre 2, paragraphe 2.3), la formule obtenue de celle-ci en retirant tous les littéraux positifs des clauses :

$$\phi_{tennis} = (\neg x_2 \vee \neg x_3) \wedge (\neg x_1 \vee \neg x_3) \wedge (\neg x_1 \vee \neg x_2)$$

est bien une formule FNC négative décrivant M_{tennis}. Notons en revanche que, même si c'est le cas ici, la formule ainsi obtenue n'est pas nécessairement première.

Proposition 7.6 (`Description[NÉG]`) *Soient V un ensemble de variables et $M \neq \emptyset$ un ensemble d'affectations à V descriptible dans* NÉG. *On peut calculer une description de M dans* NÉG *en temps $O(|M||V|^2)$, et cette description contient au plus $|M||V|$ clauses. Le problème* `Identification[NÉG]` *admet la même complexité.*

Preuve La procédure consiste à calculer tout d'abord une description ϕ de M en FNC avec l'algorithme `DecrFNC` (en temps $O(|M||V|^2)$, ϕ contenant au plus $|M||V|$ clauses), puis à retourner la formule ϕ' obtenue de ϕ en supprimant toutes les occurrences de littéraux positifs (en temps linéaire en la taille de ϕ, soit $O(|M||V|^2)$). Les remarques du début de ce paragraphe montrant que ϕ' est bien une description de M, et puisque ϕ' est par construction dans NÉG, on a le résultat pour la description.

Pour montrer le résultat pour l'identification, il faut montrer qu'on peut décider en temps $O(|M||V|^2)$ si M est descriptible dans NÉG ; si la réponse est positive, il suffira de calculer une description convenable comme ci-dessus. Nous avons vu (proposition 2.10 du chapitre 2) que M est descriptible dans NÉG si et seulement si pour toutes affectations $m \in M$ et $m' \in \{0,1\}^V$, si on a $m' \preceq m$ alors m' est également dans M. La transitivité de la relation $\preceq$ nous permet de vérifier seulement que pour toute affectation $m \in M$ et toute affectation m' à V *affectant* 1 *à*

une variable de moins que m on a $m' \in M$, soit au plus $|V|$ tests pour un $m \in M$ fixé, et donc $|M||V|$ tests au final. Nous concluons alors en rappelant qu'un arbre de décision représentant M permet de tester $m' \in M$ en temps $O(|V|)$, et qu'un tel arbre peut être construit une fois pour toutes en temps $O(|M||V|)$ (voir la preuve de la proposition 7.1). □

Evidemment, le résultat dual, pour la classe POS, est également vrai. Nous montrons que c'est également vrai pour la classe MONOT, avec une légère modification de la procédure. Il s'agit en effet de savoir avec quel signe chaque variable doit apparaître ; une fois cette information connue, on peut retirer d'une description en FNC de M les littéraux ayant le «mauvais» signe exactement comme on retire les littéraux positifs pour la classe NÉG.

Soient donc V un ensemble de variables et M un ensemble d'affectations à V descriptible dans MONOT. Soit $x \in V$ une variable. On voit facilement que x *doit* apparaître positivement dans une description de M si et seulement s'il existe une affectation $m \in M$ avec $m[x] = 1$ et telle que l'affectation m' ne différant de m qu'en x ne soit pas dans M ; le cas où x doit apparaître négativement est dual. Etant donné x, il faut donc vérifier pour tout $m \in M$ si l'affectation correspondante m' est ou non dans M. Afin que ces tests n'augmentent pas la complexité asymptotique de la description dans NÉG, il faut donc être capable de les réaliser tous en temps $O(|M||V|^2)$, et donc de rechercher m' dans M en temps $O(|V|)$ pour un m donné. Il suffit donc à nouveau de représenter M par un arbre de décision (voir le paragraphe 7.2 et la preuve de la proposition 7.6). Pour ce qui est de l'identification, les mêmes tests permettent de déterminer si un ensemble M, quelconque, d'affectations est descriptible dans MONOT (répondre «Non» si et seulement s'il existe une variable qui doit apparaître positivement *et* négativement). Finalement, nous obtenons donc le résultat suivant.

Proposition 7.7 (`Description[MONOT]`) *Soient V un ensemble de variables et $M \neq \emptyset$ un ensemble d'affectations à V descriptible dans* MONOT. *On peut calculer une telle description en temps $O(|M||V|^2)$, et cette description contient au plus $|M||V|$ clauses. Le problème* `Identification[MONOT]` *admet la même complexité.*

7.4.2 Description en FNC de Horn

Nous présentons enfin un algorithme réalisant la description dans la classe HORN d'un ensemble d'affectations M, supposé descriptible dans cette classe, à un ensemble de variables V en temps $O(|M||V|(|M| + |V|))$. Cet algorithme est de fait plus efficace que tous ceux connus jusqu'à maintenant. Afin de simplifier la présentation, nous définissons tout d'abord la formule calculée, et discutons les aspects algorithmiques ensuite.

Formule calculée

L'idée est, comme pour les classes NÉG et MONOT, d'obtenir des clauses (de Horn) subsumant celles de la formule calculée par l'algorithme `DecrFNC`, mais cette fois la description en FNC et la réduction des clauses sont menés en même temps. On sait tout d'abord que l'on peut garder tous les littéraux négatifs des clauses ; l'objectif est de garder au plus un littéral *positif* par clause.

Nous réutilisons les notions et les notations du paragraphe 7.2. Soient $V = \{x_1, x_2, \ldots, x_{|V|}\}$ un ensemble de variables et $M \neq \emptyset$ un ensemble d'affectations à V. De même que nous avions défini, pour tous $m \in M$ et $i \in \{1, 2, \ldots, |V|\}$ convenables, l'affectation partielle $p(m, i)$ à V, nous définissons, pour les mêmes m et i, l'affectation partielle $h(m, i)$ à V ; par construction,

$h(m,i)$ affectera 0 à au plus une variable de V, et la clause l'interdisant (de Horn) subsumera la clause interdisant $p(m,i)$.

Pour la définition de $h(m,i)$, nous distinguons trois cas.

Cas $i > prec(m)$ et $m[x_i] = 1$: Dans ce cas, par construction de $p(m,i)$ on sait que l'on a $p(m,i)[x_i] = 0$, et donc que la clause $cl(p(m,i))$ contient le littéral positif x_i ; nous montrons que c'est le littéral positif qu'il faut garder. On sait en effet, toujours par construction de $p(m,i)$, que l'on a $\forall i' < i, m[x_{i'}] = p(m,i)[x_{i'}]$; on en déduit que la clause $cl(p(m,i))\backslash\{x_i\}$ n'est pas satisfaite par m. Or, puisque M est descriptible dans HORN, et puisque la clause $cl(p(m,i))$ est satisfaite par tout $m \in M$, on sait qu'il existe une clause de Horn subsumant $cl(p(m,i))$ et satisfaite par tout $m \in M$; des remarques précédentes, on conclut que c'est le littéral positif x_i qu'il faut garder dans la clause $cl(p(m,i))$, et donc que la clause $cl(p(m,i))$ privée de tous ses littéraux positifs, sauf x_i, convient.

Formellement, nous définissons donc l'affectation $h(m,i)$ comme suit :

$$\begin{cases} \forall i' < i \text{ avec } m[x_{i'}] = 1, h(m,i)[x_{i'}] = 1 \\ h(m,i)[x_i] = 0 \\ \text{sinon, } h(m,i)[x_{i'}] = ? \end{cases}$$

Par exemple, considérons à nouveau l'ensemble d'affectations à $V_C = \{x_{PV}, x_{VP}, x_{VV}, x_{DT}\}$ de l'exemple du carrefour :

$$M_C = \{0000, 0010, 0101, 1000, 1010, 1101\}$$

On a $prec(0101) = 2$, donc on a $4 > prec(0101)$ et $0101[4] = 1$. On a donc $h(0101,4) = ?1?0$, interdit par la clause $cl(?1?0) = (\neg x_{VP} \vee x_{DT})$.

Il nous reste à traiter le cas $i > succ(m), m[x_i] = 0$. Nous distinguons à nouveau deux cas ; intuitivement, cette distinction sépare les clauses $cl(p(m,i))$ dans lesquelles nous garderons un littéral positif de celles dans lesquelles nous n'en garderons pas.

Pour cela, nous considérons les affectations de M qui ne satisfont aucun des littéraux négatifs de $cl(p(m,i))$, autrement dit l'ensemble :

$$I(m,i) = \{m' \in M \mid m'[x_i] = 1 \text{ et } \forall i' < i, m[x_{i'}] = 1 \Rightarrow m'[x_{i'}] = 1\}$$

Par exemple, considérons à nouveau l'ensemble M_C. On a $I(0000,4) = \{0101, 1101\}$ (avec $p(0000,4) = 0001$) et $I(0010,4) = \emptyset$ (avec $p(0010,4) = 0011$).

Cet ensemble $I(m,i)$ va nous permettre de calculer l'indice $sim(m,i)$ de l'éventuelle variable à laquelle garder l'affectation à 0 dans $p(m,i)$. Si $I(m,i)$ est vide, par convention nous posons $sim(m,i) = 0$, et sinon nous posons :

$$sim(m,i) = \max\{i' \in \{1,2,\ldots,|V|\} \mid \exists m' \in I(m,i), \forall i'' < i', m'[x_{i''}] = m[x_{i''}]\}$$

Sur l'exemple, on a donc $sim(0000,4) = 2$ et $sim(0010,4) = 0$. Nous pouvons alors traiter les deux derniers cas.

Cas $sim(m,i) = 0$ **:** Si $sim(m,i) = 0$, tout d'abord, par définition de $I(m,i)$ cela signifie que toutes les affectations de M satisfont au moins un littéral négatif de $cl(p(m,i))$; on n'a donc pas besoin de garder un littéral positif dans cette clause, et nous posons donc :

$$\begin{cases} \forall i' < i \text{ avec } m[x_{i'}] = 1, h(m,i)[x_{i'}] = 1 \\ h(m,i)[x_i] = 1 \\ \text{sinon, } h(m,i)[x_{i'}] = ? \end{cases}$$

Sur l'exemple, on obtient donc l'affectation partielle $h(0010, 4) = ??11$, interdit par la clause $cl(??11) = (\neg x_{VV} \vee \neg x_{DT})$.

Cas $sim(m,i) \neq 0$ **:** En revanche, si l'ensemble $I(m,i)$ est non vide, cela signifie qu'il faut garder une valeur à 0 dans l'affectation partielle $p(m,i)$; rappelons que nous savons, car M a une description dans HORN, que nous pouvons n'en garder qu'une. Nous choisissons de garder celle d'indice $sim(m,i)$:

$$\begin{cases} \forall i' < i \text{ avec } m[x_{i'}] = 1, h(m,i)[x_{i'}] = 1 \\ h(m,i)[x_i] = 1 \\ h(m,i)[x_{sim(m,i)}] = 0 \\ \text{sinon, } h(m,i)[x_{i'}] = ? \end{cases}$$

Sur l'exemple, nous obtenons donc l'affectation partielle $h(0000, 4) = ?0?1$, interdite par la clause $cl(?0?1) = (x_{VP} \vee \neg x_{DT}))$.

Nous devons cependant vérifier que cette définition n'est pas ambiguë, c'est-à-dire que l'on a $m[x_{sim(m,i)}] = 0$, afin que les différents cas de la définition ne soient pas en conflit pour cet indice. C'est l'objet du lemme suivant.

Lemme 7.3 *Soient V un ensemble de variables et M un ensemble d'affectations à V descriptible dans* HORN. *Soient $m \in M$ et $i \in \{1, 2, \ldots, |V|\}$ avec $i > succ(m)$ et $m[x_i] = 0$. Si on a $sim(m,i) \neq 0$, alors on a $m[x_{sim(m,i)}] = 0$.*

Preuve Supposons par l'absurde $m[x_{sim(m,i)}] = 1$. Soit alors $m' \in I(m,i)$ une affectation à V vérifiant $\forall i' < sim(m,i), m'[x_i] = m[x_i]$. Par définition de $I(m,i)$, et puisque l'on a, par définition de $succ(m)$, $sim(m,i) \leq succ(m) < i$, on a $m'[x_{sim(m,i)}] = 1$, et donc $\forall i' < sim(m,i) + 1, m'[x_{i'}] = m[x_{i'}]$, ce qui contredit la maximalité de $sim(m,i)$. □

La définition n'est donc pas ambiguë. Nous prouvons maintenant qu'elle correspond bien à notre objectif.

Lemme 7.4 *Soient V un ensemble de variables et M un ensemble d'affectations à V descriptible dans* HORN. *Soient $m \in M$ et $i \in \{1, 2, \ldots, |V|\}$ avec $i > succ(m)$ et $m[x_i] = 0$. Si on a $sim(m,i) \neq 0$, alors on a $\forall m' \in M, m' \models cl(h(m,i))$.*

Preuve Supposons par l'absurde qu'il existe une affectation $m' \in M$ ne satisfaisant pas la clause $cl(h(m,i))$. On a donc par définition de cette clause $m' \in ext(h(m,i))$. Soit alors une affectation $m'' \in I(m,i)$ avec $\forall i' \leq sim(m,i), m''[x_{i'}] = m[x_{i'}]$. Puisque M est descriptible dans HORN, l'affectation $m_0 = m' \wedge m''$ est dans M (proposition 2.2 du chapitre 2). Mais on a $\forall i' < i, m[x_{i'}] = 1 \Longrightarrow m'[x_{i'}] = 1$ et $m''[x_{i'}] = 1$, d'où $m_0[x_{i'}] = 1$; d'autre part, on a $\forall i' < sim(m,i), m[x_{i'}] = 0 \Longrightarrow m''[x_{i'}] = 0 \Longrightarrow m_0[x_{i'}] = 0$. Finalement, on a $\forall i' < sim(m,i), m_0[x_{i'}] = m[x_{i'}]$. Mais comme on a $m' \in ext(h(m,i))$, on a également $m'[x_{sim(m,i)}] = 0$, et donc $m_0[x_{sim(m,i)}] = 0$. On a donc $\forall i' < sim(m,i) + 1, m_0[x_{i'}] = m[x_{i'}]$, ce qui contredit la maximalité de $sim(m,i)$. □

La formule DescH(M) Nous pouvons donc désormais définir la formule

$$DescH(R) = fnc(\{h(m,i) \mid \text{soit } i > prec(m) \text{ et } m[x_i] = 1,\ \text{soit } i > succ(m) \text{ et } m[x_i] = 0\})$$

Ainsi, pour l'exemple du carrefour nous obtenons la formule :

$$DescH(M_C) = \begin{array}{llll} & (x_{VP} \vee \neg x_{DT}) & \wedge & (\neg x_{VV} \vee \neg x_{DT}) \\ \wedge & (\neg x_{VP} \vee x_{DT}) & \wedge & (\neg x_{VP} \vee \neg x_{VV}) \\ \wedge & (\neg x_{PV} \vee x_{VP} \vee \neg x_{DT}) & \wedge & (\neg x_{PV} \vee \neg x_{VV} \vee \neg x_{DT}) \\ \wedge & (\neg x_{PV} \vee \neg x_{VP} \vee x_{DT}) & \wedge & (\neg x_{PV} \vee \neg x_{VP} \vee \neg x_{VV}) \end{array}$$

qui est bien de Horn, et décrit bien l'ensemble d'affectations M_C. Notons encore une fois que la formule ainsi obtenue n'est pas nécessairement première ; ainsi, la formule ci-dessus est logiquement équivalente à la formule :

$$(x_{VP} \vee \neg x_{DT}) \wedge (\neg x_{VV} \vee \neg x_{DT}) \wedge (\neg x_{VP} \vee x_{DT}) \wedge (\neg x_{VP} \vee \neg x_{VV})$$

obtenue de la formule $DescH(M_C)$ en retirant les clauses subsumées par d'autres.

Finalement, nous avons donc le résultat suivant.

Proposition 7.8 ($DescH(R)$) *Soient V un ensemble de variables, et M un ensemble d'affectations à V descriptible dans* HORN. *La formule FNC $DescH(M)$ est une telle description, et contient au plus $|M||V|$ clauses.*

Calcul de la formule

Nous prouvons maintenant qu'étant donné M, la formule DescH(M) peut être calculée en temps $O(|M||V|(|M| + |V|))$. Notons tout d'abord que si $prec(m)$, $succ(m)$ et $sim(m,i)$ sont connus pour tous $m \in M$ et tous $i \in \{1, 2, \ldots, |V|\}$ convenables, le calcul de cette formule requiert un temps $O(|M||V|^2)$ en appliquant directement les définitions. Nous avons vu pour l'algorithme `DecrFNC` que le calcul de $prec(m)$ et $succ(m)$ ne posait pas de problème ; le seul point difficile est donc le calcul de $sim(m,i)$.

Remarquons qu'en calculant cette quantité naïvement, on obtient une complexité $O(|M||V|)$ pour m et i fixés, et donc une complexité $O(|M|^2|V|^2)$ pour tous les $sim(m,i)$. Nous montrons que pour $m \in M$ fixée, on peut calculer en temps $O(|M||V|)$ la valeur de $sim(m,i)$ pour tous les i convenables, et donc que l'on peut calculer la valeur de tous les $sim(m,i)$ en temps total $O(|M|^2|V|)$.

Nous proposons en effet l'algorithme `SIM` présenté sur la figure 7.5. Le principe de cet algorithme est d'identifier, pour $m \in M$ et $i \in \{1, 2, \ldots, |V|\}$ fixés, si un $m' \in M$ donné est dans l'ensemble $I(m,i)$.

En effet, par définition de cet ensemble, si pour la valeur courante de i on a $m[x_i] = 1$ et $m'[x_i] = 0$, alors on a bien $\forall i' > i, m' \notin I(m,i')$; on peut donc laisser la valeur de $sim(m,i)$ inchangée pour tous $i' > i$. Sinon, on est sûr que m' est dans l'ensemble $I(m,i)$, et on peut mettre la valeur de $sim(m,i)$ à jour en fonction de m'. Notons que l'algorithme `SIM` retourne une valeur $sim(m,i)$ pour tout $i \in \{1, 2, \ldots, |V|\}$, mais que cela n'a un sens que pour i vérifiant $i > succ(m)$ et $m[x_i] = 0$.

Nous pouvons finalement établir la complexité totale de l'algorithme, qui est résumé dans la proposition suivante.

Entrée : Un ensemble d'affectations M, descriptible dans HORN, à un ensemble de variables V, et une affectation $m \in M$
Sortie : La valeur de $sim(m,i)$ pour tout $i \in \{1,2,\ldots,|V|\}$ avec $i > succ(m)$ et $m[x_i] = 0$.

Début
  Pour $i = 1,2,\ldots,|V|$ **Faire** $sim(m,i) \leftarrow 0$ **Fin Pour** ;
  Pour tout $m' \in M$ **Faire**
    $i_0 \leftarrow \max\{i \in \{1,2,\ldots,|V|\} \mid \forall i' < i, m'[x'_i] = m[x'_i]\}$;
    $i \leftarrow i_0$;
    Tant Que $m'[x_i] = 1$ ou $m[x_i] = 0$ **Faire**
      Si $i > succ(m)$, $m[x_i] = 0$ et $m'[x_i] = 1$ **Alors**
        $sim(m,i) \leftarrow \max(sim(m,i), i_0)$;
      Fin Si ;
      $i \leftarrow i + 1$;
    Fin Tant Que ;
  Fin Pour ;
Fin

FIG. 7.5 – Algorithme `SIM`

Proposition 7.9 (`Description[HORN]`) *Soient V un ensembles de variables, et $M \neq \emptyset$ un ensemble d'affectations à M descriptible dans* HORN. *On peut calculer une telle description en temps $O(|M||V|(|M| + |V|))$, et cette description contient au plus $O(|M||V|)$ clauses. Le problème* `Identification[HORN]` *admet la même complexité.*

Preuve Il suffit de montrer que l'algorithme `SIM` est correct ; en effet, sa complexité est en $O(|M||V|)$ (c'est direct sur la figure 7.5), et la discussion du début de ce paragraphe conclut. Or, pour montrer que l'algorithme `SIM` est correct, par définition de la valeur $sim(m,i)$ il suffit de montrer que l'instruction :

$$sim(m,i) \leftarrow \max(sim(m,i), i_0) ;$$

est exécutée si et seulement si on a $m' \in I(m,i)$, puisque la valeur donnée à i_0 par l'algorithme est bien $\max\{i \in \{1,2,\ldots,|V|\} \mid \forall i' < i, m'[x'_i] = m[x_i]\}$. Mais cette condition est assurée par la condition de la boucle **Tant Que** de l'algorithme, et nous avons le résultat. □

Corollaire 7.1 (`Description[HDP]`) *Soient V un ensembles de variables, et $M \neq \emptyset$ un ensemble d'affectations à M descriptible dans* HDP. *On peut calculer une telle description en temps $O(|M||V|(|M| + |V|))$, et cette description contient au plus $O(|M||V|)$ clauses. Le problème* `Identification[HDP]` *admet la même complexité.*

Bien évidemment, les résultats duaux, pour les classes ANTI-HORN et ANTI-HDP, sont tout aussi vrais.

Pour terminer, il est bien sûr très intéressant, en plus d'être capable de décrire un ensemble d'affectations en une formule FNC d'une certaine classe, d'être capable de calculer une telle description *de taille minimale* parmi toutes (voir chapitre 6). Pour la classe des formules FNC négatives (ou celle des formules positives), nous avons vu dans le chapitre 2 qu'il existait une seule représentation en FNC première d'une théorie représentable dans cette classe. On peut donc en déduire que la formule calculée par l'algorithme `DecrIP`, lorsqu'elle est négative (resp. positive)

est minimale. Pour ce qui est de la classe HORN, les résultats de [HK92, HK95] permettent de réduire une formule FNC de Horn ϕ en une autre formule FNC de Horn, comportant au plus $|V|-1$ fois plus de clauses que la plus petite formule de Horn équivalente, en temps quadratique ; cette «minimisation» passe notamment par la suppression, une à une, des clauses *redondantes* de la formule, c'est-à-dire des clauses qui sont impliquées par la conjonction des autres clauses de la formule courante. L'algorithme `DecrIP`, comme celui présenté dans ce paragraphe, calculant une formule contenant moins de $|M||V|$ clauses, on peut dans les deux cas obtenir une formule «quasi-minimale» (dans le sens évoqué plus haut) en temps $O(|M|^2|V|^4)$. Notons qu'on obtient ainsi un algorithme plus efficace, mais aussi calculant une formule au moins aussi courte, que tous les algorithmes connus jusqu'ici. Remarquons enfin qu'en revanche, Hammer et Kogan [HK93] montrent que le problème consistant à minimiser une formule de Horn (c'est-à-dire à trouver *la* plus courte formule de Horn logiquement équivalente) admet un problème de décision associé NP-complet.

7.5 Autres classes de formules et résumé des résultats connus

Nous discutons enfin la description et l'identification pour les classes de formules *FND* qui nous intéressent. Nous prenons l'exemple de la classe HORN-FND pour montrer le lien entre les problèmes pour les classes de formules FND et pour celles de formules FNC ; le raisonnement similaire pourra être appliqué à toutes les autres classes de formules FND, sauf les classes HDP-REN-FND et DISJAFFINE, présentées dans la partie I.

Considérons donc la classe HORN-FND. Nous avons remarqué (chapitre 5, paragraphe 5.3) qu'un ensemble d'affectations à un ensemble de variables V était descriptible dans cette classe si et seulement si son complémentaire dans $\{0,1\}^V$ était descriptible dans la classe de formules FNC ANTI-HORN. Soient donc un ensemble de variables V et un ensemble d'affectations M à V. On peut tout d'abord calculer une formule FNC décrivant $\{0,1\}^V \backslash M$ en temps linéaire : la formule FNC interdisant M est en effet une telle formule, et elle contient exactement $|M|$ clauses. Ainsi, pour l'exemple du carrefour, le complémentaire de l'ensemble d'affectations :

$$M_C = \{0000, 0010, 0101, 1000, 1010, 1101\}$$

dans l'ensemble $\{0,1\}^{V_C}$ est décrit par la formule FNC interdisant M_C :

$$\begin{array}{llll} & (x_{PV} \vee x_{VP} \vee x_{VV} \vee x_{DT}) & \wedge & (x_{PV} \vee x_{VP} \vee \neg x_{VV} \vee x_{DT}) \\ \wedge & (x_{PV} \vee \neg x_{VP} \vee x_{VV} \vee \neg x_{DT}) & \wedge & (\neg x_{PV} \vee x_{VP} \vee x_{VV} \vee x_{DT}) \\ \wedge & (\neg x_{PV} \vee x_{VP} \vee \neg x_{VV} \vee x_{DT}) & \wedge & (\neg x_{PV} \vee \neg x_{VP} \vee x_{VV} \vee \neg x_{DT}) \end{array}$$

Cependant, on ne peut pas transformer cette formule en une formule *première* de la même manière qu'avec l'algorithme `DecrIP` ; en effet, l'ensemble d'affectations à décrire, autrement dit $\{0,1\}^V \backslash M$, ne nous est pas donné. Nous devons donc, comme pour la construction de la remarque 7.2, décider si on supprime chaque littéral de la clause courante en reconsultant M à chaque fois, et en vérifiant qu'il contient bien, et donc que son complémentaire ne contient pas, les affectations qui interdiraient la suppression de ce littéral ; il faut donc compter des affectations de M, en temps $O(|M||V|)$ pour chaque littéral de chaque clause.

Ainsi, en reprenant la formule de l'exemple ci-dessus, on doit tout d'abord compter combien d'affectations de M empêcheraient la suppression du littéral x_{PV} de la première clause. Ce sont les affectations qui ne satisfont pas la clause $(x_{VP} \vee x_{VV} \vee x_{DT})$, c'est-à-dire cette clause privée du littéral x_{PV}, et on trouve les affectations 0000 et 1000 dans M ; M contient donc toutes les affectations à V_C qui empêcheraient de supprimer x_{PV} de la clause, et donc son complémentaire

n'en contient aucune : comme c'est lui que nous décrivons, nous pouvons ainsi retirer x_{PV} de la première clause ci-dessus. En revanche, nous ne pouvons pas retirer ensuite son littéral x_{VP}, car parmi les affectations ne satisfaisant pas la clause $(x_{VV} \vee x_{DT})$, M ne contient que les deux affectations 0000 et 1000 ; son complémentaire contient donc deux affectations ne satisfaisant pas cette clause. Après avoir effectué ce calcul pour tous les littéraux de chaque clause, on obtient la formule en FNC :

$$(x_{VP} \vee x_{DT}) \wedge (\neg x_{VP} \vee x_{VV} \vee \neg x_{DT})$$

qui décrit bien $\{0,1\}^{V_C} \backslash M_C$, et qui est première (par construction), et donc la formule en FND obtenue de celle-ci par la transformation $Neg(.)$:

$$(\neg x_{VP} \wedge \neg x_{DT}) \vee (x_{VP} \wedge \neg x_{VV} \wedge x_{DT})$$

est première et décrit M. On voit ainsi que M n'est pas descriptible dans Horn-FND, car cette formule est première mais ne fait pas partie de cette classe, mais qu'il est descriptible dans Horn-Ren-FND (renommer x_{DT} par exemple).

Finalement, une formule FND première décrivant M peut être calculée en temps $O(|M|^2|V|^2)$, puisque la formule $fnc(M)$ ne contient que $|M|$ clauses. Ces remarques montrent que l'algorithme `DecrIP`, légèrement modifié, permet de décrire et d'identifier pour les classes de formules FND :

Horn-FND, Anti-Horn-FND, HDP-FND, Anti-HDP-FND
Nég-FND, Pos-FND, Bij-FND, Horn-Ren-FND, Monot-FND

dans un ensemble d'affectations M à un ensemble de variables V en temps $O(|M|^2|V|^2)$, l'éventuelle formule calculée contenant au plus $|M|$ termes.

Pour la classe DisjAffine, en revanche, nous devons encore une fois modifier cette procédure générale. Nous venons de montrer qu'étant donné un ensemble d'affectations M à un ensemble de variables V, on pouvait calculer en temps $O(|M|^2|V|^2)$ une formule FNC première ϕ décrivant $\{0,1\}^V \backslash M$. Nous considérons alors la formule affine $aff(\phi)$, qui peut être obtenue en temps linéaire. Puisque l'ensemble des modèles de $aff(\phi)$ n'est pas accessible directement à l'algorithme, nous ne pouvons tester $\forall m \in M, m \models aff(\phi)$. Nous utilisons alors la proposition 3.3 du chapitre 3 pour compter en temps $O(|\phi|^2|Var(\phi)|)$ le nombre d'affectations à V qui satisfont ϕ, et il ne reste qu'à vérifier que ce nombre est bien égal à $2^{|V|} - |M|$; en effet, c'est le cas si et seulement si la formule $aff(\phi)$ décrit $\{0,1\}^V \backslash M$ (comme pour la proposition 7.5). Il ne reste plus, le cas échéant, qu'à remplacer dans $aff(\phi)$ la conjonction par la disjonction et à y inverser les membres droits des équations pour obtenir une description de M dans DisjAffine. De plus, le calcul supplémentaire requiert un temps $O(|\phi|^2|Var(\phi)|)$, et donc $O(|M|^2|V|)$ puisque l'on a $|\phi| \leq |M|$. Finalement, on peut bien décrire et identifier pour la classe DisjAffine en temps $O(|M|^2|V|^2)$. Notons cependant que pour la description, la dernière étape (celle consistant à compter les modèles) n'est pas nécessaire, puisque par hypothèse M est descriptible dans DisjAffine.

Quant aux optimisations du paragraphe 7.4 pour la description, il est clair que la même complexité est valable pour les classes de formules FND Pos-FND, Nég-FND et Monot-FND (le signe de chaque variable, pour le cas monotone, étant recherché sur M puis inversé pour la description de $\{0,1\}^V \backslash M$). En revanche, il ne semble pas que le calcul d'une formule FND de Horn (ou anti-Horn) décrivant M puisse être optimisé de même. Enfin, nous avons laissé de côté la classe HDP-Ren-FND ; pour la description, par hypothèse il suffit de calculer une formule FND première, comme ci-dessus, mais il ne semble pas que l'on puisse adapter les procédures décrites dans ce chapitre pour l'*identification* de cette classe[2].

[2] C'est cependant un moindre mal, la NP-complétude du test de reconnaissance des formules de cette classe nous l'ayant fait considérer comme peu intéressante.

Classe	Description/Identification	Nombre de clauses	Référence
HORN, ANTI-HORN	$O(\|M\|\|V\|(\|M\|+\|V\|))$	$\leq \|M\|\|V\|$	*
HDP, ANTI-HDP	$O(\|M\|\|V\|(\|M\|+\|V\|))$	$\leq \|M\|\|V\|$	*
NÉG, POS, MONOT	$O(\|M\|\|V\|^2)$	$\leq \|M\|\|V\|$	*
BIJ	$O(\|M\|\|V\|^2)$	$\leq \frac{3}{2}\|V\|^2$	[DP92]
AFFINE	$O(\|M\|\|V\|^3)$	$\leq \|V\|$	* (folklore)
HORN-REN, HDP-REN	$O(\|M\|^2\|V\|^2)$	$\leq \|M\|\|V\|$	*

TAB. 7.1 – Complexité de la description et de l'identification

Nous résumons les meilleures complexités (en temps) connues pour la description et l'identification des classes de formules qui nous intéressent dans la table 7.1 : puisque pour toutes ces classes les complexités de la description et de l'identification sont les mêmes, nous ne les différencions pas ; nous indiquons également le nombre de clauses (ou d'équations linéaires) de la formule produite par l'algorithme de cette complexité. Nous omettons les classes de formules FND, en renvoyant à la discussion ci-dessus, puisque ces classes ne nous intéressent pas réellement pour représenter explicitement des connaissances.

Chapitre 8

Approximation

Sommaire

8.1 Présentation et travaux précédents 101
8.1.1 Formalisation des notions d'approximation 102
8.1.2 Motivations 103
8.1.3 D'autres approches pour approximer 106
8.1.4 Travaux précédents 106
8.2 Approximations faibles 108
8.2.1 Caractérisations 108
8.2.2 Calcul d'une approximation faible 110
8.3 Approximations fortes 114
8.3.1 Calcul d'une approximation forte maximale quelconque 116
8.3.2 Calcul d'une approximation forte avec le nombre maximal de modèles 120
8.4 Résumé des résultats connus 124

Que faire pour intégrer des connaissances non représentables dans une certaine classe de formules à un système de gestion de connaissances n'acceptant que des formules de cette classe ? Selman et Kautz [SK96] proposent deux notions d'*approximation* dans une classe de formules donnée (duales l'une de l'autre) ; nous nous intéressons dans ce chapitre à ces notions et à leur algorithmique lorsque la connaissance à intégrer est donnée par un ensemble d'exemples ou de situations envisageables. Nous rappelons et complétons les résultats déjà connus concernant les approximations *faibles*, puis ceux concernant les approximations *fortes* ; nous donnons notamment, pour ces dernières, un algorithme nouveau et efficace pour les classes de formules «renommables» qui nous intéressent. Nous montrons également que les approximations d'ensembles d'affectations dans la classe des formules affines peuvent être calculées plus efficacement que pour les autres classes qui nous intéressent.

8.1 Présentation et travaux précédents

Nous présentons tout d'abord les formalisations des problématiques d'approximation proposées par Selman et Kautz, puis les principales motivations de ces notions, et enfin un bref état de l'art sur ce sujet.

8.1.1 Formalisation des notions d'approximation

Les notions proposées par Selman et Kautz, quoiqu'initialement présentées en termes de formules, sont des notions *sémantiques*. Elles concernent donc principalement des ensembles d'affectations. Etant donnée une classe de formules $\mathcal{C}$, l'idée est d'approximer un ensemble d'affectations M par une formule $\phi \in \mathcal{C}$ dont l'ensemble de modèles soit aussi proche que possible de M. Il s'agit donc d'ajouter (virtuellement) un nombre *minimum* d'affectations à M pour obtenir un ensemble descriptible dans $\mathcal{C}$: on parle alors d'*approximation faible* de M dans $\mathcal{C}$, la formule obtenue étant logiquement plus faible qu'une description de l'ensemble d'affectations donné ; dualement, il peut s'agir également de supprimer un nombre minimal d'affectations de M, et on parle alors d'*approximation forte* de M dans $\mathcal{C}$.

Formellement, nous définissons les approximations faibles et fortes d'un ensemble d'affectations dans une classe de formules comme suit. Nous gardons les abbréviations anglaises *(L)UB* («(Least) Upper Bound») et *(G)LB* («(Greatest) Lower Bound») car elles sont devenues classiques.

Définition 8.1 (approximation faible, LUB [SK96]) *Soient M un ensemble d'affectations et $\mathcal{C}$ une classe de formules. Une formule $\phi \in \mathcal{C}$ est appelée une* $\mathcal{C}$-approximation faible *(*C-UB*) de M si toutes les affectations de M la satisfont. Si, de plus, M n'admet aucune $\mathcal{C}$-UB logiquement plus forte (strictement) que ϕ, on dit que ϕ est une $\mathcal{C}$-approximation faible* minimale *(*C-LUB*) de M.*

Définition 8.2 (approximation forte, GLB [SK96]) *Soient M un ensemble d'affectations à un ensemble de variables V et $\mathcal{C}$ une classe de formules. Une formule $\phi \in \mathcal{C}$ est appelée une* $\mathcal{C}$-approximation forte *(*C-LB*) de M si toutes les affectations à V satisfaisant ϕ sont dans M. Si, de plus, M n'admet aucune $\mathcal{C}$-LB logiquement plus faible (strictement) que ϕ, on dit que ϕ est une $\mathcal{C}$-approximation forte* maximale *(*C-GLB*) de M.*

Dans le cas où les approximations sont formées sur le même ensemble de variables que M, une $\mathcal{C}$-UB ϕ de M est donc une formule de $\mathcal{C}$ vérifiant $M \subseteq \mathcal{M}(\phi)$; c'est une $\mathcal{C}$-LUB de M si pour aucune formule ϕ' de $\mathcal{C}$ on n'a $M \subseteq \mathcal{M}(\phi') \subset \mathcal{M}(\phi)$. Pour les approximations fortes les notions sont duales.

Ainsi, pour l'exemple de la thèse, on voit que la formule FNC de Horn :

$$\phi_t''' = \begin{array}{ll} & (x_{in} \vee \neg x_{ch}) \wedge (\neg x_{in} \vee x_{ch}) \\ \wedge & (x_{me} \vee \neg x_{qu} \vee \neg x_{ch}) \wedge (\neg x_{me} \vee x_{qu} \vee \neg x_{ch}) \wedge (\neg x_{me} \vee \neg x_{qu} \wedge x_{ch}) \\ \wedge & (x_{me} \vee \neg x_{qu} \vee \neg x_{ba}) \wedge (\neg x_{qu} \vee \neg x_{ba} \vee \neg x_{ch}) \end{array}$$

est une HORN-UB de l'ensemble d'affectations M_t. En effet, c'est bien une formule FNC de Horn, et son ensemble de modèles :

$$M_t''' = \{00111, 01000, 10100, 11011, 00000, 00100, 00011, 10000\}$$

inclut M_t. C'est même une HORN-*LUB* de M_t, car on ne peut trouver aucun sous-ensemble propre de M_t''' qui à la fois inclue M_t et soit descriptible dans HORN, comme nous le verrons plus loin. Dualement, la formule FNC de Horn :

$$(x_{qu}) \wedge (\neg x_{ba}) \wedge (\neg x_{in} \vee x_{ch}) \wedge (x_{in} \vee \neg x_{ch}) \wedge (\neg x_{me} \vee x_{ch}) \wedge (x_{me} \vee \neg x_{ch})$$

dont l'ensemble de modèles $\{01000, 11011\}$ est inclus dans M_t, est une HORN-LB (et même une HORN-*GLB*) de M_t. Pour ce qui est de l'exemple du carrefour, la formule affine réduite à une équation :

$$(x_{VP} \oplus x_{DT} = 0)$$

dont l'ensemble de modèles est l'ensemble $\{00, 11\}$ (sur l'ensemble de ses variables $\{x_{VP}, x_{DT}\}$), est une AFFINE-LUB de M_C, et la formule affine :

$$(x_{VP} \oplus x_{VV} = 1) \wedge (x_{VP} \oplus x_{DT} = 0)$$

dont l'ensemble de modèles est $\{010, 101\}$ en est une AFFINE-GLB.

On voit facilement que les approximations réellement intéressantes du point de vue de l'acquisition de connaissances sont les approximations faibles *minimales* et les approximations fortes *maximales*. A titre d'illustration, notons par exemple que la formule FNC insatisfaisable (()) est notamment une HORN-, BIJ- et HORN-REN-approximation forte de tout ensemble d'affectations à un ensemble quelconque de variables ; dualement, la formule FNC tautologique () est, de même, notamment une HORN-, BIJ- et HORN-REN-approximation faible de tout ensemble d'affectations. Or, on voit que ces approximations ont peu d'intérêt, ne préservant que très peu la sémantique des connaissances de départ. Au contraire, nous verrons au chapitre 12 que les $\mathcal{C}$-approximations faibles *minimales* et fortes *maximales* préservent de nombreuses propriétés sémantiques.

8.1.2 Motivations

Comme nous l'avons dit, l'approximation dans une classe de formules prend naturellement place lorsque l'on veut intégrer une connaissance à un système n'acceptant des formules que d'une certaine classe, dans laquelle on ne peut pas représenter cette connaissance *exactement*, et où l'on s'autorise donc une perte d'information afin de pouvoir l'intégrer tout de même. Dans notre cadre, nous nous intéressons donc au cas où les classes de formules dans lesquelles on approxime sont celles présentées et motivées dans la partie I.

Pour ce qui est des motivations de ces notions, elles résident principalement dans les utilisations possibles de bases de connaissances approximatives ; elles sont au nombre de trois, et nous les exposons une par une dans la suite.

Vision «confiante» La première approche consiste à «oublier» que la base de connaissances que l'on a acquise par approximation ne représente pas exactement l'ensemble d'exemples de départ, et ainsi à faire confiance, en quelque sorte, à l'approximation calculée ; cette approche est notamment discutée par Valiant [Val84] et Khardon et Roth [KR94, KR97]. On considère alors que cette base de connaissances représente exactement la réalité qu'elle est censée décrire, ou tout du moins on l'utilise comme si c'était le cas. Cette vision de l'approximation ne nécessite donc pas de remise en cause des processus de raisonnement ou, plus généralement, d'utilisation des connaissances. C'est en particulier une approche couramment utilisée en apprentissage (où il s'agit également, en général, d'apprendre une approximation de la réalité), où l'on considère notamment qu'un (contre-)exemple que l'on n'a pas rencontré, ou pas pris en compte, en apprenant n'impliquera pas d'erreurs significatives de raisonnement, car cela signifie qu'il n'est pas lui-même très représentatif du petit monde auquel il appartient. Plus concrètement, une distribution de probabilités est supposée sur les exemples, et cette même distribution est utilisée pour à la fois :

- Formaliser la fréquence des exemples, c'est-à-dire la probabilité qu'ils soient rencontrés, ou observés, durant l'apprentissage

- Mesurer la qualité de l'approximation calculée en acquérant les connaissances, admettant ainsi qu'on prenne peu en compte les exemples de faible probabilité
- Mesurer la qualité du raisonnement effectué avec la base de connaissances approximative, accordant ainsi peu d'importance aux erreurs commises à cause des exemples de faible probabilité.

Cette distribution de probabilités supposée n'est, bien entendu, qu'un outil pour étudier la qualité des algorithmes d'approximation ou d'apprentissage ; elle n'a pas de représentation dans la base de connaissances.

A titre d'exemple, considérons l'approximation de l'ensemble d'affectations M_C de l'exemple du carrefour par la formule affine $(x_{VP} \oplus x_{VV} = 1) \wedge (x_{VP} \oplus x_{DT} = 0)$ (voir paragraphe 8.1.1). Nous avons vu que l'ensemble de modèles de cette formule était l'ensemble $\{010, 101\}$ (sur l'ensemble de variables $\{x_{VP}, x_{VV}, x_{DT}\}$). L'information perdue par le processus d'approximation est donc le fait que les situations où ni le feu pour piétons, ni le feu pour voitures ne sont verts, et où l'on n'a pas le droit de traverser, sont envisageables. Remarquons que cette situation représente intuitivement la période de latence avant le passage de l'un des deux feux au verts, et est donc la situation la moins fréquente dans la réalité. On peut alors déduire de la base de connaissances approximative, si on la considère comme décrivant exactement la réalité, qu'il y a toujours un et un seul des deux feux au vert. Ce n'est pas vrai, puisque l'ensemble initial de situations envisageables montre le contraire, mais l'erreur commise en raisonnant ainsi n'a de conséquences que dans peu de cas : la plupart du temps, c'est vrai. A l'inverse, si en approximant on perd, par exemple, l'information que des situations dans lesquelles le feu pour voitures est vert sont envisageables, on pourra alors déduire de la base de connaissances approximative qu'il ne l'est jamais. L'erreur ainsi commise aura donc des conséquences significatives dans de nombreux cas, à la mesure de l'importance des informations perdues.

Nous reviendrons sur cette vision des approximations dans le chapitre 9, mais renvoyons également le lecteur aux discussions très intéressantes sur le sujet de Valiant [Val84] et Khardon et Roth [KR94, KR97].

Vision «méfiante» La deuxième approche pour utiliser des connaissances acquises par approximation consiste à «se souvenir» du fait que la base de connaissances ne représente pas exactement la réalité qu'elle est censée décrire, et à se souvenir des liens qu'elle a avec elle ; par exemple, que c'en est une approximation faible. C'est la piste qu'explorent, tout d'abord, Selman et Kautz, en exemplifiant ce principe pour la déduction. Nous rappelons ici brièvement ce principe.

Soit M un ensemble d'affectations représentant un ensemble d'exemples, ou de situations envisageables dans un petit monde donné, soit $\mathcal{C}$ une classe de formules, et soit ϕ_{ub} (resp. ϕ_{lb}) une $\mathcal{C}$-approximation faible (resp. forte) de M. Soit une requête α (une formule propositionnelle), et soit la question : «a-t-on $\forall m \in M, m \models \alpha$?». Autrement dit, on veut décider si la formule α est toujours vraie dans notre petit monde de référence, tout en ne connaissant cette réalité que par le biais des deux approximations ϕ_{ub} et ϕ_{gb}. En supposant la classe $\mathcal{C}$ traitable pour la déduction, et les approximations de taille raisonnable, on peut décider si ϕ_{ub} et/ou ϕ_{lb} impliquent α efficacement. En fonction des réponses obtenues, on peut alors déduire les réponses suivantes.

Si ϕ_{ub} implique α, tout d'abord, cela signifie par définition de l'implication que tous les modèles de ϕ_{ub} satisfont α ; puisque, par définition d'une approximation faible, toutes les affectations de M sont des modèles de ϕ_{ub}, on en déduit donc que toutes satisfont α, et donc que la réponse à notre question est positive. Dualement, si ϕ_{lb} *n'*implique *pas* α, on peut en déduire que la réponse à notre question est négative. Restent donc les cas où ϕ_{ub} n'implique pas α, mais ϕ_{lb}, si. Dans

ce cas, on peut être amené à répondre «Je ne sais pas», ou encore à reprendre le calcul avec une description exacte de M, si on a pu en stocker une, éventuellement en utilisant plus de ressources. Si l'on choisit de répondre «Je ne sais pas», on a donc un processus de raisonnement cohérent, non complet mais efficace (en supposant, bien entendu, la classe d'approximation traitable pour la déduction, et les tailles de ϕ_{ub} et ϕ_{lb} raisonnables), et si l'on choisit de reprendre le calcul avec une description exacte de M, un processus cohérent, complet, mais efficace dans une partie des cas seulement. Pour résumer, dans cette approche on décide donc de la validité d'une réponse calculée avec une approximation *après* le calcul, en fonction de son résultat.

Pour illustrer ce principe, reprenons l'exemple du carrefour et ses AFFINE-approximations du paragraphe 8.1.1. Supposons que l'on veuille savoir si dans ce petit monde, la requête $\alpha = (\neg x_{VP} \vee x_{DT})$ («Si le feu pour piétons est vert, on a le droit de traverser») est toujours satisfaite. On voit que tous les modèles de l'AFFINE-LUB $((x_{VP} \oplus x_{DT} = 0))$ de M_C satisfont cette requête, et on peut donc en déduire que la réponse est également positive pour la «vraie» base de connaissances que représente M_C. Dualement, on voit que *même* dans les situations envisagées par la base de connaissances représentée par l'AFFINE-GLB $((x_{VP} \oplus x_{VV} = 1) \wedge (x_{VP} \oplus x_{DT} = 0))$ de M_C, la requête $(\neg x_{VV} \vee x_{DT})$ («Si le feu pour voitures est vert, on a le droit de traverser») n'est pas satisfaite, et on peut en déduire que la vraie base de connaissances ne l'implique pas. Enfin, la requête $\neg(x_{VP} \wedge x_{VV})$ (les deux feux ne peuvent pas être verts en même temps) n'est pas impliquée par l'approximation faible de M_C, mais l'est par son approximation forte ; on ne peut donc rien conclure. Intuitivement, on ne peut pas savoir si la situation, envisagée par l'approximation faible de M_C, dans laquelle les deux feux sont verts a été ajoutée (virtuellement) à M_C afin de pouvoir calculer l'approximation, ou si elle fait réellement partie des situations envisagées par M_C.

Vision «restrictive» La troisième approche, enfin, pour l'utilisation d'une base de connaissances acquise par approximation, permet des processus de raisonnement complets, cohérents et efficaces, mais en restreignant les questions qui peuvent être posées à la base de connaissances approximative. Cette vision a été développée principalement par Kautz, Kearns et Selman [KKS95, SK96] et Khardon et Roth [KR96], mais elle a été également beaucoup étudiée par del Val [dV95, dV96]. Elle fera l'objet du chapitre 12, mais à titre d'exemple rappelons que si M est un ensemble d'affectations, ϕ_{lub} est une HORN-LUB de M et α est une formule FNC *de Horn*, alors toutes les affectations de M satisfont α si et seulement si ϕ_{lub} implique α [dV95]. Si la base de connaissances n'est destinée qu'à répondre à des requêtes déductives de Horn, on peut donc y intégrer seulement une HORN-LUB de M sans réelle perte d'information.

Reprenons alors l'exemple de la thèse, et la HORN-LUB ϕ_t''' de M_t exhibée dans le paragraphe 8.1.1. La requête $(x_{me} \vee \neg x_{qu} \vee \neg x_{ch})$ («Si le document est de qualité et si le lecteur est chercheur en informatique, la lecture est mémorable») est impliquée par ϕ_t''', et comme c'est une formule FNC de Horn (réduite à une clause), on peut en déduire que toutes les affectations de M_t la satisfont. En revanche, la requête $(x_{qu} \vee x_{ba})$ («Soit le document est de qualité, soit le travail est bâclé», autrement dit «Si le travail n'est pas bâclé, le document est de qualité»), qui n'est pas en FNC de Horn, n'est pas impliquée par ϕ_t''' alors que toutes les affectations de M_t la satisfont.

Bonnes approximations Bien évidemment, rien n'empêche de combiner ces deux dernières visions de l'approximation, par exemple en intégrant à une base de connaissances une HORN-approximation faible minimale et une HORN-approximation forte maximale de l'ensemble d'exemples de départ, et en interprétant les réponses aux requêtes obtenues avec ces approximations

de façon différente suivant que la requête est en FNC de Horn ou non. Rien n'empêche non plus de privilégier, lorsqu'aucune des deux denières approches ne fournit une réponse, l'une des deux approximations et de lui faire «confiance», comme dans la première approche.

Néanmoins, dans tous les cas les approximations ne peuvent être utiles que si elles satisfont certains critères. En effet, pour que l'approximation d'un ensemble d'exemples soit plus intéressant que sa description, le raisonnement doit être plus facile avec une approximation qu'avec une description. Nous nous intéressons donc aux classes de formules présentées dans la partie I. Mais ces approximations doivent également être de taille comparable ou inférieure à celle de la connaissance initiale (l'ensemble d'affectations), sans quoi, d'une part l'espace nécessaire à leur stockage serait prohibitif, et d'autre part le processus de raisonnement ne serait finalement pas plus efficace. Ces deux points sont les qualités requises pour la *compilation* [CD97, Lib98, DM02]. Enfin, si l'approximation dans une classe de formules est vue comme un processus de compilation de connaissances, on peut estimer que la complexité de la compilation, c'est-à-dire du calcul des approximations, est peu importante, car elle est supposée menée «hors-ligne» (voir la thèse de Liberatore [Lib98]) ; il est cependant bien sûr important de développer des algorithmes de complexité raisonnable pour ces calculs, afin que la compilation soit au moins possible. C'est donc à tous ces aspects que nous nous intéressons dans ce chapitre.

8.1.3 D'autres approches pour approximer

Notons que d'autres notions d'approximations sont proposées dans la littérature pour des bases de connaissances représentées en logique propositionnelle. Mentionnons ainsi la notion d'approximation d'une *fonction booléenne partiellement définie* (*pdBf*), qui donne lieu aux mêmes problématiques que la notion qui nous intéresse, mais où il s'agit d'approximer un ensemble d'exemples *et un ensemble de contre-exemples* donnés ; tous les exemples, mais aucun contre-exemple ne doivent satisfaire la formule calculée. Il est alors intéressant d'étudier le calcul de l'approximation la plus proche de l'ensemble d'exemples, le calcul de l'approximation la plus lointaine, et ainsi de suite. Pour plus de détails nous renvoyons le lecteur au travail de Makino *et al.* [MHI99].

Une autre approche de l'approximation est l'approche par *modèles caractéristiques* [KKS95, KR96, EIM98a] : cette notion coïncide avec celle d'approximation faible minimale dans certains cas (comme le cas HORN), mais pas dans le cas général (formules affines par exemple) ; il s'agit là de calculer, étant donné un ensemble d'exemples, non plus une formule mais un ensemble d'affectations qui, vis-à-vis d'une certaine *base*, caractérise *en intension* l'ensemble des modèles de l'approximation calculée. Ces approximations sont surtout utiles lorsque l'on veut représenter les approximations de façon plus compacte que par des formules, et lorsque l'on restreint les requêtes posées à la base de connaissances à celles correspondant à la base vis-à-vis de laquelle est calculée l'approximation.

Enfin, mentionnons l'approche *disjonctive* [DS94, BGM+97], qui consiste à approximer où à décrire un ensemble d'exemples par une disjonction de formules d'une certaine classe $\mathcal{C}$. Si le nombre de formules participant à la disjonction est raisonnable, on peut alors ramener les requêtes posées à la base de connaissances à cette disjonction, et donc, dans le cas de la déduction par exemple, à un ensemble de requêtes posées à des formules de $\mathcal{C}$.

8.1.4 Travaux précédents

Origine des notions Les notions d'approximations faibles et fortes ont été formalisées par Selman et Kautz dans leur article fondateur [SK96], dans lequel ils motivent principalement

ces notions par l'utilisation «méfiante» des approximations pour le raisonnement (l'article traite principalement l'approximation dans la classe HORN). Les principaux résultats «évidents» y sont présentés et discutés. Notons qu'ils envisagent la notion d'approximation d'une fonction booléenne (avec nos notations, de $\{0,1\}^V$ dans $\{0,1\}$) par une autre, indépendamment de la représentation de ces fonctions, mais qu'en ce qui concerne les calculs, ils discutent principalement l'approximation d'une *formule FNC* par une autre, en donnant deux algorithmes pour calculer des approximations faibles et fortes.

La notion d'approximation *faible minimale* d'un ensemble d'affectations, cependant, est déjà présentée dans l'article antérieur de Dechter et Pearl [DP92] qui formalise les problèmes de description et d'identification. En effet, la notion d'*identifiabilité forte* d'une classe de formules $\mathcal{C}$ y est introduite ; elle formalise la possibilité de calculer efficacement une $\mathcal{C}$-approximation faible minimale (sous la forme d'une formule) d'un ensemble d'affectations M donné, et de décider efficacement si cette approximation *décrit*, ou approxime seulement, M. Cette notion est également présente dans la notion de *modèles caractéristiques* [KKS95, Kha95, KR96], comme nous l'avons déjà vu. Dans tous les cas, l'approximation est principalement vue comme un processus de compilation de connaissances.

Classes de formules Ce sont principalement Dechter et Pearl [DP92] et Kavvadias, Papadimitriou et Sideri [KPS93] qui se sont intéressés à l'approximation d'*ensembles d'affectations* par des formules (ce qui est notre cadre). Les premiers montrent que la HORN-LUB d'un ensemble d'affectations M donné est unique à équivalence logique près, mais que sa taille peut être exponentielle en la taille de M. Ils montrent la même propriété d'unicité «sémantique» pour la BIJ-LUB de M, mais montrent également qu'une telle formule peut être calculée efficacement. Les seconds, quant à eux, traitent exclusivement le cas de la classe HORN ; ils montrent que le problème consitant à calculer une HORN-LUB de M est plus difficile que le problème consistant à engendrer toutes les *transversales minimales* d'un hypergraphe donné (nous reviendrons sur ce point dans le paragraphe 8.2.2). Pour ce qui est des approximations *fortes*, pour lesquelles l'unicité n'est plus de mise, ils montrent qu'une HORN-GLB de M peut être calculée efficacement avec un algorithme glouton. Ils introduisent et motivent également la notion de *max*-HORN-*GLB* de M, c'est-à-dire une HORN-GLB de M avec le nombre maximal de modèles parmi toutes ; ils montrent que le calcul d'une telle formule admet un problème de décision associé NP-complet (la preuve de l'article est fausse, mais nous en redonnons une dans ce chapitre).

D'autres résultats concernant notre cadre peuvent également être trouvés dans la littérature, même s'il ne sont pas formulés dans les mêmes termes. Ainsi, Selman et Kautz [SK96] et del Val [dV95, dV96] donnent des caractérisations sémantiques des approximations, indépendantes par définition des représentations des connaissances approximée et approximante. Les premiers montrent que la HORN-LUB d'un ensemble d'affectations est logiquement équivalente à la conjonction de toutes les clauses de Horn satisfaites par toutes les affectations, et le second généralise cette proposition au cas de toutes les classes de formules définies en nature et stables par subsomption. Selman et Kautz [SK96] donnent également quelques résultats expérimentaux sur la qualité des approximations (en termes du nombre des requêtes auxquelles elles permettent de répondre), et del Val discute également ce point [dV96, paragraphe 6]. Enfin, mentionnons que del Val [dV96] discute la généralisation des notions d'approximation pour des bases de connaissances représentées au premier ordre.

Notre apport Notre contribution à cet état de l'art est la suivante. Pour ce qui est des approximations faibles, nous exhibons des exemples pour lesquels la taille d'une HDP- ou d'une

NÉG-approximation faible minimale est exponentiellement plus grande que la taille de l'ensemble d'affectations approximé, et nous montrons que pour les classes de formules $\mathcal{C}$-renommables il existe en général plusieurs approximations faibles minimales de M non logiquement équivalentes les unes aux autres, dont les nombres de modèles peuvent de plus être exponentiellement différents ; nous discutons des pistes de recherche pour étudier ces approximations. Pour ce qui est des approximations fortes, nous traitons tout d'abord le calcul d'une $\mathcal{C}$-GLB quelconque, et montrons que l'algorithme glouton proposé pour la classe HORN par Kavvadias *et al.* [KPS93] peut être adapté à toutes les classes de formules qui nous intéressent ; si la généralisation est directe pour les classes définies en nature, le résultat est moins trivial pour les classes de formules $\mathcal{C}$-renommables. Enfin, nous traitons le calcul d'une $\mathcal{C}$-GLB d'un ensemble d'affectations donné avec le nombre maximal de modèles. Nous donnons une nouvelle preuve de l'intraitabilité de ce calcul pour la classe HORN (celle de Kavvadias *et al.* étant fausse), et montrons que ce calcul est également intraitable, avec une preuve similaire, pour la classe BIJ ; nous montrons en revanche qu'il peut être effectué en temps subexponentiel pour la classe AFFINE. Ces résultats ont été publiés partiellement [Zan02a, Zan02b].

8.2 Approximations faibles

Nous traitons tout d'abord le calcul de $\mathcal{C}$-approximations *faibles* d'un ensemble d'affectations M donné, où $\mathcal{C}$ est l'une des classes de formules qui nous intéressent pour représenter explicitement des connaissances. Rappelons qu'il s'agit intuitivement d'ajouter (virtuellement) un ensemble minimal d'affectations à M afin d'obtenir un ensemble descriptible dans $\mathcal{C}$.

8.2.1 Caractérisations

Nous verrons dans ce paragraphe que, contrairement aux problèmes de description ou d'identification, les résultats de traitabilité sont très différents d'une classe $\mathcal{C}$ à l'autre pour le calcul d'approximations faibles. Néanmoins, le résultat suivant est commun à toutes les classes définies en nature qui nous intéressent.

Proposition 8.1 (unicité sémantique de la LUB) *Soit $\mathcal{C}$ l'une des classes de formules :*

HORN, ANTI-HORN, HDP, ANTI-HDP, POS, NÉG, BIJ, AFFINE

et soit M un ensemble d'affectations à un ensemble de variables V. Une $\mathcal{C}$-LUB ϕ_{lub} de M existe et est unique à l'équivalence logique près. L'ensemble des affectations à V qui la satisfont est la clôture $Cl_{\mathcal{C}}(M)$ de M par l'opération correspondant à la classe $\mathcal{C}$.

Preuve Pour toutes les classes $\mathcal{C}$ de l'énoncé nous avons vu dans la partie I qu'un ensemble d'affectations M est descriptible dans $\mathcal{C}$ si et seulement si on a $M = Cl_{\mathcal{C}}(M)$. De fait, tout ensemble d'affectations descriptible dans $\mathcal{C}$ et incluant M contient $Cl_{\mathcal{C}}(M)$; c'est donc vrai en particulier pour l'ensemble de modèles de toute $\mathcal{C}$-LUB de M (formée sur V, sans perte de généralité). Mais comme $Cl_{\mathcal{C}}(M)$ est descriptible dans $\mathcal{C}$, il inclut également l'ensemble de modèles d'une $\mathcal{C}$-LUB de M. C'est donc bien l'ensemble de modèles d'une $\mathcal{C}$-LUB de M, et comme il est inclus dans tous les autres il est bien unique. □

Ainsi, on voit que la formule FNC de Horn ϕ_t''' exhibée dans le paragraphe 8.1.1 est bien une HORN-LUB de l'ensemble d'affectations M_t, puisque l'on a $\mathcal{M}(\phi_t''') = Cl_{\text{HORN}}(M_t)$; de plus, toutes les HORN-LUB de M_t sont logiquement équivalentes à ϕ_t'''.

De façon plus générale, remarquons qu'une condition suffisante pour l'unicité sémantique de la $\mathcal{C}$-LUB de M est la stabilité de la classe $\mathcal{C}$ par conjonction. En effet, on voit alors que la conjonction de toutes les $\mathcal{C}$-approximations faibles de M est encore une $\mathcal{C}$-approximation faible de M, et que l'ensemble de ses modèles est inclus dans celui de toutes les autres.

Le cas est différent pour les classes de formules $\mathcal{C}$-renommables : l'existence est toujours garantie, mais pas l'unicité. De plus, dans ce cas plusieurs LUBs non logiquement équivalentes peuvent avoir des nombres exponentiellement différents de modèles.

Proposition 8.2 (HORN-REN-, HDP-REN-, MONOT-LUB) *Si $\mathcal{C}$ est l'une des classes de formules :*

HORN-REN, HDP-REN, MONOT

alors il existe une famille d'ensembles d'affectations $(M_n)_{n\in\mathbb{N}}$ et deux familles de formules de $\mathcal{C}$, $(\phi_n)_{n\in\mathbb{N}}$ et $(\phi'_n)_{n\in\mathbb{N}}$, telles que pour tout $n \in \mathbb{N}$, ϕ_n et ϕ'_n soient deux $\mathcal{C}$-LUBs de M_n non logiquement équivalentes et que le nombre de modèles de ϕ_n, exprimé en fonction de n, soit exponentiellement plus grand que celui de ϕ'_n.

Preuve Nous considérons le même ensemble d'affectations M_n pour les trois cas. Pour $n \in \mathbb{N}, n \geq 4$ (cette dernière hypothèse ne changeant pas le résultat), nous définissons M_n comme l'ensemble de toutes les affectations à $V_n = \{x_1, x_2, \ldots, x_n\}$ affectant 0 à exactement deux variables :

$$M_n = \{0011\ldots111, 0101\ldots111, \ldots, 111\ldots100\}$$

Pour $i, j \in \{1, 2, \ldots, n\}, i < j$, nous notons $m_{i,j}$ l'affectation qui affecte 0 à x_i et x_j, et 1 à x_k pour tout $k \in \{1, 2, \ldots, n\}, k \neq i, j$. Nous définissons alors ϕ_n et ϕ'_n par leurs ensembles de modèles M_{ϕ_n} et $M_{\phi'_n}$ (il suffit ensuite de considérer des descriptions dans $\mathcal{C}$ de ces ensembles).

[cas HORN-*renommable]* Pour $\mathcal{C}$ = HORN-REN, nous posons :

$$M_{\phi_n} = Cl_{\text{HORN}}(M_n)$$

Cet ensemble est par construction l'ensemble de modèles d'une HORN-REN-UB de M_n, et contient toutes les affectations de $\{0,1\}^{V_n}$ hormis celles qui affectent 0 à une seule, ou à aucune variable. Nous montrons que M_{ϕ_n} est l'ensemble de modèles d'une HORN-REN-*L*UB de M_n.

Pour cela, nous montrons que l'ensemble de modèles d'aucune autre HORN-REN-LUB de M_n n'est strictement inclus dans M_{ϕ_n}. Soit donc M'_n l'ensemble de modèles d'une HORN-REN-LUB de M_n. Alors il existe un renommage ρ de V_n tel que $(M'_n)_{\oplus\rho}$ est descriptible dans HORN, c'est-à-dire clos par l'opérateur $\wedge$ (proposition 2.2 page 32). Les variables jouant des rôles symétriques dans M_n, on peut supposer sans perte de généralité que ρ renomme exactement les variables $x_1, x_2, \ldots, x_k$ pour un entier $k \in \{0, \ldots, n\}$. Si $k = 0$, tout d'abord, alors on a $M'_n = M_n$. Si $k = 1$, nous montrons $M_{\phi_n} \subseteq M'_n$ (on a même l'égalité, mais cela n'apporte rien à la preuve) ; en effet, si m est une affectation de M_{ϕ_n} nous avons vu que m affecte 0 à au moins deux variables de V_n ; de fait, si m affecte 0 à x_1, alors $m_{\oplus\rho}$ est égal au produit :

$$\bigwedge_{j\in\{2,3,\ldots,n\}, m[j]=0} (m_{1,j})_{\oplus\rho}$$

qui est dans $(M'_n)_{\oplus\rho}$ puisque ce dernier inclut nécessairement $Cl_{\text{HORN}}((M_n)_{\oplus\rho})$ et que le produit n'est pas vide (m affecte 0 à au moins deux variables) ; si m affecte 1 à x_1, alors $m_{\oplus\rho}$ est égal au produit :

$$\bigwedge_{i,j\in\{2,3,\ldots,n\}, m[i]=m[j]=0} (m_{i,j})_{\oplus\rho}$$

qui est dans $(M'_n)_{\oplus\rho}$ pour les mêmes raisons. Donc on a $(M_n)_{\oplus\rho} \subseteq (M'_n)_{\oplus\rho}$, et donc $M_n \subseteq M'_n$.

Dans le cas général ($k \geq 2$), nous montrons que M'_n contient au moins une affectation que M_n ne contient pas. En effet, on voit qu'alors l'affectation :

$$m = (m_{1,n})_{\oplus\rho} \wedge (m_{2,n})_{\oplus\rho}$$

qui est par construction dans $(M'_n)_{\oplus\rho}$, affecte 0 à la variable $x_i \in V_n$ si et seulement si on a $i \leq k$ ou $i = n$, et de fait on a $m_{\oplus\rho} = \overline{1_{V_{n-1}}}0$ (l'affectation de 1 à toutes les variables de V_n, sauf à x_n), d'où on déduit $\overline{1_{V_{n-1}}}0 \in M'_n$; or, nous avons vu que toute affectation de M_{ϕ_n} affectait 0 à au moins deux variables, d'où $M'_n \not\subseteq M_{\phi_n}$.

Finalement, M_{ϕ_n} est bien l'ensemble de modèles d'une HORN-REN-LUB de M_n. Mais par construction, l'ensemble d'affectations $M_{\phi'_n}$ défini par :

$$M'_n = Cl_{\text{HORN}}((M_n)_{\oplus\overline{1_{V_n}}})_{\oplus\overline{1_{V_n}}}$$

est également l'ensemble de modèles d'une HORN-REN-UB de M_n ; or M_{ϕ_n} n'est pas inclus dans M'_n, puisque pour $n \geq 4$, l'affectation m avec $m[x_1] = 1$ et $\forall x \in V_n, x \neq x_1 \Longrightarrow m[x] = 0$ est dans M_{ϕ_n} mais pas dans M'_n ; donc il existe un ensemble d'affectations $M_{\phi'_n} \subseteq M'_n, M_{\phi'_n} \neq M_n$, tel que $M_{\phi'_n}$ est l'ensemble de modèles d'une HORN-REN-LUB de M_n. Or, il est facile de voir que l'on a $|M_{\phi_n}| = 2^n - n - 1$ mais $|M_{\phi'_n}| \leq \frac{n(n-1)}{2} + n + 1$.

[cas HDP*-renommable et monotone]* Pour le cas des HDP-REN-LUB, il suffit de considérer les ensembles $M_{\phi_n} \cup \{\overline{1_{V_n}}\}$ et $M_{\phi'_n} \cup \{\overline{0_{V_n}}\}$ (voir la proposition 2.5 du chapitre 2) ; les affectations $\overline{1_{V_n}}$ et $\overline{0_{V_n}}$ n'étant pas impliquées dans la preuve pour les HORN-REN-LUBs, et la classe HDP-REN étant incluse dans la classe HORN-REN, on a le résultat. Enfin, pour le cas monotone il suffit de remarquer que les ensembles M_{ϕ_n} et $M_{\phi'_n}$ du cas HORN-renommable sont monotones (avec les mêmes renommages dans NÉG que dans HORN), et encore une fois de conclure avec l'inclusion MONOT $\subset$ HORN-REN. □

8.2.2 Calcul d'une approximation faible

Nous nous intéressons maintenant au *calcul* d'une $\mathcal{C}$-approximation faible minimale d'une relation donnée. Formellement, le problème (avec sortie) que nous traitons est donc le suivant. Soit $\mathcal{C}$ une classe de formules.

Problème 8.1 (LUB[$\mathcal{C}$])

Donnée : *Un ensemble d'affectations M*
Sortie : *Une $\mathcal{C}$-approximation faible minimale de M.*

Approximations faibles minimales de taille exponentielle Comme nous l'avons dit, Kautz, Kearns et Selman [KKS95] montrent (dans un but différent du nôtre) que la taille (nombre d'occurrences de littéraux) de la plus petite HORN-LUB d'un ensemble d'affectations M à un ensemble de variables V est potentiellement exponentiellement plus grande que la taille $|M||V|$ de M. Nous rappelons ce résultat ci-dessous, et montrons qu'il vaut également pour les classes

HDP et NÉG. Bien entendu, si l'on considère les exemples duaux (obtenus de ceux-ci en renommant toutes les variables), on voit qu'ils montrent le même résultat pour les classes duales ANTI-HORN, ANTI-HDP et POS. De fait, nous considérons que les problèmes :

LUB[HORN], LUB[ANTI-HORN], LUB[HDP], LUB[ANTI-HDP], LUB[NÉG], LUB[POS]

sont intraitables (voir la discussion des préliminaires généraux, paragraphe 2.3 sur l'intraitabilité des problèmes avec sortie). De plus, ce résultat montre qu'il n'est pas nécessairement avantageux de stocker systématiquement dans une base de connaissances de telles approximations plutôt que l'ensemble d'affectations de départ : d'une part l'espace nécessaire est potentiellement prohibitif, et d'autre part les algorithmes de raisonnement ont alors à traiter des données de grande taille.

Proposition 8.3 (LUB exponentielle [KKS95, théorème 6]) *Soit $\mathcal{C}$ l'une des classes de formules :*

HORN, ANTI-HORN, NÉG, POS, HDP, ANTI-HDP

Alors il existe une famille $(M_n)_{n\in\mathbb{N}}$ d'ensembles d'affectations et une famille $(\phi_n)_{n\in\mathbb{N}}$ de formules de $\mathcal{C}$ telles que pour tout $n \in \mathbb{N}$, ϕ_n est une $\mathcal{C}$-LUB de M_n, il n'existe aucune formule de $\mathcal{C}$ logiquement équivalente à ϕ_n et de taille inférieure, et la taille de ϕ_n, exprimée en fonction de n, est exponentiellement plus grande que la taille de M_n.

Preuve Les auteurs traitent le cas Horn, et considèrent pour un $n \in \mathbb{N}$ l'ensemble de variables $V_n = \{x_i, x'_i \mid i \in \{1, 2, \ldots, n\}\}$ et l'ensemble d'affectations M_n qui contient, pour tout $i \in \{1, 2, \ldots, n\}$, toutes les affectations m à V_n avec $m[x_i] = m[x'_i] = 0$ et $\exists j \in \{1, 2, \ldots, n\}, j \neq i$ tel que $\forall k \in \{1, 2, \ldots, n\}, k \neq i, j \Longrightarrow m[x_k] = m[x'_k] = 1$ et, soit $m[x_j] = 1$, soit $m[x'_j] = 1$ (soit les deux). Si on suppose que l'ordre sur les variables est $x_1 < x'_1 < x_2 < x'_2 < \cdots < x_n < x'_n$, on a donc :

$$
\begin{aligned}
M_n \;=\;& \{00111111\ldots11, 00011111\ldots11, 00101111\ldots11\} \\
\cup\;& \{00111111\ldots11, 00110111\ldots11, 00111011\ldots11\} \\
\cup\;& \cdots \\
\cup\;& \{001111\ldots1111, 001111\ldots1101, 001111\ldots1110\} \\
\\
\cup\;& \{11001111\ldots11, 01001111\ldots11, 10001111\ldots11\} \\
\cup\;& \{11001111\ldots11, 11000111\ldots11, 11001011\ldots11\} \\
\cup\;& \cdots \\
\cup\;& \{110011\ldots1111, 110011\ldots1101, 110011\ldots1110\} \\
\\
\cup\;& \cdots \\
\\
\cup\;& \{111111\ldots1100, 011111\ldots1100, 101111\ldots1100\} \\
\cup\;& \{111111\ldots1100, 110111\ldots1100, 111011\ldots1100\} \\
\cup\;& \cdots \\
\cup\;& \{1111\ldots111100, 1111\ldots110100, 1111\ldots111000\}
\end{aligned}
$$

Les auteurs montrent alors que la formule FNC de Horn :

$$\phi_n = \bigwedge_{\forall i\in\{1,2,\ldots,n\}, y_i\in\{x_i,x'_i\}} (\neg y_1 \vee \neg y_2 \vee \cdots \vee \neg y_n)$$

est une HORN-LUB de M_n et qu'il n'existe pas de formule FNC logiquement équivalente et plus petite. Or, la taille de M_n est $|M_n||V_n| \leq 6n^2(n-1)$, et celle de ϕ est $n2^n$.

Le résultat dual vaut pour la classe ANTI-HORN. Mais remarquons également que pour tout $n \in \mathbb{N}$ la formule ϕ_n est en FNC négative ; c'est donc une NÉG-UB de M_n, mais comme c'est une HORN-LUB de M_n c'est *a fortiori* une NÉG-*L*UB de M_n. A nouveau le résultat dual vaut pour la classe POS.

Pour la classe HDP, remarquons que si l'énoncé n'était pas vrai, alors pour tout ensemble d'affectations M à un ensemble de variables V il existerait une HDP-LUB ϕ de taille non exponentiellement plus grande que celle de M. On pourrait alors construire une HORN-LUB ϕ' de M, de taille non exponentiellement plus grande que celle de M, en posant $\phi' = \phi$ si $\overline{1_V} \in M$, et $\phi' = \phi \wedge \bigvee_{x \in V} \neg x$ sinon ; ceci contredit la première partie de l'énoncé. Bien entendu, le résultat dual est valable pour la classe ANTI-HDP. □

Transversales d'un hypergraphe On peut cependant exprimer plus finement la complexité de ces problèmes, même si nous les considérons comme intraitables. Ces problèmes sont en effet étroitement liés au problème du calcul des *transversales minimales d'un hypergraphe* [EG95b]. Ce problème peut être formulé comme le problème de *dualisation d'une formule négative*, c'est-à-dire du calcul d'une formule FNC logiquement équivalente à une formule FND négative donnée (voir l'article de Domingo *et al.* [DMP99] par exemple). Eiter et Gottlob [EG95b] discutent de nombreux problèmes algorithmiques liés à ce calcul, et nous renvoyons à leur article pour les définitions. Pour ce problème, dont la sortie (nombre de clauses de la formule FNC) peut être exponentiellement plus grande que l'entrée, le meilleur algorithme connu est de complexité subexponentielle en la taille de la sortie [FK96, KS03] (voir l'article de Johnson *et al.* [JYP88] pour les notions de complexité en fonction de la taille de la sortie). Kavvadias *et al.* [KPS93] montrent que si l'on peut calculer une HORN-LUB d'un ensemble d'affectations donné en temps polynomial *en la taille de la plus petite telle LUB*, alors on peut également calculer les transversales minimales d'un hypergraphe donné en temps polynomial (toujours en la taille de la sortie) [KPS93, théorème 1], et de même si on demande que les algorithmes *énumèrent* les clauses de la HORN-LUB (resp. les transversales de l'hypergraphe) en temps polynomial. Le calcul d'une HORN-LUB est donc plus difficile que le calcul des transversales. Les auteurs laissent la réciproque ouverte, mais conjecturent qu'elle est vraie, et donc que les problèmes sont équivalents. Khardon [Kha95] complète ces travaux en montrant que ce problème est équivalent au problème de décision associé, consistant à décider si une formule de Horn donnée est ou non une HORN-LUB d'un ensemble d'affectations donné, et en donnant d'autres résultats reliés.

Bien entendu, il en va de même pour la classe HDP (voir la preuve de la proposition 8.3). Enfin, le calcul d'une NÉG-LUB est, lui, *équivalent* au problème des transversales d'hypergraphe, puisqu'étant donné un ensemble d'affectations M à un ensemble de variables V, la formule *FND* :

$$\bigvee_{m \in M, \forall m' \in M, m' \preceq m} \left(\bigwedge_{x \in V, m[x]=0} \neg x \right)$$

est logiquement équivalente à toute NÉG-LUB de M.

Enfin, citons l'algorithme probabiliste de Kautz *et al.* [KKS95, théorème 15], qui calcule une HORN-UB d'un ensemble d'affectations qui en est avec forte probabilité une bonne approximation (en termes de nombres de modèles) ; cet algorithme est basé sur l'algorithme d'Angluin *et al.* [AFP92] pour *PAC-apprendre* une formule de Horn (voir chapitre 9).

Pour plus de détails sur le problème des transversales d'hypergraphe, nous renvoyons le lecteur au récent article de Kavvadias et Stavropoulos [KS03].

Cas polynomiaux En fait, parmi les classes de formules qui nous intéressent, seules deux sont traitables pour le calcul d'une approximation faible minimale : les classes BIJ et AFFINE. Le cas bijonctif étant traité dans [DP92, KS98] et le cas affine dans [KS98], nous rappelons simplement les résultats et l'idée des algorithmes. Le cas bijonctif tire avantage du nombre quadratique (en $|V|$) de 2clauses formées sur un ensemble de variables V, et le cas affine de la caractérisation d'une approximation faible minimale par une base.

Proposition 8.4 (LUB[BIJ]) *Soit M un ensemble d'affectations à un même ensemble de variables V. On peut calculer une* BIJ*-LUB de M en temps $O(|M||V|^2)$, et la formule obtenue contient moins de $\frac{3}{2}|V|^2$ clauses.*

Idée de la preuve Nous rappelons simplement la construction, et renvoyons à [DP92] pour la preuve que la formule ϕ construite est bien une BIJ-LUB de M. Considérons la formule ϕ définie comme la conjonction de toutes les 2clauses C sur V qui vérifient $\forall m \in M, m \models C$. On peut calculer cette formule en temps $O(|M||V|^2)$; en effet, il existe $O(|V|^2)$ 2clauses sur V, que l'on peut facilement énumérer, et pour chacune il suffit de décider si $\forall m \in M, m \models C$, en temps $O(|M|)$ pour chacune. Si M est vide, on peut de plus poser directement $\phi = ((\,))$. Dans le cas contraire, pour une paire $\{x, x'\}$ de variables de V, ϕ ne peut contenir les quatre clauses de cardinal 2 formées sur $\{x, x'\}$ (sinon elle serait insatisfaisable), donc elle en contient au plus trois ; de même, pour une variable donnée $x \in V$, ϕ ne peut contenir les deux clauses unitaires (x) et $(\neg x)$, et ϕ ne peut contenir la clause vide $()$. Donc ϕ contient au plus $\frac{3}{2}|V|(|V|-1)+|V|$ clauses, soit moins de $\frac{3}{2}|V|^2$. □

Considérons par exemple l'ensemble d'affectations M_t du petit monde de la thèse :

$$M_t = \{00111, 01000, 10100, 11011\}$$

sur l'ensemble de variables $\{x_{me}, x_{qu}, x_{ba}, x_{in}, x_{ch}\}$. On voit que les seules restrictions de M_t à un couple de variables différentes de $\{00, 01, 10, 11\}$ sont ses restrictions aux couples (x_{qu}, x_{ba}) et (x_{in}, x_{ch}), qui sont $\{01, 10\}$ et $\{00, 11\}$, respectivement. De fait, la formule 2FNC :

$$(x_{qu} \vee x_{ba}) \wedge (\neg x_{qu} \vee \neg x_{ba}) \wedge (x_{in} \vee \neg x_{ch}) \wedge (\neg x_{in} \vee x_{ch})$$

est une BIJ-LUB de M_t.

Proposition 8.5 (LUB[AFFINE]) *Soit M un ensemble d'affectations à un même ensemble de variables V. On peut calculer une* AFFINE*-LUB de M en temps $O(|M||V|^3)$.*

Preuve Nous supposons $\overline{0_V} \in M$, sans perte de généralité (voir la proposition 3.4 page 46 et la discussion qui suit), afin de nous ramener au cadre des espaces vectoriels. Nous cherchons donc une description de l'*espace vectoriel* $Cl_{\text{AFFINE}}(M)$. Par définition de l'AFFINE-clôture, tout ensemble linéairement indépendant d'affectations de M maximal pour l'inclusion est une base de cet espace vectoriel. Nous cherchons donc à calculer un tel ensemble ; or, ce calcul peut être effectué de façon gloutonne, en testant pour chaque affectation candidate si elle est linéairement indépendante de l'ensemble B courant, en temps $O(|B|^2|V|)$ avec le pivot de Gauss. On peut donc calculer une base en temps $O(|M||V|^3)$, puisque l'on a $|B| \leq |V|$, et il suffit ensuite de calculer une description affine de l'espace admettant cette base, en temps $O(|V|^3)$ (proposition 3.5 du chapitre 3). □

Ainsi, pour l'ensemble d'affectations M_C de l'exemple du carrefour :

$$\{0000, 0010, 0101, 1000, 1010, 1101\}$$

qui contient déjà l'affectation $\overline{0_{V_C}}$, on peut calculer la base $\{0010, 0101, 1000\}$, puis calculer la formule affine :

$$(x_{VP} \oplus x_{DT} = 0)$$

qui, comme nous l'avons vu dans le paragraphe 8.1.1, est bien une AFFINE-LUB de M_C.

Pour ce qui est des classes des formules $\mathcal{C}$-renommables, puisque en général il n'y a pas unicité sémantique de la $\mathcal{C}$-LUB, il serait pertinent de s'intéresser au calcul d'une $\mathcal{C}$-LUB quelconque et à celui d'une $\mathcal{C}$-LUB *avec le nombre minimal de modèles parmi toutes les $\mathcal{C}$-LUBs*. Nous ne sommes pas en mesure de donner des résultats complets concernant ces problématiques, mais nous pouvons formuler plusieurs remarques à propos de la classe des formules $\mathcal{C}$-renommables, pour une certaine classe de formules $\mathcal{C}$.

De façon générale, on peut lier les approximations dans cette classe à celles dans la classe $\mathcal{C}$. En effet, on voit aisément qu'étant donnée une classe de formules $\mathcal{C}$ et un ensemble d'affectations M, une formule ϕ est une approximation faible de M dans la classe des formules $\mathcal{C}$-renommables, renommable dans $\mathcal{C}$ par un renommage ρ, si et seulement si la formule $\phi_{\oplus\rho}$ est une $\mathcal{C}$-approximation faible de l'ensemble $M_{\oplus\rho}$; en conséquence, si ϕ est une approximation faible *minimale* de M dans la classe des formules $\mathcal{C}$-renommables, alors $\phi_{\oplus\rho}$ est une $\mathcal{C}$-approximation faible minimale de $M_{\oplus\rho}$. On peut ainsi ramener le calcul d'une LUB de M dans la classe des formules $\mathcal{C}$-renommables à celui d'une $\mathcal{C}$-LUB de certains ensembles d'affectations de la forme $M_{\oplus\rho}$. Malheureusement, nous avons vu que ce calcul est intraitable pour les classes $\mathcal{C}$ dans lesquelles la renommabilité a un réel sens (HORN, HDP et NÉG) ; on ne peut donc pas effectuer une recherche exhaustive.

8.3 Approximations fortes

Nous nous intéressons maintenant aux $\mathcal{C}$-approximations *fortes* maximales d'une relation. Contrairement au cas des approximations faibles, l'unicité de la $\mathcal{C}$-GLB, pour laquelle une condition suffisante est la stabilité de la classe $\mathcal{C}$ par disjonction, n'est garantie que pour les classes NÉG et POS. Pour toutes les autres classes $\mathcal{C}$, on peut même trouver des familles de relations avec des $\mathcal{C}$-GLBs différentes et possédant des nombres de modèles exponentiellement différents, comme le montre la proposition suivante.

Proposition 8.6 (GLBs multiples) *Soit $\mathcal{C}$ l'une des classes de formules :*

HORN, ANTI-HORN, HDP, ANTI-HDP, BIJ, AFFINE, HORN-REN, HDP-REN, MONOT

Il existe une famille d'ensembles d'affectations $(M_n)_{n\in\mathbb{N}}$ et deux familles de formules de $\mathcal{C}$, $(\phi_n)_{n\in\mathbb{N}}$ et $(\phi'_n)_{n\in\mathbb{N}}$, telles que pour tout $n \in \mathbb{N}$, ϕ_n et ϕ'_n sont deux $\mathcal{C}$-GLBs de M_n non logiquement équivalentes, et le nombre de modèles de ϕ_n, exprimé en fonction de n, est exponentiellement plus grand que celui de ϕ_n.

Preuve La construction est similaire pour toutes les classes, et nous détaillons donc principalement le cas Horn.

[Cas HORN et ANTI-HORN] Considérons tout d'abord la classe HORN. Pour un $n \in \mathbb{N}$ nous considérons l'ensemble de variables $V_n = \{x_1, x_2, \ldots, x_n\}$, et définissons l'ensemble d'affectations M_n à V_n $(n \geq 2)$ par :

$$M_n = \{100\ldots0\} \cup \{m \in \{0,1\}^{V_n} \mid m[x_1] = 0 \text{ et } m[x_2] = 1\}$$

Alors il est facile de voir que l'ensemble d'affectations $\{100\ldots0\}$ à V_n est l'ensemble de modèles d'une HORN-GLB de M_n, puisque la conjonction ($\wedge$) de son unique élément avec toute autre affectation de M_n est l'affectation $\overline{0_{V_n}}$, qui n'est pas dans M_n ; donc il est bien maximal. Pour la même raison, on voit que l'ensemble d'affectations $\{m \in \{0,1\}^{V_n} \mid m[x_1] = 0 \text{ et } m[x_2] = 1\}$ (de cardinal 2^{n-2}) est l'ensemble de modèles d'une HORN-GLB de M_n. On a alors le résultat en considérant des descriptions ϕ_n et ϕ'_n de ces ensembles dans HORN.

Le résultat pour la classe ANTI-HORN est dual.

[Cas HDP *et* ANTI-HDP*]* Pour la classe HDP, il suffit de considérer l'ensemble $M_n \cup \overline{1_V}$, et d'ajouter de même l'affectation $\overline{1_V}$ aux ensembles de modèles de ϕ_n et de ϕ'_n considérés pour le cas HORN. Pour la classe ANTI-HDP le résultat est dual.

[Cas BIJ *et* AFFINE*]* Pour la classe BIJ, étant donné $n \in \mathbb{N}$ ($n \geq 3$) nous considérons l'ensemble :

$$M_n = \{01000\ldots0, 00100\ldots0\} \cup \{m \in \{0,1\}^{V_n} \mid m[x_1] = 1 \text{ et } m[x_2] = m[x_3] = 0\}$$

Alors l'ensemble $\{01000\ldots0, 00100\ldots0\}$ est l'ensemble de modèles d'une BIJ-LUB de M_n car il ne peut être augmenté par une affectation de M_n, et de même pour l'ensemble :

$$\{m \in \{0,1\}^{V_n} \mid m[x_1] = 1 \text{ et } m[x_2] = m[x_3] = 0\} \cup \{01000\ldots0\}$$

Le même M_n montre également le résultat pour les AFFINE-GLBs, puisque les ensembles $\{m \in \{0,1\}^{V_n} \mid m[x_1] = 1 \text{ et } m[x_2] = m[x_3] = 0\}$ et $\{01000\ldots0, 00100\ldots0\}$ sont les ensembles de modèles de deux AFFINE-GLBs de M_n.

[Cas renommables] Pour la classe HORN-REN, étant donné $n \in \mathbb{N}$ ($n \geq 3$) nous considérons encore l'ensemble M_n des cas BIJ et AFFINE, et les mêmes ensembles de modèles de GLBs que pour la classe BIJ. Quel que soit le renommage choisi, on voit que l'ensemble $\{01000\ldots0, 00100\ldots0\}$ ne peut pas être augmenté en une HORN-REN-LB de M_n en considérant une troisième affectation quelconque de M_n ; les variables $x_4, x_5, \ldots, x_n$ n'interviennent pas, et il suffit de considérer les différents renommages de x_1, x_2, x_3. La «grande» GLB étant renommable dans HORN par $10111\ldots1$ et la «petite» par $10111\ldots1$, on a donc le résultat.

Pour la classe HDP-REN la preuve est la même, en considérant les mêmes ensembles et les mêmes renommages.

Enfin, pour la classe MONOT on peut considérer la même «grande» GLB que pour AFFINE (renommable dans NÉG par $01000\ldots0$), et il suffit de remarquer que l'ensemble réduit à un élément $\{01000\ldots0\}$ est lui aussi maximal. □

Pour cette raison, étant donnée une classe de formules $\mathcal{C}$ et un ensemble d'affectations M, deux problématiques sont pertinentes : le calcul d'une $\mathcal{C}$-GLB quelconque de M et le calcul d'une $\mathcal{C}$-GLB de M *avec le nombre maximal de modèles parmi toutes*. La première, plus facile algorithmiquement, trouve sa place lorsque l'on cherche à préserver certaines propriétés sémantiques (par exemple, la satisfaisabilité, voir le chapitre 12), sans pour autant avoir besoin d'une approximation très *proche* de l'ensemble d'exemples ; la seconde consiste en revanche à chercher une approximation «maximalement proche» de l'ensemble d'exemples, but naturel lorsque, par exemple, on compte utiliser la base de connaissances approximative en lui faisant confiance (voir paragraphe 8.1.2).

Nous traitons maintenant ces deux problématiques, en complétant l'état de l'art. Nous ne traitons la classe NÉG et sa classe duale POS que dans le premier paragraphe, puisque la NÉG-GLB et la POS-GLB d'un ensemble d'affectations donné sont uniques, à l'équivalence logique près.

Entrée : Un ensemble d'affectations M
Sortie : Une $\mathcal{C}$-GLB de M.

Début
  `modeles` $\leftarrow \emptyset$;
  Pour tout $m \in M \backslash$`modeles` **Faire**
    `tmp-modeles` $\leftarrow Cl_{\mathcal{C}}($`modeles` $\cup \{m\})$;
    Si `tmp-modeles` $\subseteq M$ **Alors** `modeles` $\leftarrow$ `tmp-modeles` **Fin Si** ;
  Fin Pour ;
  retourner une description de `modeles` dans $\mathcal{C}$;
Fin

FIG. 8.1 – Algorithme `GLB[`$\mathcal{C}$`]` (modèle)

8.3.1 Calcul d'une approximation forte maximale quelconque

Nous nous intéressons tout d'abord au calcul d'une $\mathcal{C}$-approximation forte maximale *quelconque* d'un ensemble d'affectations donné. Formellement, nous traitons le problème, toujours avec sortie, suivant.

Problème 8.2 (`GLB[`$\mathcal{C}$`]`)

Donnée : *Un ensemble d'affectations M*
Sortie : *Une $\mathcal{C}$-approximation forte maximale de M.*

Nous montrons dans ce paragraphe que ce calcul s'unifie pour toutes les classes qui nous intéressent en un algorithme glouton, qui généralise celui proposé pour la classe HORN par Kavvadias *et al.* [KPS93] ; seule une étape supplémentaire devra être ajoutée pour les classes de formules $\mathcal{C}$-renommables. L'idée est de calculer tout d'abord l'*ensemble de modèles* d'une $\mathcal{C}$-GLB de l'ensemble d'affectations M donné, puis de décrire cet ensemble dans $\mathcal{C}$. On tire donc avantage, du point de vue de la complexité, d'une part du fait que l'ensemble de modèles calculé est par définition inclus, et en particulier plus petit, que la donnée M du problème, et d'autre part du fait que l'on peut décrire efficacement un ensemble d'affectations dans toutes les classes de formules qui nous intéressent ; de plus, les résultats présentés au chapitre 7 montrent que la formule calculée sera toujours de taille au plus polynomiale en la taille de M.

L'algorithme que nous proposons pour calculer l'ensemble de modèles d'une approximation forte de M utilise les propriétés de clôture des classes. Comme celles-ci impliquent un nombre constant de modèles pour toutes les classes considérées, à chaque étape on peut tester la clôture du sous-ensemble de M calculé jusqu'alors en temps polynomial. Notons que selon l'ordre dans lequel on considère les affectations de M, le sous-ensemble calculé ne sera pas toujours le même (sinon pour les classes NÉG et POS, puisqu'il est alors unique) ; néanmoins, dans tous les cas sa description sera une réponse correcte au problème `GLB[`$\mathcal{C}$`]`.

Classes de formules non renommables Nous présentons tout d'abord sur la figure 8.1 le modèle de l'algorithme pour les classes de formules définies en nature. Nous montrons ensuite que ce modèle est correct, et calculons la complexité exacte des algorithmes obtenus pour chaque classe $\mathcal{C}$, puis nous donnons un exemple d'exécution de l'algorithme dans le cas affine.

Nous résumons les résultats pour chaque classe $\mathcal{C}$ dans la proposition suivante, en incluant les optimisations possibles.

Proposition 8.7 (`GLB[`$\mathcal{C}$`]`) *Soit M un ensemble d'affectations à un même ensemble de variables V. On peut calculer une $\mathcal{C}$-GLB de M :*
- *en temps $O(|M||V|(|M|+|V|))$ si $\mathcal{C}$ =*Horn, Anti-Horn
- *en temps $O(|M||V|(|M|+|V|))$ si $\mathcal{C}$ =*HDP *ou* Anti-HDP *et une $\mathcal{C}$-GLB existe*
- *en temps $O(|M||V|^2)$ si $\mathcal{C}$ =*Pos *ou* Nég
- *en temps $O(|M|^3|V| + |M||V|^2)$ si $\mathcal{C}$ =*Bij
- *en temps $O(|M|^2|V|^2)$ si $\mathcal{C}$ =*Affine.

Dans tous les cas, la formule calculée contient au plus $|M||V|$ clauses (ou équations).

Preuve Pour toutes les classes $\mathcal{C}$ de l'énoncé nous montrons tout d'abord que l'algorithme `GLB[`$\mathcal{C}$`]` calcule bien une $\mathcal{C}$-GLB de M sur une donnée M. Seul le cas HDP est particulier : nous supposons que M contient l'affectation $\overline{1_V}$; dans le cas contraire, nous savons en effet qu'il n'existe pas de HDP-GLB de $\mathcal{C}$.

[correction] Soit donc ϕ la formule retournée par l'algorithme ; par construction, l'ensemble M_ϕ des affectations à V qui satisfont ϕ vérifie $M_\phi = Cl_\mathcal{C}(M_\phi)$ (c'est un invariant de la variable `modeles` de l'algorithme) ; donc M_ϕ est bien descriptible dans $\mathcal{C}$, et par construction on a $M_\phi \subseteq M$. Reste à montrer qu'il n'existe pas d'ensemble d'affectations $M' \subseteq M$ descriptible dans $\mathcal{C}$ et incluant strictement M_ϕ. Supposons donc $M' \subseteq M$ descriptible dans $\mathcal{C}$ et incluant M_ϕ ; alors M' vérifie $M' = Cl_\mathcal{C}(M')$, et en particulier $\forall M'' \subseteq M', \forall m \in M', Cl_\mathcal{C}(M'' \cup \{m\}) \subseteq M' \subseteq M$. Donc l'algorithme a bien ajouté toutes les affectations de M' à l'ensemble `modeles`, et on a $M_\phi = M'$.

[complexité] Nous montrons maintenant que cet algorithme a bien la complexité annoncée pour chacune des classes de l'énoncé. Pour la classe Horn, tout d'abord, le calcul de $Cl_\mathcal{C}$(`modeles` $\cup \{m\}$) peut être effectué en temps $O(|M||V|)$; en effet, puisque `modeles` vérifie toujours `modeles` $= Cl_{\text{Horn}}$(`modeles`), comme remarqué plus haut, on voit facilement que l'on a $Cl_\mathcal{C}$(`modeles` $\cup \{m\}$) $=$ `modeles` $\cup \{m \wedge m' \mid m' \in$ `modeles`$\}$. On peut ensuite tester l'inclusion `tmp-modeles` $\subseteq M$ en temps $O(|M||V|)$ (par exemple en triant les deux ensembles avec un arbre de décision, voir le chapitre 7), donc la boucle principale de l'algorithme requiert un temps $O(|M|(|M||V| + |M||V|)) = O(|M|^2|V|)$. Enfin, la description de `modeles` dans Horn peut être calculée en temps $O(|M||V|(|M|+|V|))$ (chapitre 7, proposition 7.9), puisque l'on a `modeles` $\subseteq M$.

Le résultat est dual pour la classe Anti-Horn, et similaire pour les classes HDP et Anti-HDP puisque en supposant $\overline{1_V} \in M$ (pour HDP) on se ramène au cas Horn.

Pour la classe Nég, on peut améliorer la complexité de l'algorithme en construisant l'ensemble d'affectations `modeles` par ordre de poids croissant, où le poids d'une affectation est le nombre de variables auxquelles elle affecte 1 ; on peut alors vérifier que la clôture $Cl_\mathcal{C}$(`modeles` $\cup \{m\}$) est incluse dans M sans la calculer, en vérifiant simplement que toutes les affectations de poids inférieur de 1 à celui de m sont dans `modeles`, en temps total $O(|V|^2)$ pour un m donné si `modeles` est représenté par un arbre de décision ; si l'inclusion est vérifiée, alors le calcul de la clôture se résume à l'ajout de m, en temps constant. La boucle principale a donc une complexité en $O(|M||V|^2)$, et puisqu'on peut décrire dans Nég en temps $O(|M||V|^2)$ (chapitre 7, proposition 7.6), on a le résultat. A nouveau, pour la classe Pos le résultat est dual.

Les résultats pour les classes Bij et Affine sont similaires à ceux pour la classe Horn. Pour la classe Bij, il suffit de remarquer que l'opération de clôture implique trois modèles au lieu de deux, d'où une complexité en $O(|M|^3|V|)$ pour la boucle principale, la description demandant

un temps $O(|M||V|^2)$ (voir chapitre 7).

Pour la classe AFFINE, on se ramène à un test de clôture impliquant seulement deux modèles en renommant M par un de ses modèles et en initialisant l'ensemble `modeles` à $\{\overline{0_V}\}$; on recherche donc un *espace vectoriel* inclus dans M et maximal ; la boucle principale requiert donc un temps $O(|M|^2|V|)$, et il suffit ensuite de renommer à nouveau l'ensemble calculé pour obtenir une AFFINE-GLB de M ; pour terminer, on peut décrire cet ensemble en temps $O(|M|^2|V|^2)$ (chapitre 7, proposition 7.5), d'où le résultat. □

A titre d'illustration, nous reprenons l'exemple du carrefour et montrons comment cet algorithme peut calculer l'AFFINE-GLB $(x_{VP} \oplus x_{VV} = 1) \wedge (x_{VP} \oplus x_{DT} = 0)$ de l'ensemble d'affectations $M_C = \{0010, 0101, 1000, 1010, 1101, 0000\}$ (dont nous réordonnons la présentation pour les besoins de l'exposé) à l'ensemble de variables $V_C = \{x_{PV}, x_{VP}, x_{VV}, x_{DT}\}$. La variable `modeles` de l'algorithme est donc initialisée à $\emptyset$.

Supposons que la première affectation de M_C considérée est $m = 0010$; comme nous sommes dans le cas affine, afin de se ramener au cadre des espaces vectoriels l'algorithme considère m comme l'affectation 0000, en renommant donc la troisième variable, x_{VV}, dans cette affectation et les suivantes. L'ensemble M_C devient donc pour l'algorithme l'ensemble

$$(M_C)_{\oplus 0010} = \{0000, 0111, 1010, 1000, 1111, 0010\}$$

et la variable `modeles`, l'ensemble $\{0000\}$. Supposons que l'algorithme considère ensuite l'affectation $m = 0111$ de l'ensemble $(M_C)_{\oplus 0010}$; la variable `tmp-modeles` reçoit alors la valeur $\{0000, 0111\}$, et cet ensemble est clos par combinaisons linéaires. La variable `modeles` reçoit donc la valeur $\{0000, 0111\}$, puis l'algorithme considère l'affectation 1010 de l'ensemble $(M_C)_{\oplus 0010}$. La clôture de l'ensemble $\{0000, 0111, 1010\}$ par combinaisons linéaires étant l'ensemble d'affectations $\{0000, 0111, 1010, 1101\}$, et comme on a $1101 \notin (M_C)_{\oplus 0010}$, l'algorithme ne modifie pas la valeur de la variable `modeles`. Est ensuite considérée l'affectation 1000 de l'ensemble $(M_C)_{\oplus 0010}$: la variable `tmp-modeles` reçoit donc la valeur $\{0000, 0111, 1000\}$, et la clôture de cet ensemble par combinaisons linéaires est l'ensemble $\{0000, 0111, 1000, 1111\}$; puisque cet ensemble est inclus dans $(M_C)_{\oplus 0010}$, il est affecté à la variable `modeles`. Il ne reste donc plus qu'à considérer l'affectation 0010, mais puisque l'on a $0111 \in$ `modeles` et $0111 \oplus 0010 = 0101 \notin (M_C)_{\oplus 0010}$, la valeur de la variable `modeles` reste inchangée. Finalement, l'algorithme a calculé l'ensemble de modèles $\{0000, 0111, 1000, 1111\}$ d'une AFFINE-GLB de $(M_C)_{\oplus 0010}$, donc sa description affine $(x_{VP} \oplus x_{VV} = 0) \wedge (x_{VP} \oplus x_{DT} = 0)$ est une AFFINE-GLB de $(M_C)_{\oplus 0010}$, et finalement le renommé de cette formule par 0010 est bien l'AFFINE-GLB de M_C annoncée.

Classes de formules renommables Nous passons maintenant aux classes des formules $\mathcal{C}$-renommables. Nous montrons que le principe d'un algorithme glouton est toujours valable, mais on doit déterminer un nouveau renommage à plusieurs reprises. L'idée générale consiste à calculer, pour la classe des formules $\mathcal{C}$-renommables, l'ensemble de modèles d'une $\mathcal{C}$-GLB de l'ensemble d'affectations donné, puis à chercher un renommage ρ permettant d'augmenter cette approximation, à calculer alors une deuxième $\mathcal{C}$-GLB, incluant la première, du renommé de la relation par ρ, et ainsi de suite. Informellement, l'algorithme est polynomial car pour toutes les classes considérées, d'une part le calcul d'une $\mathcal{C}$-GLB est polynomial, et d'autre part le choix d'un meilleur renommage est guidé par les affectations de l'ensemble donné. Notons qu'un algorithme glouton similaire est donné par Boufkhad [Bou98] pour le calcul d'une HORN-REN-GLB d'une *formule FNC* ; les principes que nous utilisons sont les mêmes.

A nouveau, la figure 8.2 présente un modèle d'algorithme, puis nous donnons les détails relatifs à chaque classe $\mathcal{C}$ dans la preuve de la proposition 8.8, et enfin nous présentons un

Entrée : Un ensemble d'affectations M à un ensemble de variables V
Sortie : Une $\mathcal{C}$-REN-GLB de M.

Début
  `modeles` $\leftarrow \emptyset$;
  Pour tout renommage ρ de V *candidat* **Faire** */* notion dépendante de $\mathcal{C}$ */*
    `tmp-modeles` $\leftarrow Cl_{\mathcal{C}}(\texttt{modeles}_{\oplus\rho})$;
    Si `tmp-modeles` $\subseteq M_{\oplus\rho}$ **Alors**
      `modeles` $\leftarrow$ l'ensemble de modèles d'une $\mathcal{C}$-GLB de $M_{\oplus\rho}$ incluant `tmp-modeles` ;
      `modeles` $\leftarrow \texttt{modeles}_{\oplus\rho}$;
    Fin Si ;
  Fin Pour ;
  retourner une description de `modeles` dans $\mathcal{C}$-REN ;
Fin

FIG. 8.2 – Algorithme `GLB-Ren[`$\mathcal{C}$`]` (modèle)

exemple d'exécution. Rappelons simplement que la classe MONOT des formules FNC monotones peut être vue comme la classe des formules NÉG-renommables.

Proposition 8.8 (`GLB-Ren[`$\mathcal{C}$`]`) *Soit M un ensemble d'affectations à un même ensemble de variables V. On peut calculer une $\mathcal{C}$-REN-GLB quelconque de M :*
- *en temps $O(|M|^2|V|(|M|+|V|))$ si $\mathcal{C}$ =*HORN
- *en temps $O(|M|^2|V|(|M|+|V|))$ si $\mathcal{C}$ =*HDP *et $M \neq \emptyset$*
- *en temps $O(|M|^2|V|^2)$ si $\mathcal{C}$ =*NÉG.

Dans tous les cas, la formule calculée contient au plus $|M||V|$ clauses.

Preuve Comme pour la proposition 8.7, nous montrons tout d'abord que le modèle d'algorithme est correct, puis nous étudions sa complexité pour les différentes classes de formules.

[correction] Nous montrons tout d'abord que l'algorithme est correct si l'on considère comme *candidats* tous les renommages de V, quelle que soit la classe $\mathcal{C}$. Par construction, le fait que l'ensemble `modeles` de l'algorithme soit inclus dans M et descriptible dans $\mathcal{C}$ est un invariant de l'algorithme (pour $\mathcal{C}$ = HDP, à partir du moment où une affectation lui a été ajoutée). Si maintenant, par l'absurde, la description ϕ retournée n'est pas une $\mathcal{C}$-REN-GLB de M, alors il existe un renommage ρ de V et un ensemble M' d'affectations à V tels que l'on ait $(M_\phi)_{\oplus\rho} \subseteq M' \subseteq M_{\oplus\rho}$, où M_ϕ désigne l'ensemble des affectations à V qui satisfont ϕ. De fait, ni $(M_\phi)_{\oplus\rho}$ ni aucun de ses sous-ensembles n'est l'ensemble de modèles d'une $\mathcal{C}$-GLB de $M_{\oplus\rho}$, et en particulier l'ensemble `modeles` calculé par l'algorithme en considérant ρ, ce qui est absurde.

[complexité] Pour ce qui est de la complexité, nous utilisons les résultats du chapitre 4 (paragraphe 4.2). Pour la classe HORN, tout d'abord, nous avons vu qu'un ensemble d'affectations était renommable en un ensemble descriptible dans HORN si et seulement s'il l'était par l'un de ses modèles (chapitre 4, proposition 4.4). Nous pouvons donc considérer comme renommages «candidats» les seuls $\rho \in M$; la boucle principale de l'algorithme est donc exécutée $|M|$ fois ; pour un renommage ρ fixé, on peut tester $Cl_{\text{HORN}}(\texttt{modeles}_{\oplus\rho}) \subseteq M_{\oplus\rho}$ en calculant les éléments de cette clôture un par un, et en testant pour chacun s'il est dans $M_{\oplus\rho}$ (en temps $O(|V|)$ avec une représentation de M sous forme d'un arbre de décision) ; dès que ce n'est pas le cas, on peut arrêter le calcul, et si on a bien l'inclusion au plus $|M|$ affectations auront été énumérées.

Pour l'énumération, il suffit de considérer les produits $m \wedge m'$ pour $m, m' \in \texttt{modeles}_{\oplus\rho}$ ou m, m' déjà énumérés, et finalement le test d'inclusion requiert un temps $O(|M|^2|V|)$; le calcul de l'ensemble de modèles d'une HORN-GLB de $M_{\oplus\rho}$ requiert alors un temps $O(|M||V|(|M|+|V|))$ en utilisant l'algorithme `GLB[HORN]` avec son ensemble de modèles initialisé à $\texttt{modeles}_{\oplus\rho}$, et finalement la boucle principale a une complexité en $O(|M|(|M|^2|V| + |M||V|(|M|+|V|)))$, donc en $O(|M|^2|V|(|M|+|V|))$.

La complexité est la même pour la classe HDP-REN en vertu de la proposition 4.7 du chapitre 2, qui montre que l'on peut se limiter aux renommages ρ tels que $\rho \oplus \overline{1_V}$ est dans M.

Pour le cas $\mathcal{C} =$ NÉG, enfin, on peut tester $Cl_{\text{NÉG}}(\texttt{modeles}_{\oplus\rho}) \subseteq M_{\oplus\rho}$ en énumérant les affectations inférieures à chaque $m \in \texttt{modeles}$ par ordre de poids décroissant (voir la preuve de la proposition 8.7), en temps $O(|M||V|)$ au total si l'on s'arrête dès qu'on sort de M, et le calcul d'une NÉG-GLB de $M_{\oplus\rho}$ requiert un temps $O(|M||V|^2)$ (proposition 8.7). Puisque l'on peut se limiter aux renommages $\rho \in M$ (proposition 4.9 page 59), la boucle principale demande un temps $O(|M|^2|V|^2)$, et comme la description requiert un temps $O(|M||V|^2)$ (proposition 7.6 du chapitre 7), on a le résultat. □

8.3.2 Calcul d'une approximation forte avec le nombre maximal de modèles

Nous nous intéressons maintenant au calcul d'une $\mathcal{C}$-GLB d'un ensemble d'affectations M donné *admettant un nombre maximal de modèles* parmi toutes les $\mathcal{C}$-GLBs de M. Plus formellement, nous définissons la notion de *max-$\mathcal{C}$-GLB*.

Définition 8.3 (max-$\mathcal{C}$-GLB) *Soient $\mathcal{C}$ une classe de formules et M un ensemble d'affectations à un même ensemble de variables V. Une $\mathcal{C}$-approximation forte maximale ϕ de M est appelée une* max-$\mathcal{C}$-approximation forte maximale *(*max-$\mathcal{C}$-GLB*) de M si pour toute $\mathcal{C}$-GLB ϕ' de M, les nombres n (resp. n') d'affectations à V satisfaisant ϕ (resp. ϕ') vérifient $n \geq n'$.*

Le problème que nous étudions est donc, étant donnée une classe de formules $\mathcal{C}$, celui du calcul d'une max-$\mathcal{C}$-GLB d'un ensemble d'affectations donné. Rappelons que ce problème est le même que celui du calcul d'une $\mathcal{C}$-GLB *quelconque* lorsque $\mathcal{C}$ est la classe NÉG ou la classe POS, puisque alors la $\mathcal{C}$-GLB d'un ensemble d'affectations donné est unique à l'équivalence logique près.

Problème 8.3 (`Max-GLB[`$\mathcal{C}$`]`)

Donnée : *Un ensemble d'affectations M*
Sortie : *Une max-$\mathcal{C}$-approximation forte maximale de M.*

Rappelons que l'étude de ce problème est principalement motivée par le fait que, pour les classes de formules considérées ici, il existe des familles de relations ayant plusieurs familles de $\mathcal{C}$-GLBs, éventuellement avec des nombres de modèles exponentiellement différents ; il est donc naturel de considérer comme *meilleures* celles dont le nombre de modèles est le plus élevé.

A notre connaissance, ce problème n'a été étudié que pour la classe des formules de Horn, dans [KPS93]. Les auteurs y montrent que ce calcul est intraitable, en montrant la NP-difficulté du problème de décision associé suivant.

Problème 8.4 (D-Max-GLB[$\mathcal{C}$])

Donnée : *Un ensemble d'affectations M à un même ensemble de variables V et un entier positif $k \leq |M|$*
Question : *Existe-t-il une $\mathcal{C}$-GLB ϕ de M telle que le nombre n d'affectations à V satisfaisant ϕ vérifie $n \geq k$?*

Clairement, si le calcul d'une max-$\mathcal{C}$-GLB est polynomial, alors ce problème l'est également. Nous rappelons le résultat de [KPS93] et en redonnons une preuve ; celle de l'article est en effet douteuse (leur réduction à partir du problème CLIQUE donne un résultat faux même avec un triangle).

Proposition 8.9 (Max-GLB[Horn] **[KPS93]**) *Les problèmes :*

D-Max-GLB[Horn], D-Max-GLB[Anti-Horn]

sont NP-complets.

Preuve Nous ne prouvons explicitement que le cas Horn ; le cas anti-Horn est symétrique. Pour l'appartenance à NP, il suffit de deviner l'ensemble de modèles d'une $\mathcal{C}$-GLB ϕ de M : par définition d'une (G)LB, le nombre de ces modèles est inférieur à $|M|$, et on peut donc vérifier en temps polynomial (en la taille de M) que ce nombre est supérieur ou égal à k et que l'ensemble deviné forme une *$\mathcal{C}$-LB* de M (avec la propriété de clôture par exemple). Pour la NP-difficulté, nous réduisons le problème Independent-Set au problème D-Max-GLB[Horn] en temps polynomial ; la NP-complétude de ce problème [GJ79] conclura.

[ensembles indépendants] Si G est un graphe non orienté, d'ensemble de sommets S, et S' est un sous-ensemble de S, S' est appelé un *ensemble indépendant* pour G si aucune paire $\{x, x'\}$ de sommets de S' n'est jointe par une arête de G. Le problème de décision Independent-Set consiste alors à déterminer, étant donné un graphe non orienté G sur un ensemble de sommets S et un entier positif $k \leq |S|$, s'il existe un ensemble indépendant pour G de cardinal au moins k. Etant donnés un graphe non orienté G et un entier k, nous montrons comment construire en temps polynomial un ensemble d'affectations $M(G)$ à un certain ensemble de variables $V(G)$, qui admettra une Horn-GLB satisfaite par au moins $k+1$ affectations à $V(G)$ si et seulement si G admet un ensemble indépendant de cardinal au moins k.

[réduction] Soit donc G un graphe non orienté sur un ensemble de sommets S. L'ensemble de variables $V(G)$ est défini comme suit : $V(G)$ contient une variable x_a pour chaque arête a de G, et une variable x_s pour tout sommet $s \in S$; intuitivement, les variables correspondant aux sommets serviront à identifier uniquement les affectations de $M(G)$. L'ensemble d'affectations $M(G)$ est défini comme l'ensemble $\overline{0_{V(G)}} \cup \{m_s \in \{0,1\}^{V(G)} \mid s \in S\}$, où m_s est donné par :

$$\begin{cases} m_s[x_a] = 0 \text{ si l'arête } a \text{ de } G \text{ n'est pas adjacente au sommet } s \\ m_s[x_a] = 1 \text{ si l'arête } a \text{ de } G \text{ est adjacente au sommet } s \\ m_s[x_s] = 1 \\ m_s[x_{s'}] = 0 \text{ pour tout sommet } s' \neq s \end{cases}$$

Clairement, $V(G)$ et $M(G)$ peuvent être construits en temps polynomial en la taille de G.

[correction] Supposons que G admet un ensemble indépendant $S' \subseteq S$ de taille $|S'| \geq k$, et soit M' l'ensemble d'affectations à $V(G)$ $\{\overline{0_{V(G)}}\} \cup \{m_s \mid s \in S'\}$. Nous montrons que M' est clos

par l'opérateur $\wedge$. Tout d'abord, on a $\overline{0_{V(G)}} \wedge m_s = \overline{0_{V(G)}} \in M'$ pour tout $m_s \in M'$. Soient maintenant $m_s, m_{s'} \in M'$, $m_s \neq m_{s'}$. Puisque S' est indépendant, aucune arête de G ne joint s et s', donc pour aucune arête a de G on n'a $m_s[x_a] = m_{s'}[x_a] = 1$; de plus, puisque m_s et $m_{s'}$ sont différents, pour aucun sommet $s'' \in S$ on n'a $m_s[x_{s''}] = m_{s'}[x_{s''}] = 1$. Finalement, on a $m_s \wedge m_{s'} = \overline{0_{V(G)}} \in M'$, et donc M' est descriptible dans HORN, ce qui montre que $M(G)$ admet une HORN-GLB satisfaite par au moins $|M'| = |S'| + 1 \geq k + 1$ affectations à $V(G)$.

Réciproquement, supposons que G n'admet pas d'ensemble indépendant de taille au moins k. Cela signifie que pour tout sous-ensemble S' de S de taille au moins k, il existe deux sommets $s, s' \in S'$ et une arête a de G tels que a joint s et s'. De fait, pour tout sous-ensemble M' de $M(G)$ de taille au moins $k+1$ (le $+1$ provenant de l'affectation $\overline{0_{V(G)}}$), il existe deux affectations $m_s, m_{s'} \in M'$ et une variable $x_a \in V(G)$ avec $m_s[x_a] = m_{s'}[x_a] = 1$. On en déduit $(m_s \wedge m_{s'})[x_a] = 1$, mais puisque m_s et $m_{s'}$ sont différents on a pour tout sommet $s'' \in S$, $(m_s \wedge m_{s'})[x''_s] = 0$. Une telle affectation n'existe pas dans $M(G)$, donc M' ne vérifie pas $M' = Cl_{\text{HORN}}(M')$, et comme cela est vrai pour tout $M' \subseteq M$ de taille au moins $k+1$ on en déduit que M' n'admet pas de HORN-GLB satisfaite par au moins $k+1$ affectations à $V(G)$. □

La sémantique des formules FNC de Horn définies positives étant très proche de celle des formules FNC de Horn, on déduit facilement de la proposition 8.9 le corollaire suivant.

Corollaire 8.1 (`Max-GLB[HDP]`) *En supposant $P \neq NP$, une max-*HDP*-GLB d'un ensemble d'affectations donné, M, à un ensemble de variables V avec $\overline{1_V} \in M$ ne peut pas être calculée en temps polynomial. Le même résultat est valable pour la classe duale* ANTI-HDP.

Preuve Supposons par l'absurde que l'on peut calculer une max-HDP-GLB de tout ensemble d'affectations à un ensemble de variables V contenant $\overline{1_V}$ en temps polynomial. Soit alors M un ensemble d'affectations à V. Si M contient $\overline{1_V}$, alors on peut calculer une max-HORN-GLB de M en temps polynomial, puisque alors les HORN-GLBs et les HDP-GLBs de M sont les mêmes. Si à l'inverse M ne contient pas $\overline{1_V}$, alors on peut calculer une max-HDP-GLB ϕ de $M \cup \{\overline{1_V}\}$ en temps polynomial ; il est alors facile de voir que la formule $\phi' = \phi \wedge (\bigvee_{x \in V} \neg x)$ vérifie $\mathcal{M}(\phi') = \mathcal{M}(\phi) \backslash \{\overline{1_V}\}$ et est en FNC de Horn ; ϕ' est donc une max-HORN-GLB de M, et finalement on peut toujours calculer une max-HORN-GLB de M en temps polynomial, ce qui contredit la proposition 8.9. □

Nous montrons ci-dessous que le même résultat est encore valable pour la classe des formules 2FNC, avec une preuve similaire, puis que, à l'inverse, le calcul d'une max-AFFINE-GLB est de complexité subexponentielle.

Proposition 8.10 (`Max-GLB[BIJ]`) *Le problème* `D-Max-GLB[BIJ]` *est NP-complet.*

Preuve L'appartenance à NP se montre exactement comme pour la proposition 8.9. Pour la difficulté, nous montrons que la même construction convient également. Nous renvoyons donc à la preuve de cette proposition pour la construction et les notations.

Soit donc G un graphe non orienté sur un ensemble de sommets S, et soit $k \leq |S|$ un entier. Soit tout d'abord $S' \subseteq S$ un ensemble indépendant de G avec $|S'| \geq k$, et soit M' le sous-ensemble de $M(G)$ défini comme $\{\overline{0_{V(G)}}\} \cup \{m_s \mid s \in S'\}$. Remarquons tout d'abord que pour $m_s, m_{s'} \in M'$, on a $maj(\overline{0_V}, m_s, m_{s'}) = \overline{0_V} \in M'$, car pour une variable $x_a \in V(G)$ associée à une arête de G, m_s et $m_{s'}$ n'affectent pas tous deux 1 à x_a, et de même pour les variables x_s associées aux sommets. Pour la même raison, si $m_{s''}$ est une troisième affectation de M' on

a $maj(m_s, m_{s'}, m_{s''}) = \overline{0_V} \in M'$. Donc M' vérifie $M' = Cl_{\text{BIJ}}(M')$, et donc M admet une BIJ-GLB satisfaite par au moins $|M'| = |S'| + 1 \geq k + 1$ affectations à $V(G)$. La réciproque se montre de même que pour la proposition 8.9, en distinguant trois affectations $m_s, m_{s'}, m_{s''}$ et une variable x_a telles que l'arête a joint les sommets s et s' dans G. □

Nous montrons enfin le résultat pour la classe AFFINE. La preuve utilise toujours le fait qu'une base d'un espace vectoriel le caractérise de façon unique tout en étant de taille logarithmique. Nous utilisons également une recherche par dichotomie de la taille maximale d'une telle base d'une AFFINE-GLB de l'ensemble d'affectations donné. Remarquons qu'une complexité subexponentielle est en quelque sorte intermédiaire entre une complexité polynomiale et un résultat de NP-complétude, puisqu'on estime généralement qu'en supposant $P \neq NP$, aucun problème NP-complet n'admet d'algorithme subexponentiel (voir [SHI90] par exemple).

Proposition 8.11 (`Max-GLB[AFFINE]`) *Soit M un ensemble d'affectations à un même ensemble de variables V. Une max-*AFFINE*-GLB de M peut être calculée en temps subexponentiel*

$$O((\log_2 \log_2 |M|)|M||V|2^{(\log_2 |M|)^2})$$

Preuve Nous supposons toujours $\overline{0_V} \in M$, sans perte de généralité (proposition 3.4 page 46 et discussion qui suit), afin de nous ramener au cas des espaces vectoriels.

Nous cherchons donc un sous-espace vectoriel de M de taille maximale ; nous remarquons qu'une base d'un tel espace vectoriel est également de taille maximale parmi les bases des sous-espaces vectoriels de M. Nous cherchons donc le plus grand entier $k \in \{1, 2, \ldots, \log |M|\}$ tel qu'il existe une base de taille k d'un espace vectoriel inclus dans M.

Pour un tel entier k fixé, nous essayons les $\binom{|M|}{k}$ sous-ensembles de M de taille k ; pour B un tel ensemble, nous testons si B est linéairement indépendant, en temps $O(k^2|V|)$ avec le pivot de Gauss, et si l'espace vectoriel E de base B est inclus dans M ; ce dernier test peut être effectué en temps $O(|M||V|)$ en générant une par une toutes les affectations de E, puisque le test peut alors s'arrêter dès que l'on trouve une affectation qui n'est pas dans M, et car les affectations égales à des combinaisons linéaires différentes d'affectations linéairement indépendantes sont nécessairement différentes.

Pour un entier k fixé, on peut donc déterminer s'il existe un sous-espace vectoriel de M de taille 2^k en temps $O(\binom{|M|}{k})(k^2|V| + |M||V|)$, et donc en temps $O(|M|^k(k^2|V| + |M||V|))$. Nous recherchons alors le plus grand k pour lequel la réponse est positive par dichotomie dans l'ensemble $\{1, 2, \ldots, \log_2 |M|\}$, et finalement on obtient une complexité totale en :

$$O((\log_2 \log_2 |M|)|M|^{\log_2 |M|}((\log_2 |M|)^2|V| + |M||V|))$$

et donc en :

$$O((\log_2 \log_2 |M|)2^{(\log_2 |M|)^2}(|M||V|))$$

Il suffit alors de décrire la base obtenue par une formule affine, ce qui est possible en temps $O(|M||V|^3)$ (proposition 3.5 du chapitre 3) ; nous considérons cette complexité négligeable devant la précédente. □

Problème LUB[$\mathcal{C}$]			
Classe $\mathcal{C}$	Complexité	Nb. clauses	Référence
Horn, Anti-Horn	intraitable	exponentiel	[KKS95], *
HDP, Anti-HDP	intraitable	exponentiel	*
Nég, Pos	intraitable	exponentiel	*
Bij	$O(\|M\|\|V\|^2)$	$\leq \frac{3}{2}\|V\|^2$	[DP92]
Affine	$O(\|M\|\|V\|^3)$	$\leq \|V\|$	[KS98], *

Problème GLB[$\mathcal{C}$]			
Classe $\mathcal{C}$	Complexité	Nb. clauses	Référence
Horn, Anti-Horn	$O(\|M\|\|V\|(\|M\|+\|V\|))$	$\leq \|M\|\|V\|$	[KPS93], *
HDP, Anti-HDP	$O(\|M\|\|V\|(\|M\|+\|V\|))$	$\leq \|M\|\|V\|$	*
Nég, Pos	$O(\|M\|\|V\|^2)$	$\leq \|M\|\|V\|$	*
Bij	$O(\|M\|^3\|V\|+\|M\|\|V\|^2)$	$\leq \frac{3}{2}\|V\|^2$	*
Affine	$O(\|M\|^2\|V\|^2)$	$\leq \|V\|$	*
Horn-Ren, HDP-Ren	$O(\|M\|^2\|V\|(\|M\|+\|V\|))$	$\leq \|M\|\|V\|$	*
Monot	$O(\|M\|^2\|V\|^2)$	$\leq \|M\|\|V\|$	*

Problème Max-GLB[$\mathcal{C}$] (ou D-Max-GLB[$\mathcal{C}$])			
Classe $\mathcal{C}$	Complexité	Nb. clauses	Référence
Horn, Anti-Horn	intraitable	—	[KPS93], *
HDP, Anti-HDP	intraitable	—	*
Bij	intraitable	—	*
Affine	$O((\log_2 \log_2 \|M\|)\|M\|\|V\|2^{(\log_2 \|M\|)^2})$	$\leq \|V\|$	*

Tab. 8.1 – Complexité de l'approximation

8.4 Résumé des résultats connus

Nous résumons les résultats connus pour la complexité du calcul des approximations faibles et fortes d'un ensemble d'affectations M dans la table 8.1. Nous indiquons également, comme pour la description et l'identification, le nombre de clauses (ou d'équations linéaires) de la formule produite par l'algorithme correspondant, lorsqu'on en connaît un de complexité polynomiale.

Chapitre 9

PAC-apprentissage

Sommaire

9.1 Présentation et travaux précédents 125
9.1.1 Formalisation du problème 125
9.1.2 Motivations et remarques 127
9.1.3 Travaux précédents 129
9.2 PAC-apprenabilité de la classe des formules affines 130
9.2.1 Algorithme de mise à jour 130
9.2.2 Complexité 132
9.3 Résumé des résultats connus 134

Ce dernier chapitre consacré à l'acquisition de connaissances s'intéresse à l'*apprentissage à partir d'exemples*, et plus précisément à la formalisation de cette problématique proposée par Valiant, le *PAC-apprentissage*. Nous proposons un état de l'art sur le sujet, dans le cadre des classes de formules qui nous intéressent, et montrons que celle des formules affines est PAC-apprenable, résultat nouveau à notre connaissance.

9.1 Présentation et travaux précédents

Dans ce premier paragraphe, nous présentons tout d'abord la formalisation du problème d'apprentissage que propose Valiant [Val84], puis nous motivons son étude dans notre cadre, et enfin nous proposons un bref état de l'art sur le sujet.

9.1.1 Formalisation du problème

Le problème du PAC-apprentissage d'une classe de formules $\mathcal{C}$ peut être vu comme un prolongement naturel des problématiques d'acquisition de connaissances que nous avons étudiées dans les chapitres précédents. En effet, la *description* et l'*identification* s'intéressent à la représentation *exacte* d'un ensemble d'affectations donné dans une classe de formules $\mathcal{C}$ donnée, l'*approximation* à sa représentation *approchée* dans $\mathcal{C}$; le PAC-apprentissage, lui, s'intéresse à la représentation *approchée* dans $\mathcal{C}$ d'un ensemble d'affectations *qui n'est pas donné directement*. Informellement, étant donnée une classe de formules $\mathcal{C}$, le PAC-apprentissage d'un ensemble d'affectations M descriptible dans $\mathcal{C}$ est le problème consistant à calculer une formule ϕ de $\mathcal{C}$ dont l'ensemble de modèles soit proche de M, et ce sans que M soit donné à l'algorithme : seul un *sous-ensemble* de M, tiré aléatoirement, est disponible.

Points clefs de la formalisation La formalisation proposée par Valiant [Val84], que nous reprenons strictement, passe par trois points importants. Dans la suite, nous supposons fixée une classe de formules $\mathcal{C}$ et un ensemble d'affectations M à un ensemble de variables V, *descriptible dans* $\mathcal{C}$ (nous discuterons cette restriction dans le paragraphe 9.1.2).

Le premier point de la formalisation concerne le sous-ensemble de M accessible. On suppose fixée une distribution de probabilités D sur M, et le sous-ensemble de M accessible à un algorithme de PAC-apprentissage pour $\mathcal{C}$ est tiré aléatoirement, affectation par affectation et indépendamment, selon cette distribution. Néanmoins, l'algorithme ne connaît pas D ; en cela, le modèle de Valiant est indépendant de la distribution de probabilité sur les affectations. Intuitivement, le tirage aléatoire d'un sous-ensemble de M formalise l'observation *au hasard*, et la distribution D formalise la probabilité que l'on a d'observer chacun des exemples.

Le deuxième point concerne la qualité de la formule apprise. Etant donné le sous-ensemble de M tiré aléatoirement (rappelons que les tirages sont effectués affectation par affectation et indépendamment), un algorithme doit produire une formule ϕ de la classe $\mathcal{C}$, telle que les affectations à V qui satisfont ϕ forment un sous-ensemble de M (ϕ doit donc être une $\mathcal{C}$-approximation forte de M) ; cette restriction est appelée *one-sided errors*, et on dit qu'on n'autorise que des erreurs *négatives*. La qualité de la réponse ϕ de l'algorithme est alors mesurée par la divergence entre M et l'ensemble des modèles de ϕ. Plus précisément, on utilise toujours la distribution de probabilités D sur l'ensemble M, et, informellement, l'erreur de ϕ est mesurée par la somme des probabilités des affectations de $M \backslash \mathcal{M}(\phi)$. Plus cette erreur est importante, moins bonne est l'approximation de M par ϕ. Par exemple, si M est l'ensemble d'affectations de l'exemple de la thèse :

$$M_t = \{00111, 01000, 10100, 11011\}$$

sur l'ensemble de variables $V_t = \{x_{me}, x_{qu}, x_{ba}, x_{in}, x_{ch}\}$, muni de la distribution de probabilités uniforme, la formule :

$$\phi_t \wedge (x_{ba} \vee x_{in} \vee x_{ch})$$

satisfaite par l'ensemble d'affectations à V_t $\{00111, 01000, 11011\}$ réalise une erreur de $1/4$.

Nous demanderons que le nombre minimum d'affectations de M qui doivent être rendues disponibles à l'algorithme pour obtenir une formule ϕ d'une qualité donnée croisse lentement avec cette qualité. Intuitivement, la mesure de la qualité de la formule calculée formalise l'idée que l'on ne peut pas tout comprendre d'un concept, ou d'un petit monde, à partir de quelques exemples, mais que l'on désire converger rapidement vers une compréhension exacte lorsque l'on utilise plus de ressources pour obtenir des exemples.

Enfin, le troisième point de la formalisation concerne le taux de réussite de l'algorithme. En effet, puisque les affectations accessibles sont tirées aléatoirement, on ne peut demander à un algorithme de calculer une approximation de M de bonne qualité avec probabilité 1. On demande donc qu'il calcule une approximation de bonne qualité avec une certaine probabilité, mais que pour rendre cette probabilité plus forte on n'ait besoin que d'un petit nombre d'exemples supplémentaires.

Notons que les deuxième et troisième points expliquent l'acronyme «PAC» : «Probablement Approximativement Correct».

Définitions Nous pouvons finalement donner une formalisation complète du problème ; nous la scindons en deux parties, la définition du *problème d'apprentissage*, tout d'abord, puis celle de la *PAC-apprenabilité* d'une classe de formules. Notons cependant que les formalisations de ces notions sont nombreuses dans la littérature ; nous reviendrons sur certaines d'entre elles à la fin de ce paragraphe.

Problème 9.1 (PAC-apprentissage[$\mathcal{C}$])

Donnée : *Un sous-ensemble M' d'un ensemble d'affectations M à un ensemble de variables V (M étant muni d'une distribution de probabilités D fixée), et un réel $\varepsilon_Q \in]0,1[$*
Sortie : *Une formule $\phi \in \mathcal{C}$ telle que la somme des probabilités $D(m)$, pour toutes les affectations m à V qui sont dans M mais ne satisfont pas ϕ, soit au plus ε_Q.*

Nous mesurons la complexité d'un algorithme qui résout ce problème en fonction du nombre $|M'|$ d'affectations accessibles, du nombre $|V|$ de variables et de l'inverse $1/\varepsilon_Q$ du paramètre ε_Q. Nous nous intéressons de plus à la probabilité avec laquelle cet algorithme résout le problème, et comme nous l'avons dit, nous attendons de l'algorithme que le nombre minimal d'exemples pour lequel il résout le problème avec une bonne probabilité croisse lentement avec la qualité et le taux de réussite demandés. Nous introduisons donc la définition suivante.

Définition 9.1 (support minimal) *Soient $\mathcal{C}$ une classe de formules et* A *un algorithme pour le problème* PAC-apprentissage[$\mathcal{C}$]. *Soient V un ensemble de variables, M un ensemble d'affectations à V muni d'une distribution de probabilités D et $\varepsilon_Q, \varepsilon_P \in]0,1[$ deux réels. On appelle* support minimal de A pour $(M, \varepsilon_Q, \varepsilon_P)$ *le plus petit entier k tel que sur une donnée (M', ε_Q), où M' est un sous-ensemble de M de taille k tiré aléatoirement (selon D, affectation par affectation et indépendamment),* A *résout le problème* PAC-apprentissage[$\mathcal{C}$] *avec une probabilité supérieure ou égale à $1 - \varepsilon_P$.*

Nous sommes alors en mesure de définir la *PAC-apprenabilité* d'une classe de formules.

Définition 9.2 (PAC-apprenabilité) *Soit $\mathcal{C}$ une classe de formules. On dit que $\mathcal{C}$ est* PAC-apprenable *s'il existe un algorithme* A *pour le problème* PAC-apprentissage[$\mathcal{C}$] *tel que, pour tout ensemble d'affectations M à un ensemble de variables V et pour tous réels $\varepsilon_Q, \varepsilon_P \in]0,1[$:*
- *le support minimal de* A *pour $(M, \varepsilon_Q, \varepsilon_P)$ est polynomial en $1/\varepsilon_Q$, $1/\varepsilon_P$ et $|V|$*
- A *est de complexité polynomiale.*

Notons qu'on autorise en général le support de A à être également polynomial en la taille de la plus petite description de M dans $\mathcal{C}$. Mais nous n'utilisons pas cette possibilité pour la classe des formules affines.

9.1.2 Motivations et remarques

Classification Comme nous l'avons dit, le PAC-apprentissage d'une classe de formules formalise intuitivement l'apprentissage à partir d'exemples, ou de situations, observés au hasard. Tel qu'il est formalisé dans le paragraphe 9.1.1, ce problème correspond particulièrement à celui de la *classification supervisée* : il s'agit d'apprendre une définition en intension d'un concept, ici représenté par l'ensemble d'affectations M, à partir d'exemples et de contre-exemples fournis par un expert du domaine (supposé infaillible) ; ici, le rôle de l'expert est donc formalisé par le tirage aléatoire des affectations de M. Cet expert peut également être vu comme un professeur, comme le propose Valiant, et l'algorithme d'apprentissage est alors vu comme un élève. Ce problème de classification survient naturellement dans de nombreuses applications pratiques ; nous renvoyons le lecteur à l'ouvrage de Hastie *et al.* [HTF01] pour plus de détails.

Biais de langage Dans notre cadre de systèmes à base de connaissances, la classe de formules $\mathcal{C}$ représente la classe des formules acceptées par le système de gestion des connaissances. L'ensemble d'affectations M «à intégrer», dont on requiert qu'il soit *descriptible dans* $\mathcal{C}$, doit intuitivement être interprété, non comme la représentation exacte, en extension, du concept à intégrer, mais plutôt comme l'extension *du concept descriptible dans* $\mathcal{C}$ *le plus proche de celui à intégrer* ; intuitivement, c'est donc «ce que l'on peut apprendre de mieux», puisqu'on ne s'autorise que des formules de $\mathcal{C}$. Mais la plupart du temps, la classe $\mathcal{C}$ est également vue, en apprentissage, comme un *biais* : on suppose alors que le «vrai» concept à intégrer peut être représenté avec très peu d'erreur dans $\mathcal{C}$, et cette classe permet alors à un algorithme de *généraliser* l'ensemble des exemples accessibles (on pourrait même dire que le biais l'y «oblige») ; en effet, si l'on n'impose pas de restrictions sur la classe de formules $\mathcal{C}$, la stratégie d'apprentissage la plus sûre est l'apprentissage *par coeur* de l'ensemble d'exemples reçu, tandis que si la classe $\mathcal{C}$ est peu expressive, pour pouvoir apprendre une formule de $\mathcal{C}$ il faut généraliser cet ensemble ; ce processus est appelé *induction*, et la classe $\mathcal{C}$ lui sert en quelque sorte de guide.

De fait, un algorithme d'apprentissage a d'autant plus de chances d'apprendre une définition fidèle d'un concept, et ce à partir de peu d'exemples, que son biais de langage $\mathcal{C}$ permet de représenter le concept en intension, d'une part, et qu'il est peu expressif, d'autre part ; à l'extrême, si la classe $\mathcal{C}$ ne contient qu'une formule, un algorithme biaisé par $\mathcal{C}$ peut induire la définition dans $\mathcal{C}$ la plus proche d'un concept donné à partir d'aucun exemple ; en revanche, il y a peu de chances pour que la définition apprise soit fidèle.

L'article de Sebag [Seb96], par exemple, discute largement le problème des biais d'apprentissage.

PAC-apprentissage dynamique Mais revenons à notre exemple de l'extra-terrestre qui veut apprendre le fonctionnement du petit monde du carrefour. Nous avons affirmé (chapitre 6) que la problématique du PAC-apprentissage formalisait le cas où l'extra-terrestre ne peut observer le carrefour que de façon sporadique. Cette interprétation du PAC-apprentissage est en fait plus naturelle avec une version *dynamique* des définitions du paragraphe 9.1.1. En plus de ces définitions, il s'agit alors d'imposer qu'un algorithme capable d'apprendre une formule ϕ à partir d'un ensemble M' d'exemples puisse également *mettre* ϕ *à jour* s'il reçoit un nouvel exemple $m \in M$; par *mise à jour de* ϕ, nous entendons le calcul, à partir de ϕ et de m, d'une nouvelle formule ϕ' qui soit une réponse correcte au problème `PAC-apprentissage[`$\mathcal{C}$`]` pour la donnée $(M' \cup \{m\} \subseteq M, \varepsilon_Q)$, avec ε_Q le même paramètre que celui selon lequel a été apprise ϕ. Intuitivement, il ne s'agit donc plus d'apprendre une définition à partir d'une base de données d'exemples, mais plutôt d'apprendre au hasard des observations. Notons que les algorithmes donnés par Valiant [Val84] et par Angluin *et al.* [AFP92] sont dynamiques. Dans ce chapitre, nous montrons que la classe des formules affines est PAC-apprenable avec ce rafinement de la problématique.

Utilisation des connaissances acquises Comme pour les autres problématiques que nous étudions dans cette partie, nous avons en tête l'utilisation future de la base de connaissances acquise pour des tâches de raisonnement. Puisqu'une base de connaissances acquise par PAC-apprentissage n'est qu'une approximation de la réalité qu'elle décrit, d'une part à cause du biais de langage, et d'autre part à cause du tirage aléatoire d'exemples, son utilisation n'est pas directe. Néanmoins, il apparaît que la façon la plus naturelle de raisonner avec une telle base de connaissances est celle lui faisant «confiance», dans l'acception présentée dans le chapitre 8 (paragraphe 8.1.2). En effet, les deux autres visions du raisonnement à partir de connaissances approximatives sont inadaptées, puisque les liens exacts entre la base de connaissances et l'en-

semble d'affectations M qu'elle est censée représenter au mieux ne sont pas connus ; l'ensemble M lui-même n'est pas connu.

9.1.3 Travaux précédents

Le nombre de variantes de la problématique du PAC-apprentissage ainsi que le nombre de résultats présents dans la littérature étant très importants, nous ne présentons ici que les résultats concernant les classes de formules qui nous intéressent pour représenter des connaissances explicitement, et décrivons informellement les cadres dans lesquels ils ont été établis.

Origine et variantes de la problématique Comme nous l'avons dit, la problématique du PAC-apprentissage a été formalisée par Valiant dans son article fondateur [Val84]. Il y propose les définitions que nous avons reprises ici, mais propose également une notion d'apprentissage plus *active* : il s'agit de donner à l'algorithme d'apprentissage la possibilité de demander à un «oracle» si une affectation donnée est ou non dans l'ensemble d'affectations à apprendre ; cette nouvelle *requête* (Valiant présente également le tirage aléatoire d'une affectation comme une requête) est appelée *requête d'appartenance* («membership query»), et fournit donc une première variante de la notion de *PAC-apprenabilité*.

Une autre forme de requête est introduite par Angluin *et al.* dans leurs travaux sur la classe Horn [AFP90, AFP92]. Il s'agit des *requêtes d'équivalence avec contre-exemples*, qui permettent à l'algorithme de demander si une formule donnée ϕ est ou non une description (exacte) de l'ensemble d'affectations M à apprendre ; si la réponse est négative, l'«oracle» retourne de plus un *contre-exemple*, c'est-à-dire une affectation de la différence symétrique de M et $\mathcal{M}(\phi)$. Ces requêtes sont surtout utilisées dans le cadre de l'*identification exacte*, c'est-à-dire lorsque l'on recherche une *description* de l'ensemble M à apprendre ; c'est donc la problématique de la description (chapitre 7), mais avec un accès limité à M. Encore une fois, cette requête propose une vision plus *active* de l'apprentissage.

Nous renvoyons le lecteur à l'article d'Aizenstein *et al.* [AHHP98, introduction], par exemple, pour une discussion des différents formalismes proposés pour le PAC-apprentissage.

Classes de formules Valiant [Val84] donne de premiers résultats concernant certaines classes de formules. Il montre principalement que la classe des formules kFNC, pour une constante k fixée, est PAC-apprenable à partir d'exemples seulement et que la classe Pos-FND est PAC-apprenable à partir d'exemples *et* de requêtes d'équivalence (avec une complexité mesurée en fonction de la taille de la plus petite formule FND décrivant M).

La classe des formules FNC de Horn est montrée *exactement identifiable* par Angluin, Frazier et Pitt [AFP92], si l'on autorise des requêtes d'appartenance et d'équivalence (avec contre-exemples). Bien entendu, cet algorithme permet de montrer le même résultat pour la classe duale Anti-Horn, mais également pour les classes HDP et Anti-HDP ; il suffit en effet (pour HDP) de considérer dans tous les cas qu'on a reçu l'affectation de toutes les variables à 1, puis d'utiliser un algorithme pour la classe Horn. Notons qu'un algorithme pour l'identification exacte avec requêtes d'appartenance et d'équivalence peut être transformé en un algorithme pour le PAC-apprentissage à partir d'exemples et de requêtes d'appartenance ; informellement, il s'agit de remplacer les requêtes d'équivalence par un test *passif*, en attendant de voir surgir un contre-exemple parmi ceux tirés aléatoirement (voir l'explication d'Angluin [Ang90]).

La PAC-apprenabilité de la classe Bij, quant à elle, est un cas particulier de celle de la classe des formules kFNC, celui où k vaut 2. De fait, cette classe est PAC-apprenable à partir

d'exemples seulement [Val84]. Enfin, la classe POS-FND des formules FND positives est exactement identifiable avec des requêtes d'appartenance et d'équivalence, et donc PAC-apprenable à partir d'exemples et de requêtes d'appartenance [Bsh95, DMP99], et de même pour la classe POS des formules *FNC* positives [DMP99, annexe] ; par dualité enfin, il en va de même pour les classes NÉG et NÉG-FND. On peut également voir que le même algorithme permet d'identifier les formules FNC ou FND *monotones* ; en effet, il suffit de déterminer le signe de chaque variable comme nous l'avons proposé pour la description (chapitre 7, paragraphe 7.4.1) ; essentiellement, ce problème est le même que pour les classes de formules négatives[3].

Quelques résultats négatifs ont également été formulés ; en particulier, Angluin [Ang90] donne de tels résultats *absolus* (qui ne dépendent pas de conjectures telles que P≠NP, par exemple), en montrant notamment que la classe des formules FNC positives n'est pas apprenable avec des requêtes d'équivalence seulement. On en déduit, les requêtes d'équivalence étant symétriques, que le même résultat vaut pour les classes NÉG, POS-FND et NÉG-FND, et pour les mêmes raisons que ci-dessus pour les classes MONOT et MONOT-FND. Enfin, à notre connaissance aucun résultat, ni positif, ni négatif, n'a été donné dans la littérature pour les classes de formules HORN-REN et HDP-REN.

Notre apport Nous montrons dans le reste de ce chapitre que la classe des formules affines est PAC-apprenable à partir d'exemples seulement [Zan02a]. L'algorithme que nous donnons est de plus *dynamique*.

9.2 PAC-apprenabilité de la classe des formules affines

Nous montrons maintenant que la classe des formules affines est PAC-apprenable à partir d'exemples, de manière dynamique. Nous donnons tout d'abord l'algorithme polynomial qui permet de mettre à jour une formule précédemment apprise en fonction de nouveaux exemples, puis nous montrons que le support minimal de notre algorithme croît lentement avec les inverses des paramètres d'erreur ε_Q et ε_P.

9.2.1 Algorithme de mise à jour

L'algorithme d'apprentissage que nous proposons pour la classe AFFINE permet de mettre à jour une formule affine en fonction d'un nouvel exemple. Plus précisément, soit M un ensemble d'affectations à un ensemble de variables V, et soit ϕ une AFFINE-approximation forte de M ; l'algorithme que nous proposons, étant donnés ϕ et un exemple $m \in M$, calcule alors une (nouvelle) approximation forte ϕ' de M, satisfaite par (au moins) toutes les affectations à V qui satisfont ϕ et par m.

Soit donc un ensemble M d'affectations à un ensemble de variables V, soit ϕ une AFFINE-approximation forte de M, et soit $m \in M$. Notons M_ϕ l'ensemble des affectations à V qui satisfont ϕ (on a donc $M_\phi \subseteq M$). Si m satisfait ϕ, ce qui est vérifiable en temps linéaire en la taille de ϕ, nous la laissons inchangée. Dans le cas contraire, puisque la formule ϕ' que nous voulons calculer doit être affine, l'ensemble des affectations à V qui la satisferont ne peut pas en général être l'ensemble $M_\phi \cup \{m\}$; au mieux, c'est-à-dire avec le moins d'induction possible, ce sera $Cl_{\text{AFFINE}}(M_\phi \cup \{m\})$. Nous montrons qu'on peut calculer une formule ϕ' satisfaite par *exactement* cet ensemble d'affectations à V en supprimant de ϕ une de ses équations non satisfaite

[3] La littérature d'apprentissage, en particulier, nomme «monotones» les formules que nous nommons «positives» [Bsh95, DMP99].

Entrée : Un ensemble de variables V, une formule affine ϕ et une affectation m à V
Sortie : Une formule affine ϕ' satisfaite par les affectations de $Cl_{\text{AFFINE}}(\{m' \in \{0,1\}^V \mid m' \models \phi\} \cup \{m\})$, et par aucune autre affectation à V.

Début
  Si $m \models \phi$ **Alors**
    $\phi' \leftarrow \phi$;
  Sinon
    $E \leftarrow$ une équation de ϕ avec $m \not\models E$;
    $\phi' \leftarrow \emptyset$;
    Pour toute équation $E' \in \phi$ avec $E' \neq E$ **Faire**
      Si $m \models E'$ **Alors** $\phi' \leftarrow \phi' \cup \{E'\}$
      Sinon $\phi' \leftarrow \phi' \cup \{E' \oplus E\}$
      Fin Si ;
    Fin Pour ;
  Fin Si ;
  retourner ϕ' ;
Fin

FIG. 9.1 – Algorithme `MAJAffine`

par m, et en l'ajoutant membre à membre aux autres équations de ϕ non satisfaites par m. Plus précisément, nous proposons l'algorithme de la figure 9.1.

Ainsi, soit M_t l'ensemble d'affectations de l'exemple de la thèse, et soit à mettre à jour son AFFINE-approximation forte :

$$(x_{me} = 0) \wedge (x_{qu} = 0) \wedge (x_{qu} \oplus x_{ba} = 1) \wedge (x_{me} \oplus x_{qu} \oplus x_{in} = 1) \wedge (x_{ch} = 1)$$

dont l'ensemble de modèles est $\{00111\}$, en fonction de l'affectation $01000 \in M_t$. On voit que la deuxième équation de la formule n'est pas satisfaite par cette affectation, nous la supprimons donc de la formule et l'ajoutons membre à membre à la seule autre équation non satisfaite, $(x_{ch} = 1)$. Nous obtenons ainsi la formule mise à jour :

$$(x_{me} = 0) \wedge (x_{qu} \oplus x_{ba} = 1) \wedge (x_{me} \oplus x_{qu} \oplus x_{in} = 1) \wedge (x_{qu} \oplus x_{ch} = 1)$$

dont l'ensemble de modèles est bien l'ensemble $\{00111, 01000\} = Cl_{\text{AFFINE}}(\{00111\} \cup \{01000\})$.

Lemme 9.1 *Soient V un ensemble de variables, ϕ une formule affine et m une affectation à V. L'algorithme* `MAJAffine` *calcule une formule affine ϕ', satisfaite par les affectations de $Cl_{\text{AFFINE}}(\{m' \in \{0,1\}^V \mid m' \models \phi\} \cup \{m\})$ et par aucune autre affectation à V, en temps linéaire en la taille de ϕ.*

Preuve La complexité de l'algorithme est directe ; nous montrons donc qu'il est correct. Le cas $m \models \phi$ étant trivial, nous supposons le contraire.

[sens direct] Soit tout d'abord m_0 une affectation à V satisfaisant ϕ ; alors pour toutes équations linéaires $E, E' \in \phi$, m_0 satisfait E et E', donc m_0 satisfait E' et $E \oplus E'$, et finalement m_0 satisfait ϕ'. Nous montrons maintenant que m satisfait ϕ'. Soit $E = (x_{i_1} \oplus x_{i_2} \oplus \cdots \oplus x_{i_k} = a)$, avec $x_{i_1}, x_{i_2}, \ldots, x_{i_k} \in V$ et $a \in \{0,1\}$, l'équation linéaire de ϕ vérifiant $m \not\models E$ et sélectionnée

par l'algorithme. De $m \not\models E$ on déduit $m[x_{i_1}] \oplus m[x_{i_2}] \oplus \cdots \oplus m[x_{i_k}] = a \oplus 1$. Soit alors E_0 une équation de ϕ' ; si on a $E_0 = E'$ pour une équation $E' \in \phi$, alors par construction on a $m \models E_0$; dans le cas contraire, on a $E_0 = E \oplus E'$ pour une équation $E' \in \phi$ avec $m \not\models E'$. En notant $E' = (x'_{i_1} \oplus x'_{i_2} \oplus \cdots \oplus x'_{i_{k'}} = a')$, on a $m[x'_{i_1}] \oplus m[x'_{i_2}] \oplus \cdots \oplus m[x'_{i_{k'}}] = a' \oplus 1$, et on en déduit $m[x_{i_1}] \oplus m[x_{i_2}] \oplus \cdots \oplus m[x_{i_k}] \oplus m[x'_{i_1}] \oplus m[x'_{i_2}] \oplus \cdots \oplus m[x'_{i_{k'}}] = a \oplus 1 \oplus a' \oplus 1 = a \oplus a'$, c'est-à-dire $m \models E \oplus E' = E_0$. Finalement, m satisfait ϕ. Nous avons donc $\forall m_0 \in \{m' \in \{0,1\}^V \mid m' \models \phi\} \cup \{m\}, m_0 \models \phi'$, et comme ϕ est affine nous avons donc $\forall m_0 \in Cl_{\text{AFFINE}}(\{m' \in \{0,1\}^V \mid m' \models \phi\} \cup \{m\}), m_0 \models \phi'$.

[réciproque] Nous montrons maintenant la réciproque ; soit donc m' une affectation à V satisfaisant ϕ'. Si on a $m' = m$ ou $m' \models \phi$ c'est trivial. Nous supposons donc $m' \neq m$ et $m' \not\models \phi$. Soit alors m_0 une affectation à V satisfaisant ϕ (si ϕ est insatisfaisable, tout est trivial) ; nous montrons que l'affectation $m_0 \oplus m \oplus m'$ satisfait ϕ.

Soit E' une équation de ϕ. Si on a $m \models E'$, alors on a $m_0, m, m' \models E'$, et donc on a bien $m_0 \oplus m \oplus m' \models E'$. Soit maintenant E' avec $m \not\models E'$. Alors, par construction de ϕ' l'équation $E'' = E \oplus E'$ est dans ϕ' (où E est l'équation sélectionnée par l'algorithme), et on a donc $m_0 \oplus m \oplus m' \models E''$. D'autre part, puisque m' ne satisfait pas ϕ on a $m' \not\models E$; en effet, dans le cas contraire m' satisferait toutes les équations de ϕ' et toutes leurs sommes modulo 2 avec E, donc toutes les équations de ϕ. Finalement, en notant $E = (x_{i_1} \oplus x_{i_2} \oplus \cdots \oplus x_{i_k} = a)$ on a :

$$\begin{array}{rcl} \bigoplus_{j=1}^{k}(m_0 \oplus m \oplus m')[x_{i_j}] & = & (\bigoplus_{j=1}^{k} m_0[x_{i_j}]) \oplus (\bigoplus_{j=1}^{k} m[x_{i_j}]) \oplus (\bigoplus_{j=1}^{k} m'[x_{i_j}]) \\ & = & (a) \oplus (a \oplus 1) \oplus (a \oplus 1) \\ & = & (a) \end{array}$$

donc $m_0 \oplus m \oplus m' \models E$, et comme $m_0 \oplus m \oplus m' \models E'' = E \oplus E'$ on a finalement $m_0 \oplus m \oplus m' \models E \oplus E \oplus E' = E'$.

On conclut finalement que l'affectation $m_0 \oplus m \oplus m'$ est égale à une affectation m'' à V avec $m'' \models \phi$, et on a donc $m' = m_0 \oplus m \oplus m''$, qui est par définition de l'AFFINE-clôture dans $Cl_{\text{AFFINE}}(\{m' \in \{0,1\}^V \mid m' \models \phi\} \cup \{m\})$. □

9.2.2 Complexité

Nous montrons maintenant que l'algorithme présenté dans le paragraphe 9.2.1 permet de prouver la PAC-apprenabilité de la classe AFFINE. L'algorithme d'apprentissage que nous utilisons consiste tout d'abord en l'initialisation de la formule affine ϕ par la première affectation $m \in M$ reçue : nous posons alors $\phi = \{(x = m[x]) \mid x \in V\}$, qui est satisfaite par une seule affectation à V, m. Puis, tant que nous avons accès à de nouveaux exemples, nous mettons ϕ à jour avec l'algorithme `MAJAffine`. Remarquons que l'adaptation non dynamique de cet algorithme est directe : si toutes les affectations de M tirées aléatoirement sont disponibles à l'initialisation de ϕ, il suffit ensuite d'appliquer l'algorithme précédent en considérant ces affectations une par une. Nous appelons `ApprendreAffine` cette version non dynamique.

Si nous reprenons l'exemple de la thèse (paragraphe 9.2.1), avec $M' = \{00111, 01000\}$, l'algorithme calcule donc tout d'abord la formule :

$$(x_{me} = 0) \wedge (x_{qu} = 0) \wedge (x_{ba} = 1) \wedge (x_{in} = 1) \wedge (x_{ch} = 1)$$

à partir de l'affectation 00111, puis il met à jour cette formule en fonction de l'affectation 01000, obtenant ainsi la formule :

$$(x_{me} = 0) \wedge (x_{ba} \wedge x_{qu} = 1) \wedge (x_{in} \wedge x_{qu} = 1) \wedge (x_{ch} \wedge x_{qu} = 1)$$

La complexité de l'algorithme `ApprendreAffine` est donc en $O(|M'||V|^2)$ pour une donnée (M', ε_Q), puisqu'il consiste en $|M'|$ exécutions de l'algorithme `MAJAffine` sur une formule contenant toujours au plus $|V|$ équations.

Nous montrons maintenant que le support minimal pour cet algorithme est polynomial en ses paramètres. Afin de pouvoir utiliser des outils de probabilité classiques, nous considérons le tirage aléatoire d'une affectation $m \in M$ dans le contexte de l'algorithme, en considérant la formule ϕ calculée avant la mise à jour qu'implique m. Nous associons alors à ce tirage l'événement probabiliste réalisé exactement lorsque m ne satisfait pas ϕ ; nous le considérons intuitivement comme un succès car il nous permet de mettre réellement ϕ à jour, et donc de diminuer strictement la divergence entre l'ensemble des affectations à V qui satisfont ϕ et l'ensemble M. Rappelons du paragraphe 9.1.1 que les tirages des affectations sont indépendants les uns des autres.

Nous pouvons alors montrer la proposition suivante.

Proposition 9.1 (support minimal) *Soit M un ensemble d'affectations à un ensemble de variables V, avec M muni d'une distribution de probabilités fixée D, et soient deux réels $\varepsilon_Q, \varepsilon_P \in]0,1[$. Le support minimal de l'algorithme* `ApprendreAffine` *pour $(M, \varepsilon_Q, \varepsilon_P)$ est inférieur à $2\max(1/\varepsilon_Q, 1/\varepsilon_P)(|V| + \ln\max(1/\varepsilon_Q, 1/\varepsilon_P))$.*

Preuve Nous formalisons tout d'abord le problème en termes d'événements probabilistes et de réussite, puis nous concluons avec un outil de probabilité classique. Nous supposons, sans perte de généralité, que la première affectation tirée aléatoirement dans M est l'affectation $\overline{0_V}$, afin de nous placer dans le cadre des espaces vectoriels.

[tirages et succès] Rappelons tout d'abord que l'espace vectoriel M est caractérisé par une base de taille $\log_2 |M|$. Remarquons alors que si les tirages aléatoires retournent un ensemble M' contenant un nombre maximum d'affectations linéairement indépendantes, puisque l'algorithme calcule un sur-ensemble affine de M' il calcule une description affine de ϕ. Nous considérons donc le tirage aléatoire d'une affectation m de M comme un *succès* si m est linéairement indépendant de l'ensemble d'affectations tiré auparavant.

Etant donnés un ensemble d'affectations M et la donnée $(M' \subseteq M, \varepsilon_Q \in]0,1[)$ d'un problème d'apprentissage pour la classe AFFINE, nous considérons la situation où l'algorithme `ApprendreAffine` *ne* résout *pas* le problème, et montrons qu'elle ne survient qu'avec une probabilité au plus ε_P si la taille de M' est supérieure à la valeur de l'énoncé.

Supposons donc que l'algorithme a échoué, et notons ϕ la formule calculée et M_ϕ l'ensemble d'affectations à V satisfaisant ϕ. On a donc $\sum_{m \in M \setminus M_\phi} D(m) > \varepsilon_Q$. Or, ϕ étant une formule affine (homogène), toutes les affectations de $M \setminus M_\phi$ sont linéairement indépendantes de M_ϕ, et donc également de tout sous-ensemble de M_ϕ. On en déduit que tous les tirages aléatoires d'affectations avaient une probabilité de succès strictement supérieure à ε_Q.

[conclusion] Nous supposons toujours que l'algorithme a échoué. Nous ne savons pas combien de tirages aléatoires ont été un succès, mais nous savons qu'il y en a eu strictement moins de $\log_2 |M|$, car sinon l'algorithme aurait calculé une description de M. Donc, formellement, $|M'|$ tirages aléatoires indépendants ont été menés, tous avec une probabilité de succès strictement supérieure à ε_Q, mais strictement moins de $\log_2 |M|$ succès ont été obtenus (en particulier, strictement moins que $|V|$). Or la première proposition de [Val84] (paragraphe 4, p.1137) nous dit qu'il suffit de moins de $2\max(1/\varepsilon_Q, 1/\varepsilon_P)(|V| + \ln\max(1/\varepsilon_Q, 1/\varepsilon_P))$ tirages pour que cela arrive avec une probabilité inférieure à ε_P. □

Nous en déduisons immédiatement la proposition suivante.

Classe	Requêtes	Cadre	apprenable	Références
HORN, ANTI-HORN	équiv., appart.	identification	oui	[AFP92]
HDP,ANTI-HDP	équiv., appart.	identification	oui	[AFP92]
NÉG,POS	équiv., appart.	identification	oui	[Bsh95, DMP99]
NÉG,POS	équiv.	identification	non	[Ang90]
BIJ	exemples	PAC	oui	[Val84]
AFFINE	exemples	PAC	oui	*
MONOT	équiv., appart.	identification	oui	[Bsh95, DMP99]
MONOT	équiv.	identification	non	[Ang90]

TAB. 9.1 – Complexité de l'apprentissage

Corollaire 9.1 (PAC-apprenabilité de AFFINE) *La classe des formules affines est PAC-apprenable à partir d'exemples seulement, et de manière dynamique.*

Remarque 9.1 (VC-dimension) *Un autre outil classique en apprentissage est la* dimension de Vapnik-Chervonenkis *(VC-dimension) d'une classe de formules. Cet outil permet lui aussi de montrer que le nombre d'exemples suffisant pour apprendre un ensemble d'affectations affine est petit, mais nous avons préféré la preuve plus directe donnée pour la proposition 9.1, par souci de simplicité. Nous renvoyons cependant le lecteur à l'article de Floyd et Warmuth [FW95], par exemple, pour plus de détails. Notons que la VC-dimension de la classe des formules affines sur $|V|$ variables est $|V|+1$ (intuitivement, c'est le cardinal maximal d'un ensemble d'affectations M à V tel qu'aucune affectation $m \in M$ n'est dans l'*AFFINE*-clôture de $M\backslash\{m\}$).*

9.3 Résumé des résultats connus

Nous terminons par un résumé des résultats connus concernant la PAC-apprenabilité des classes de formules qui nous intéressent (FNC et affines). Pour chaque classe $\mathcal{C}$, la table 9.1 indique si $\mathcal{C}$ est PAC-apprenable ou non, ainsi que les requêtes autorisées et le cadre dans lequel le résultat a été établi. Lorsque nous indiquons «équivalence», nous sous-entendons «avec contre-exemples», et de même nous sous-entendons «exacte» lorsque nous parlons d'«identification».

Troisième partie

Raisonnement

Chapitre 10

Introduction

Sommaire

10.1 Présentation et motivations 137
10.2 Problématiques étudiées 138
10.3 Déduction, point de vue «exact» 140

Nous nous intéressons dans cette dernière partie aux problèmes de *raisonnement* à partir d'une base de connaissances représentée dans l'une des classes de formules présentées dans la partie I. Les problèmes que nous abordons sont encore une fois définis formellement dans chaque chapitre, mais nous les présentons ici brièvement, de façon globale.

10.1 Présentation et motivations

Traditionnellement, la dénomination «processus de raisonnement» englobe de nombreux problèmes très différents, autant dans leurs applications que dans leur algorithmique : certains sont fondamentaux, comme c'est le cas du problème `SAT`, d'autres correspondent à des problèmes très naturels, à la formalisation souvent plus complexe, comme le diagnostic ; certains sont des problèmes de décision, comme la déduction, d'autres des problèmes avec sortie, comme l'abduction. De façon générale cependant, le raisonnement implique une partie *fixe*, la base de connaissances, qui sera utilisée pour répondre à de nombreuses *requêtes* différentes. De fait, la donnée d'un problème de raisonnement comprend en général ces deux parties, une base de connaissances et une requête.

Considérons à nouveau l'exemple du carrefour, et l'extra-terrestre entré en scène dans le chapitre 6. Une fois qu'il a acquis une base de connaissances à propos de ce petit monde, par exemple celle présentée dans les préliminaires généraux (paragraphe 3.2), il peut l'utiliser pour répondre à de nombreuses requêtes concernant ce même petit monde. Par exemple, il peut lui «demander» s'il est vrai qu'il a le droit de traverser dès que le feu pour piétons est vert, ou s'il y a toujours au moins un des deux feux au vert ; il peut encore lui demander ce qui suffit à lui interdire de traverser, ce qu'il devrait changer s'il voulait en reprendre le droit ; ou encore, il peut lui demander s'il est possible que le feu pour véhicules et le feu pour piétons soient verts en même temps, etc. Pour ce qui est de l'exemple de la thèse, la base de connaissances présentée dans les préliminaires généraux (paragraphe 3.3) peut par exemple être utilisée pour répondre à des questions comme «Que dois-je faire pour que la lecture de ma thèse soit mémorable ?», «Puis-je produire un document de qualité tout en le bâclant ?», etc.

Encore une fois, nous supposons l'existence d'une base de connaissances, et ne discutons pas la pertinence des variables sur lesquelles elle est formée pour modéliser la réalité qu'elle est censée décrire, ni celle de l'ensemble des situations qu'elle envisage implicitement. Nous supposons de plus que les bases de connaissances sont représentées par des formules des classes présentées dans la partie I ; nous nous intéressons au raisonnement à partir de ces bases elles-mêmes, supposées représentées de fait par des formules en FNC ou affines, mais également au raisonnement à partir des *négations* de ces bases, représentées de fait par des formules en FND ou de la classe DisjAffine. Nous ne revenons pas sur nos motivations pour considérer ces représentations, motivations exposées notamment dans le chapitre 1. A nouveau, nous étudions autant de fois chaque problème qu'il y a de classes de formules, mais nous étudions également les classes des *requêtes*, puiqu'elles ont bien entendu une influence significative sur la complexité des problèmes.

10.2 Problématiques étudiées

Nous nous intéressons plus précisément à trois problématiques de raisonnement, de natures très différentes, et nous considérons chacune d'elles sous deux angles différents. Ces problématiques sont classiques, et font l'objet de nombreux travaux dans la littérature. Notons que contrairement aux problèmes traités dans la partie II, pour ceux-ci la classe de formules $\mathcal{C}$ dans laquelle est représentée la base de connaissances de référence paramètre la *donnée* des algorithmes ; en conséquence, si un problème est facile pour une classe $\mathcal{C}$, il l'est au moins autant pour toute sous-classe $\mathcal{C}'$ de $\mathcal{C}$.

Cohérence et déduction, point de vue «exact» Le premier angle sous lequel nous considérons les problématiques consiste à considérer que la base de connaissances à notre disposition représente *exactement* la réalité qu'elle est censée décrire, comme c'est le cas, par exemple, pour une base de connaissances acquise par *description* (voir le chapitre 7). Sous cet angle, à la fois le plus naturel et le plus classique, nous abordons trois tâches de raisonnement. Tout d'abord, nous considérons la *vérification de cohérence* d'une base de connaissances, plus formellement le problème `SAT` pour cette base. Intuitivement, cette tâche consiste à vérifier que notre connaissance du monde n'est pas absurde, c'est-à-dire qu'elle envisage bien au moins une situation possible dans le petit monde de référence, ou définit au moins un exemple du concept qu'elle représente. Le contraire pourrait se produire, par exemple, après l'ajout de nouvelles contraintes à la base.

Cette problématique est particulière, en ce sens qu'elle ne dépend pas d'une requête. De fait, pour une même base de connaissances le problème de vérification de cohérence n'a besoin d'être résolu qu'une fois. Cette problématique est donc plus naturelle si la base est souvent mise à jour. Bien sûr, les bases de connaissances des exemples du carrefour et de la thèse sont toutes cohérentes.

La deuxième problématique que nous considérons avec le point de vue «exact» est celle de la *déduction* : il s'agit là, étant donnée une *requête*, c'est-à-dire une formule, formalisant intuitivement une *assertion* à propos du petit monde de référence, de décider si cette requête est ou non impliquée par la base de connaissances.

Dans le petit monde du carrefour, un problème de déduction doit par exemple être résolu lorsque l'extra-terrestre se demande s'il a le droit de traverser dès que le feu pour piétons est vert. En effet, il s'agit alors de demander à sa base de connaissances si elle implique la requête «Si le feu est vert, on a le droit de traverser». C'est encore le problème à résoudre lorsqu'il se demande s'il y a toujours au moins l'un des deux feux au vert, puisqu'il s'agit alors de demander à la base de connaissances si elle implique la requête «Soit le feu pour piétons, soit le feu pour

véhicules est vert».

Mais la déduction est également le problème à résoudre pour répondre à des questions telles que «Est-il possible que le feu pour piétons et celui pour véhicules soient verts en même temps ?». En effet, cela revient à «demander» à la base de connaissances si elle n'implique pas le *contraire* de cette assertion, en l'occurrence l'assertion «Il y a toujours au moins un des deux feux qui n'est pas au vert». De même, c'est un problème de déduction de ce type qui formalise la question «Puis-je produire un document de qualité tout en le bâclant ?» de l'exemple de la thèse ; il s'agit en effet de savoir si la base de connaissances *n'*implique *pas* l'assertion «Si je produis un document de qualité, c'est que je ne l'aurai pas bâclé».

Néanmoins, les problématiques de vérification de cohérence et de déduction, sous le point de vue exact, ne soulèvent que peu de problèmes intéressants dans notre cadre. En effet, par hypothèse les classes de formules dans lesquelles nous concevons la représentation de connaissances sont toutes traitables pour le problème `SAT`, puisque c'est l'une de nos motivations pour les étudier ; nous renvoyons donc à la partie I pour les algorithmes efficaces correspondants. Pour ce qui est de la déduction, nous rappelons simplement des résultats classiques et très généraux dans le paragraphe 10.3.

Abduction, point de vue «exact» Toujours en considérant que la base de connaissances à notre disposition représente *exactement* la réalité qu'elle est censée décrire, nous étudions ensuite la problématique de l'*abduction* (chapitre 11) : il s'agit désormais, étant donné un *but*, c'est-à-dire une formule formalisant une assertion à propos du petit monde de référence, d'*expliquer* ce but en essayant de lui trouver une cause plausible, cohérente avec la base de connaissances. C'est donc un problème avec sortie, contrairement à la vérification de cohérence ou à la déduction. Le but, quant à lui, peut être vu comme représentant un fait observé dans le petit monde et que l'on veut expliquer, ou encore un objectif à atteindre, un fait dont on veut s'assurer, etc. On demande également, en général, que les explications fassent partie d'un certain ensemble d'*hypothèses* ; nous détaillerons ce point dans le chapitre 11, mais de fait, la *requête*, pour un problème d'abduction, est constituée d'un but *et* d'un ensemble d'hypothèses.

Si l'on considère le petit monde du carrefour, un problème d'abduction doit par exemple être résolu lorsque notre extra-terrestre se demande ce qui suffit à lui interdire de traverser ; en effet, il s'agit alors de trouver une explication plausible à la validité (imaginaire) du but «Il est interdit de traverser». L'explication «Le feu pour véhicules est vert» est alors une réponse correcte à ce problème, puisque, conjointe à la base de connaissances, elle implique bien la requête, et parce qu'elle est cohérente avec la base. A l'inverse, l'assertion «Les deux feux sont verts» n'est pas une réponse correcte à ce problème, puisque sa conjonction avec la base de connaissances n'est pas cohérente.

La problématique de l'abduction formalise également la question «Que dois-je faire pour que la lecture de ma thèse soit mémorable ?» de l'exemple de la thèse. Le but «La lecture de ma thèse est mémorable» est alors plus naturellement compris comme un objectif à atteindre, et l'éventuelle explication trouvée, comme un moyen d'atteindre cet objectif. Ainsi, une réponse correcte à ce problème est la formule «Bâcler le travail et le faire lire à quelqu'un qui n'est pas chercheur en informatique».

Remarquons qu'un problème d'abduction n'admet pas toujours une réponse. Ainsi, le problème «Que dois-je faire pour avoir le droit de traverser alors que le feu pour véhicules est vert ?», soumis à la base de connaissances de l'exemple du carrefour, n'en admet aucune. Remarquons également que le processus d'abduction *ne préserve pas la vérité*. Cela signifie intuitivement que l'éventuelle réponse trouvée n'est pas nécessairement la véritable cause du fait observé, par

exemple. Ainsi, dans l'exemple du carrefour, si le fait que le feu pour voitures soit vert justifie bien une interdiction de traverser, il y a des situations où il n'est pas vert, mais où l'on n'a pas non plus le droit de traverser ; par exemple, la situation où les deux feux sont rouges.

Le processus d'abduction a également été beaucoup étudié dans la littérature, et sa complexité est déjà connue pour toutes les classes de formules FNC qui nous intéressent (avec des classes de requêtes naturelles). Nous complétons cet état de l'art dans le chapitre 11, en donnant un modèle d'algorithme général, et en l'étudiant lorsque la base de connaissances de référence est représentée par une formule affine ou par une formule FND. Rappelons que nous entendons cette dernière situation comme le cas où l'on veut raisonner avec la *négation* d'une base de connaissances (voir le chapitre 1, paragraphe 1.1).

Point de vue «approximation» Nous abordons enfin à nouveau les trois mêmes problématiques (chapitre 12), mais avec un point de vue différent. Il s'agit cette fois de considérer que la base de connaissances dont nous disposons n'est qu'une *approximation* de notre réelle connaissance du monde, par exemple acquise avec l'un des algorithmes présentés dans le chapitre 8, et d'essayer de comprendre quelle est la sémantique des processus de raisonnement dans ce cadre ; c'est donc l'étude des utilisations «méfiante» et «restrictive» d'une base de connaissances approximative (voir la discussion du chapitre 8, paragraphe 8.1.2). Ainsi, la question générale que nous nous posons est la suivante :

> Supposons que nous répondions à un problème de raisonnement avec une base de connaissances approximative. Que pouvons-nous déduire, à partir de la réponse obtenue, sur celle que nous aurions obtenue avec une représentation exacte de la réalité ?

Notons que, si la vérification de cohérence et la déduction sont déjà bien étudiées dans la littérature sous ce point de vue, ce n'est pas le cas pour l'abduction ; nos résultats pour ce problème sont de fait principalement de premières pierres dans ce domaine.

10.3 Déduction, point de vue «exact»

Comme nous l'avons dit, les résultats pour la déduction, lorsque la base de connaissances de référence est représentée dans l'une des classes de formules qui nous intéressent, sont à la fois simples, classiques et généraux.

Rappelons tout d'abord que pour les questions du type «Est-il possible que le feu pour piétons et celui pour véhicules soient verts en même temps ?», on se ramène à une question du type «La base de connaissances implique-t-elle une requête donnée ?» en considérant la *négation* de la requête précédente. Plus formellement, nous traitons la question «α est-il possible ?», pour une requête α donnée, comme la question «ϕ (n')implique-t-elle (pas) $\neg\alpha$?», où ϕ est la base de connaissances de référence. Remarquons que si α est en FNC ou en FND, on peut représenter $\neg\alpha$ par la formule $Neg(\alpha)$ sans changer la réponse au problème.

Nous nous intéressons donc au problème de décision suivant ; puisque sa complexité dépend de la forme syntaxique de la base de connaissances, mais aussi de celle de la requête, nous le paramétrons par deux classes de formules, $\mathcal{C}_\phi$ et $\mathcal{C}_\alpha$.

Problème 10.1 (`Deduction`$[\mathcal{C}_\phi,\mathcal{C}_\alpha]$)

Donnée : *Deux formules* ϕ, α *avec* $\phi \in \mathcal{C}_\phi$ *et* $\alpha \in \mathcal{C}_\alpha$
Question : *A-t-on* $\phi \models \alpha$ *?*

On voit tout d'abord aisément que la traitabilité du problème de satisfaisabilité $\texttt{SAT}[\mathcal{C}_\phi]$, pour une classe de formules $\mathcal{C}_\phi$ donnée, est une condition presque suffisante à la traitabilité du problème $\texttt{Deduction}[\mathcal{C}_\phi,\textsc{Clauses}]$. En effet, on voit que si ϕ est une formule quelconque, et $\alpha = (\ell_1 \vee \ell_2 \vee \cdots \vee \ell_k)$, une clause quelconque, les ℓ_i étant des littéraux, alors on a $\phi \models \alpha$ si et seulement si la formule $\phi \wedge Neg(\alpha) = \phi \wedge \overline{\ell_1} \wedge \overline{\ell_2} \wedge \cdots \wedge \overline{\ell_k}$, où $\overline{\ell_i}$ est le littéral opposé du littéral ℓ_i, est insatisfaisable. En considérant alors l'affectation m à $Var(\alpha)$ définie pour tous les littéraux ℓ de α et toutes les variables x de $Var(\alpha)$ par : $\ell = x \Longrightarrow m[x] = 1$ et $\ell = \neg x \Longrightarrow m[x] = 0$, on voit que ϕ implique α si et seulement si la formule obtenue en propageant m dans ϕ est insatisfaisable. Comme toutes les classes de formules que nous avons vues dans la partie I sont des sous-classes de l'une des classes Horn-Ren (pour les formules 2FNC, seulement celles qui sont satisfaisables, mais dans le cas contraire les problèmes sont alors triviaux), Affine, Fnd ou DisjAffine, et comme ces trois classes sont stables par propagation d'affectations quelconques et traitables pour SAT, on peut conclure que la déduction de clauses est traitable pour une base de connaissances représentée dans l'une quelconque des classes qui nous intéressent. Pour ce qui est de la déduction de formules FNC, le même résultat est valable, puisqu'il suffit de remarquer qu'une formule implique une conjonction d'autres formules si et seulement si elle implique chacune d'elles. Le problème de la déduction d'une formule FNC contenant k clauses se ramène donc à k problèmes de déduction de clauses.

A titre d'illustration, reprenons la base de connaissances de l'exemple du carrefour :

$$\phi_C = (\neg x_{DT} \vee x_{VP}) \wedge (\neg x_{VP} \vee x_{DT}) \wedge (\neg x_{PV} \vee \neg x_{VV} \vee \neg x_{DT}) \wedge (\neg x_{VP} \vee \neg x_{VV})$$

et soit la requête en FNC :

$$\alpha_C = (\neg x_{VP} \vee \neg x_{VV}) \wedge (x_{VP} \vee x_{VV})$$

Le problème de déduction pour (ϕ_C, α_C) revient donc à décider l'insatisfaisabilité de la formule $(\phi_C \wedge Neg(\alpha_C))$, et donc à décider l'insatisfaisabilité des deux formules $(\phi_C \wedge x_{VP} \wedge x_{VV})$ et $(\phi_C \wedge \neg x_{VP} \wedge \neg x_{VV})$. Or, en considérant la deuxième de ces formules, après la propagation de l'affectation «forcée» 00 à $\{x_{VP}, x_{VV}\}$ dans ϕ_C, on obtient la formule :

$$(\neg x_{DT})$$

qui est satisfaisable. On en déduit que la formule $(\phi_C \wedge \neg x_{VP} \wedge \neg x_{VV})$, et donc la formule $(\phi_C \wedge Neg(\alpha_C))$, est satisfaisable, et on peut donc donner la réponse $\phi_C \not\models \alpha_C$ au problème.

Proposition 10.1 ($\texttt{Deduction}[\mathcal{C}_\phi,\textsc{Fnc}]$) *Le problème* $\texttt{Deduction}[\mathcal{C}_\phi,\textsc{Fnc}]$ *est traitable pour toutes les classes de formules présentées dans la partie I.*

En revanche, lorsque la requête est une formule en FND, il est facile de voir que le problème de déduction est coNP-difficile dans toute sa généralité, et le reste même si on se restreint aux bases de connaissances représentées dans l'une des classes de formules FNC qui nous intéressent. En effet, pour une formule FND ϕ donnée, une instance de ce problème de déduction est la question : la formule FNC sans clause, (), implique-t-elle ϕ ? Or, il est facile de voir que ce problème est exactement le problème consistant à décider si ϕ est une tautologie ; en revanche, le problème est bien dans coNP, puisqu'une affectation aux variables satisfaisant la formule qui représente la base de connaissances, mais pas celle qui représente la requête, est un témoin (négatif) convenable. Le résultat pour les classes de formules qui nous intéressent découle ensuite du fait que la formule FNC sans clauses fait partie de toutes ces classes, et que vue comme une conjonction vide elle est également affine.

Classe $\mathcal{C}_\phi$	Classe $\mathcal{C}_\alpha$	Deduction[$\mathcal{C}_\phi$,$\mathcal{C}_\alpha$]
Horn-Ren	Fnc	traitable
Affine	Fnc	traitable
Horn-Ren-FND	Fnc	traitable
DisjAffine	Fnc	traitable
Horn, HDP, Nég et duales	Fnd	coNP-complet
Bij, Affine	Fnd	coNP-complet
Horn-Ren, HDP-Ren, Monot	Fnd	coNP-complet
Fnd	Horn-Ren	traitable
Fnd	Affine	traitable
Fnd	Horn-Ren-FND	traitable
Fnd	DisjAffine	traitable

Tab. 10.1 – Résumé des complexités des problèmes de déduction

Proposition 10.2 (Deduction[$\mathcal{C}_\phi$,Fnd]) *Le problème* Deduction[C_ϕ,Fnd] *est coNP-complet pour toutes les classes de formules présentées dans la partie I.*

Quant au problème de la déduction avec des bases de connaissances en FND, il se ramène à la déduction à partir de *termes*, puisque pour toutes formules ϕ_1, ϕ_2, α, on a $\phi_1 \vee \phi_2 \models \alpha$ si et seulement si on a $\phi_1 \models \alpha$ et $\phi_2 \models \alpha$. Or, par contraposition décider $\phi \models \alpha$ revient à décider $\neg\alpha \models \neg\phi$; de fait, lorsque ϕ est un terme le problème revient à un problème de déduction de *clauses*.

Nous synthétisons toutes ces remarques dans la table 10.1. Rappelons que pour les problèmes de raisonnement, si une classe $\mathcal{C}$ de formules (celle de la base de connaissances) admet un algorithme polynomial pour une certaine classe de requêtes, alors toute sous-classe de $\mathcal{C}$ admet un algorithme de même complexité (le même !) pour ces requêtes ; nous ne reportons donc dans la table 10.1, et dans les autres résumés de cette partie, que les résultats pour les classes de formules *maximales* pour l'inclusion de classes.

Enfin, pour la déduction et l'abduction nous avons choisi de ne pas détailler la complexité *exacte* des problèmes ; nous nous intéressons simplement à leur *traitabilité*, le problème le plus intéressant étant de déterminer des restrictions polynomiales. Néammoins, pour les deux problèmes les restrictions polynomiales que nous discutons admettent des complexités de l'ordre du temps linéaire ou quadratique.

Nous étudions donc dans cette partie la problématique de l'abduction avec un point de vue exact (chapitre 11), puis les problématiques de vérification de cohérence, de déduction et d'abduction avec un point de vue «approximation» (chapitre 12). Notons que ce dernier chapitre s'intéresse à la *sémantique* des processus de raisonnement, et non plus à leur *algorithmique*.

Encore une fois, les résultats présentés dans le corps des chapitres sont pour la plupart des contributions originales de cette thèse, et dans le cas contraire nous mentionnons l'origine des travaux rapportés.

Chapitre 11

Abduction

Sommaire

11.1 Présentation et travaux précédents 143
11.1.1 Formalisation des notions 144
11.1.2 Motivations 146
11.1.3 Travaux précédents 146
11.2 Modèle général d'algorithme 148
11.2.1 Projection 148
11.2.2 Modèle d'algorithme 150
11.3 Classes polynomiales 152
11.3.1 Formules FND 153
11.3.2 Formules affines 157
11.3.3 Classes retrouvées 158
11.4 Discussion et résumé des résultats connus 159

Nous nous intéressons dans ce chapitre à l'*abduction*, c'est-à-dire à la recherche d'une explication pour un «but» donné. Ce processus permet de formaliser de nombreux problèmes de raisonnement, comme la recherche de causes à des faits observés, de moyens d'atteindre un objectif etc. ; le raisonnement «(J'observe que) je suis, donc (il est plausible que) je pense», par exemple, peut être vu comme un raisonnement abductif. Nous rappelons les résultats déjà connus concernant l'algorithmique de ce processus, puis nous donnons un modèle d'algorithme général pour résoudre le problème ; son principe est basé sur la notion de *projection* d'une formule sur un ensemble de littéraux. Nous étudions alors des algorithmes dérivés de ce modèle pour le cas où la base de connaissances de référence est représentée par une formule affine, puis pour les cas où elle est représentée par une formule en FND ; nous découvrons ainsi de nouvelles classes traitables pour ce problème, et complétons donc l'état de l'art sur le sujet pour les classes de formules qui nous intéressent.

11.1 Présentation et travaux précédents

L'abduction est un processus de raisonnement avec sortie, qui fait intervenir trois données : une base de connaissances, un but et un ensemble d'hypothèses. Nous formalisons tout d'abord ces notions, ainsi que la notion d'*explication*, puis nous motivons l'étude de ce problème, et enfin nous présentons un rapide état de l'art sur le sujet.

11.1.1 Formalisation des notions

Hypothèses et explications Nous avons vu qu'un problème d'abduction faisait intervenir une *base de connaissances* et un *but*. Mais comme nous l'avons dit, on restreint également, en général, l'ensemble d'*hypothèses* parmi lesquelles l'explication doit être recherchée ; suivant le modèle classique, nous imposons que les explications soient des conjonctions de littéraux (des termes), formées sur un ensemble de littéraux distingués, l'ensemble des littéraux *abductibles*. La *requête* d'un problème d'abduction est donc constituée de ces deux objets, le but et l'ensemble des littéraux abductibles. La recherche d'une explication est enfin guidée par un critère de préférence entre les explications, relativement auquel on doit rechercher une *meilleure* explication ; nous choisissons le critère le plus classique, celui privilégiant les explications logiquement les plus faibles, c'est-à-dire les termes minimaux pour l'inclusion.

Toutes ces notions sont formalisées dans notre cadre par les deux définitions suivantes. Nous définissons tout d'abord, afin de simplifier l'exposé, la notion de *problème d'abduction*. Comme pour la déduction, cette notion dépend de deux classes de formules $\mathcal{C}_\phi$ et $\mathcal{C}_\alpha$, celles de la base de connaissances et celle du but ; notons que nous demandons que les formules représentant ces deux objets soient satisfaisables, et que le problème est trivial dans le cas contraire.

Définition 11.1 (problème d'abduction pour $\mathcal{C}_\phi$ et $\mathcal{C}_\alpha$) *Soient ϕ et α deux formules propositionnelles satisfaisables avec $\phi \in \mathcal{C}_\phi$ et $\alpha \in \mathcal{C}_\alpha$, et soit L un ensemble de littéraux. Le triplet (ϕ, α, L) est appelé un* problème d'abduction pour $\mathcal{C}_\phi$ et $\mathcal{C}_\alpha$.

Définition 11.2 (hypothèse, explication) *Soit $\Pi = (\phi, \alpha, L)$ un problème d'abduction pour deux classes de formules quelconques. Un terme $E \subseteq L$ est appelé une* hypothèse *pour Π, et une hypothèse E pour Π est appelée une* explication *pour Π si on a $\phi \wedge E \models \alpha$ (E* justifie *α sachant ϕ) et si $\phi \wedge E$ est satisfaisable (E est* cohérent *avec ϕ). Si de plus, aucun sous-terme propre de E n'est une explication pour Π, E est appelé une explication* minimale *pour Π.*

Exemples Afin d'illustrer ces notions, considérons l'exemple du carrefour et la question «Comment être sûr qu'on a le droit de traverser, en ne regardant que les feux ?». Cette question peut être formalisée dans notre cadre en définissant le problème d'abduction $\Pi_C = (\phi_C, \alpha_C, L_C)$ suivant. La base de connaissances de ce problème, tout d'abord, est la formule ϕ_C introduite dans les préliminaires généraux (paragraphe 3.2). Le but α_C, ensuite, est la formule réduite à une variable x_{DT}. Enfin, l'ensemble L_C des littéraux abductibles est l'ensemble $\{x_{VP}, \neg x_{VP}, x_{VV}, \neg x_{VV}\}$, c'est-à-dire tous les littéraux représentant les couleurs des feux. Les *hypothèses* pour ce problème sont donc les neuf termes :

$$(), (x_{VP}), (\neg x_{VP}), (x_{VV}), (\neg x_{VV}), (x_{VP} \wedge x_{VV}), (x_{VP} \wedge \neg x_{VV}), (\neg x_{VP} \wedge x_{VV}), (\neg x_{VP} \wedge \neg x_{VV})$$

En effet, puisque nous définissons les hypothèses comme des *termes*, les conjonctions de littéraux telles que $(x_{VP} \wedge \neg x_{VP})$ ne sont pas des hypothèses ; remarquons que cela n'est pas une restriction, puisque ces hypothèses ne peuvent être cohérentes avec la base de connaissances (elles sont insatisfaisables). Parmi les hypothèses pour Π_C, on voit facilement que les deux seules explications sont :

$$(x_{VP}), (x_{VP} \wedge \neg x_{VV})$$

En effet, pour le deuxième terme, par exemple, la formule $\phi_C \wedge x_{VP} \wedge \neg x_{VV}$ est bien satisfaisable, et implique le but x_{DT}. A l'inverse, l'hypothèse $(\neg x_{VP})$ n'est pas une explication pour Π_C car on n'a

pas $\phi_C \wedge \neg x_{VP} \models x_{DT}$, comme le montre l'affectation 0010 aux variables $\{x_{PV}, x_{VP}, x_{VV}, x_{DT}\}$, et l'hypothèse $(x_{VP} \wedge x_{VV})$ n'est pas une explication car elle n'est pas cohérente avec ϕ_C. Finalement, la seule explication minimale pour Π_C est (x_{VP}) («Il suffit de vérifier que le feu pour piétons est vert»), puisque l'autre explication en est un sur-terme.

Pour ce qui est de l'exemple de la thèse, la question «Comment faire pour que la lecture de ma thèse soit mémorable sans travailler ?» peut également être formalisée dans notre cadre. On peut en effet la formaliser par le problème d'abduction $\Pi_t = (\phi_t, \alpha_t, L_t)$ où : ϕ_t est la base de connaissances introduite dans les préliminaires généraux (paragraphe 3.3) ; α_t est la formule réduite à la variable x_{me} ; L_t est l'ensemble de littéraux $\{x_{ba}, x_{ch}, \neg x_{ch}\}$. Intuitivement, l'ensemble des abductibles peut être vu comme l'ensemble des littéraux qui représentent des actions possibles : ainsi, le doctorant ne veut pas se retrouver forcé à travailler, donc il considère comme abductible le littéral x_{ba} («Bâcler le travail»), mais pas son opposé ; en revanche, il veut bien avoir à choisir un lecteur qui soit chercheur en informatique, ou un lecteur qui ne le soit pas, et il considère donc comme abductibles les littéraux x_{ch} et $\neg x_{ch}$; enfin, les autres variables, intuitivement, représentent des faits que l'on ne peut pas réaliser directement. De fait, les hypothèses pour Π_t sont les termes :

$$(), (x_{ba}), (x_{ch}), (\neg x_{ch}), (x_{ba} \wedge x_{ch}), (x_{ba} \wedge \neg x_{ch})$$

parmi lesquels seul le terme $(x_{ba} \wedge \neg x_{ch})$ est une explication pour Π_t. Le terme $(\neg x_{ba} \wedge x_{ch})$, quant à lui, explique bien la requête α_t, mais il n'est pas formé sur l'ensemble de littéraux abductibles L_t, et n'est donc pas une explication pour Π_t.

Problèmes algorithmiques Nous pouvons maintenant énoncer le problème avec sortie que nous traitons dans ce chapitre. Encore une fois, nous paramétrons ce problème par deux classes de formules, $\mathcal{C}_\phi$ et $\mathcal{C}_\alpha$.

Problème 11.1 (`Abduction[`$\mathcal{C}_\phi$`,`$\mathcal{C}_\alpha$`]`)

Donnée : *Un problème d'abduction* Π *pour* $\mathcal{C}_\phi$ *et* $\mathcal{C}_\alpha$
Sortie : *Une explication minimale pour* Π *s'il en existe une, et sinon «Pas d'explication».*

Cette formalisation de la problématique est à la fois classique et générale. En effet, nous n'imposons aucune restriction sur la classe de formules C_α dans laquelle sont représentés les buts, ni sur les ensembles de littéraux abductibles. Notons également que nous autorisons un littéral abductible à être formé sur une variable apparaissant dans la requête. Outre la généralité qu'elle permet, cette possibilité peut être justifiée en pratique. Ainsi, si l'on considère dans le petit monde de la thèse la question «Comment caractériser les thèses à la fois mémorables et de qualité décrites dans une base de données de rapporteurs ?», on peut la formaliser par le problème d'abduction $(\phi_t, (x_{me} \wedge x_{qu}), \{x_{ch}, \neg x_{ch}\})$. Mais ce problème n'a pas d'explication. En revanche, si l'on ajoute les littéraux abductibles x_{me} et $\neg x_{me}$ (intuitivement, on s'autorise à demander aux rapporteurs s'ils se souviennent de la thèse), on peut obtenir l'explication $(x_{me} \wedge x_{ch})$ («Ce sont celles qui ont été rapportées par un chercheur en informatique qui s'en souvient»). Le littéral x_{me}, ici, non seulement participe à expliquer x_{me} dans le but, mais il participe également à expliquer x_{qu}, en ce sens que ϕ_t et x_{ch} seuls ne suffisent pas à l'expliquer.

Nous nous intéressons donc à la complexité du problème `Abduction[`$\mathcal{C}_\phi$`,`C_α`]` pour les classes $\mathcal{C}_\phi$ présentées dans la partie I, en recherchant les classes $\mathcal{C}_\alpha$ maximales pour lesquelles il est

traitable. Suivant la littérature, nous calculons la complexité d'un algorithme traitant ce problème en fonction de la taille de la base de connaissances ϕ et du but α, ainsi que du nombre $|L|$ d'*abductibles* (nous ne considérons pas que l'ensemble des *hypothèses* fait partie de la donnée du problème). Pour montrer qu'un tel problème est difficile, nous considérons le problème de décision associé suivant.

Problème 11.2 (`D-Abduction[`$\mathcal{C}_\phi$`,`$\mathcal{C}_\alpha$`]`)

Donnée : *Un problème d'abduction* Π *pour* C_ϕ *et* C_α
Question : *Existe-t-il au moins une explication pour* Π *?*

Clairement, si cette question est difficile, alors le *calcul* d'une explication l'est au moins autant.

11.1.2 Motivations

Comme nous l'avons vu dans le chapitre 10, la problématique de l'abduction permet de formaliser de nombreux problèmes de raisonnement. Si l'on voit le but comme un fait inattendu que l'on a observé, ce processus formalise alors la recherche d'une explication plausible de ce fait. On peut ainsi formaliser le problème général du *diagnostic*, qui survient dans le domaine médical (diagnostic de maladies, le but formalisant les symptômes observés), dans le domaine de l'électronique (diagnostic de pannes, les explications formalisant des ensembles de composants possiblement défectueux) [CMM98, SW01], etc. Si l'on voit le but comme un fait dont on veut s'assurer, ce problème formalise alors des problèmes de *vérification*, où il s'agit de déterminer ce qu'il suffit de vérifier pour s'assurer d'un fait donné. Si l'on voit le but comme un objectif à atteindre, le processus d'abduction peut encore formaliser des problèmes de *planification* [HLM01] (avec des restrictions). Ce processus, enfin, est au centre des systèmes de raisonnement proposés par de Kleer, les CMS («Clause Management Systems») et ATMS («Assumption-Based Truth Maintenance Systems») [dK86, RdK87].

Plus concrètement, l'abduction est également une problématique centrale dans des problèmes de *configuration* : par exemple, dans le cas d'un client d'une entreprise, qui commande un produit «sur mesure» en indiquant les composants qui l'intéressent un par un, des problèmes d'abduction surgissent lorsqu'il faut lui expliquer lesquels de ses choix précédents l'empêchent de choisir tel nouveau composant, afin par exemple de lui permettre de revenir sur ces choix («restauration de la cohérence») ; Amilhastre *et al.* [AFM02, paragraphe 5] décrivent une telle application en détail, pour la configuration de voitures. De façon plus élaborée, l'abduction a été utilisée pour formaliser la *compréhension de langage naturel* [HSAM93] : il s'agit d'expliquer la structure logique d'une phrase à partir d'une base de connaissances concernant le sens commun («Si j'habite dans une ville, alors j'habite dans le pays où se situe cette ville», par exemple) et les règles du langage. Les abductibles sont alors, par exemple, l'identification d'un pronom avec un nom cité dans le texte. Cette application est décrite en détail par les auteurs [HSAM93].

11.1.3 Travaux précédents

Résultats généraux Le travail d'Eiter et Gottlob [EG95a] donne les principaux résultats généraux concernant la complexité des problèmes d'abduction formalisés dans un cadre proche du nôtre. Ils montrent notamment que le problème `D-Abduction[`FNC`,`LITS`]` est très difficile, et plus précisément qu'il est Σ_2^P-*complet* ; rappelons que la classe Σ_2^P correspond intuitivement à la classe des problèmes pour lesquels la vérification d'un témoin est un problème de NP [Pap94,

chapitre 17]. Concernant encore les résultats généraux, il est facile de voir que le problème «descend» dans NP lorsque le problème SAT et la déduction sont traitables pour les classes de formules considérées, puisque alors un témoin (l'explication) est vérifiable en temps polynomial.

Amilhastre *et al.* [AFM02] donnent également des résultats généraux, pour des bases de connaissances formées sur des variables *multivaluées* (cadre des Problèmes de Satisfaction de Contraintes) ; leurs résultats sont essentiellement les mêmes que dans le cas propositionnel.

L'article de Bylander *et al.* [BATJ89] traite de nombreuses formalisations différentes pour l'abduction, et notamment différents critères de préférence entre les hypothèses. Les auteurs montrent également les liens entre l'abduction et d'autres problèmes de raisonnement, et donnent un historique assez riche de l'étude de l'abduction. Dans un cadre plus précis, l'article d'Eiter et Gottlob [EG95a] traite également ces aspects. Pour le lien entre l'abduction et d'autres problèmes, on pourra également consulter l'article de Marquis sur les impliqués premiers et le «consequence-finding» [Mar00].

Classes de formules Concernant des classes plus précises de formules, citons notamment le travail de Selman et Levesque [SL90] sur la complexité du problème D-Abduction[HORN,VARS]. La remarque précédente montre que ce problème est dans NP si la requête est en FNC (il faut que la déduction soit traitable), mais les auteurs montrent qu'il est NP-*complet*, et ce même si on se restreint aux formules FNC de Horn *acycliques* (nous renvoyons à l'article pour les définitions) et si la requête est une variable. Ils montrent cependant qu'en se restreignant au cas où tous les littéraux formés sur des variables qui n'apparaissent pas dans la requête sont abductibles, on obtient un problème traitable ; ce résultat, s'il n'est pas très intéressant en pratique, leur permet d'identifier le sous-problème de «restriction» à un ensemble de variables («support set selection») comme le cœur du problème. Pour obtenir des problèmes traitables avec des formules de Horn, il faut se restreindre au cas des formules FNC de Horn *définies positives*, avec des buts donnés par des termes positifs [EG95a, proposition 5.3]. Remarquons que la NP-complétude du problème pour la classe des formules de Horn dans toute sa généralité implique sa NP-difficulté pour la classe plus générale des formules Horn-renommables, et donc sa NP-complétude, les problèmes de satisfaisabilité et de déduction restant polynomiaux.

En ce qui concerne les autres classes de formules qui nous intéressent dans ce mémoire, les bases de connaissances représentées par des formules 2FNC ou FNC monotones (incluant les cas négatif et positif) permettent une abduction polynomiale pour des classes de requêtes naturelles. Ces cas sont notamment traités par Marquis [Mar00] dans un cadre plus général, avec la notion d'impliqué premier.

Mais le problème d'abduction a également été montré polynomial pour des classes de formules FNC définies en *structure*. C'est le cas pour les formules FNC de Horn acycliques *dont le problème* SAT *pour la pseudo-complétion peut être traité par la résolution unitaire seule*, lorsque le but est donné par une variable [Esh93]. C'est également le cas pour les formules *dont la structure a une largeur de noyau induite bornée* [dV00a] (voir aussi le travail expérimental correspondant [SdV01]) ; encore une fois, nous renvoyons aux articles pour les définitions. Enfin, même si cette représentation n'est pas une formule de la logique propositionnelle, il est intéressant de noter que l'abduction est traitable pour des bases de connaissances représentées par leurs modèles caractéristiques [KKS95, KR96] (avec certaines restrictions, pour lesquelles nous renvoyons toujours aux articles).

Enfin, un article récent d'Eiter et Makino [EM02] traite le problème du calcul de *plusieurs* explications pour un même problème. A notre connaissance, c'est le premier réel travail dans ce sens, et les auteurs donnent de premiers résultats pour les bases de connaissances de Horn.

Notre apport Dans ce chapitre, nous complétons l'état de l'art pour le problème de l'abduction tel que nous l'avons formalisé. Nous donnons un modèle d'algorithme pour ce problème, général et indépendant des classes de formules considérées ; ce modèle est basé sur la notion de *projection* d'une formule sur un ensemble de littéraux. Puis nous étudions les classes de formules pour lesquelles ce modèle donne des algorithmes polynomiaux, et montrons ainsi que l'abduction est traitable pour des bases de connaissances affines, ou représentées dans les classes de formules FND qui nous intéressent [Zan02c, Zan03b]. Nous montrons également que ce modèle unifie plusieurs résultats de traitabilité déjà connus.

11.2 Modèle général d'algorithme

Nous présentons dans ce paragraphe le modèle d'algorithme que nous proposons pour résoudre un problème d'abduction. Ce modèle étant basé sur la notion de *projection* d'une formule sur un ensemble de littéraux, nous commençons toutefois par présenter cette notion.

11.2.1 Projection

L'opération consistant à *projeter* une formule sur un ensemble de littéraux est présentée dans [Mar00] sous l'appellation «oubli de littéraux». Intuitivement, si l'on considère les ensembles de modèles, elle généralise l'opération de projection d'une base de données relationnelle sur un sous-ensemble de ses attributs.

Nous définissons tout d'abord cette opération de manière *sémantique*, c'est-à-dire en termes d'ensembles de modèles ; néanmoins, nous l'appliquerons bien à des formules, et nous discuterons le calcul associé dans le paragraphe 11.3. Notre définition différant de celle de [Mar00], nous montrons ensuite l'équivalence entre les deux.

Définitions Nous définissons tout d'abord la notion de *projeté* d'une affectation. Soient V un ensemble de variables et L un ensemble de littéraux formés sur V. Rappelons qu'une *affectation partielle* à V est une application de V dans $\{0, 1, ?\}$. Si m est une affectation à V et L un ensemble de littéraux (non nécessairement tous formés sur V), on appelle *projeté de m sur L*, et on note $m_{|L}$, l'affectation partielle à V définie pour toute variable $x \in V$ par :

$$\begin{cases} m_{|L}[x] = 0 \text{ si } m[x] = 0 \text{ et } \neg x \in L \\ m_{|L}[x] = 1 \text{ si } m[x] = 1 \text{ et } x \in L \\ m_{|L}[x] =? \text{ sinon} \end{cases}$$

Ainsi, si l'on considère l'exemple de la thèse, le projeté de l'affectation $00111 \in M_t$ à $V_t = \{x_{me}, x_{qu}, x_{ba}, x_{in}, x_{ch}\}$ sur l'ensemble de littéraux $L'_t = \{\neg x_{me}, x_{ba}, x_{ch}, \neg x_{ch}\}$ est l'affectation partielle 0?1?1 à V_t ; quant au projeté de 01000 sur L'_t, c'est l'affectation partielle 0????0.

Nous pouvons alors définir la notion de *projeté* d'une formule sur un ensemble de littéraux. Rappelons des préliminaires généraux (paragraphe 1.1) que l'*extension* $ext(p)$ d'une affectation partielle p à un ensemble de variables V est l'ensemble des affectations m à V telles que l'on a :

$$\forall x \in V, p[x] \neq ? \Longrightarrow m[x] = p[x]$$

Définition 11.3 (projeté) *Soient ϕ une formule propositionnelle et L un ensemble de littéraux. On appelle* projeté de ϕ sur L *toute formule propositionnelle ϕ' telle que l'ensemble des affectations à $Var(\phi)$ satisfaisant ϕ' est $\bigcup_{m \in \mathcal{M}(\phi)} ext(m_{|L})$.*

Ainsi, une formule propositionnelle admet en général plusieurs projetés sur un même ensemble de littéraux, mais tous sont logiquement équivalents les uns aux autres. Si l'on reprend l'exemple de la thèse, l'ensemble des modèles d'un projeté de ϕ_t sur L'_t est donc l'ensemble :

$$ext(00111_{|L'_t}) \cup ext(01000_{|L'_t}) \cup ext(10100_{|L'_t}) \cup ext(11011_{|L'_t})$$

c'est-à-dire l'ensemble :

$$ext(0?1?1) \cup ext(0???0) \cup ext(??1?0) \cup ext(????1) = \{0,1\}^{V_t} \backslash \{10000, 10010, 11000, 11010\}$$

De fait, un projeté de ϕ_t sur L_t est par exemple la formule FNC réduite à une clause :

$$(\neg x_{me} \vee x_{ba} \vee x_{ch})$$

Propriétés de la projection Il est utile de lister dès maintenant quelques propriétés immédiates de l'opération de projection. Tout d'abord, la projection est distributive sur la disjonction, autrement dit, si ϕ_1 et ϕ_2 sont deux formules, si L est un ensemble de littéraux et si ϕ'_1 (resp. ϕ'_2) est un projeté de ϕ_1 (resp. ϕ_2) sur L, alors la formule $\phi'_1 \vee \phi'_2$ est un projeté de la formule $\phi_1 \vee \phi_2$ sur L. En revanche, dans le cas général la projection n'est pas distributive sur la conjonction ; elle ne l'est que si ϕ_1 et ϕ_2 dépendent réellement d'ensembles de variables disjoints (on dit qu'une formule ϕ *dépend réellement* de l'ensemble de variables V si V est l'(unique) ensemble minimal pour l'inclusion tel que tout projeté de ϕ sur $V \cup \{\neg x \mid x \in V\}$ est logiquement équivalent à ϕ). Enfin, en règle générale la projection n'est pas distributive sur la négation.

Equivalence avec la notion d'oubli Nous montrons enfin que les projetés d'une formule ϕ sur un ensemble de littéraux L sont logiquement équivalents à la formule obtenue de ϕ en oubliant les littéraux formés sur $Var(\phi)$ qui ne sont pas dans L. On peut également trouver cette équivalence dans le récent article de Lang *et al.* [LLM03, proposition 15].

Cette notion d'oubli est définie inductivement comme suit.

Définition 11.4 (oubli de littéraux, [Mar00, définition 3.14]) *Soient ϕ une formule propositionnelle et L un ensemble de littéraux formés sur $Var(\phi)$. La formule $fg(\phi, L)$ est la formule définie inductivement par les règles suivantes :*

$$\begin{cases} fg(\phi, L) = \phi \text{ si } L = \emptyset \\ fg(\phi, L) = \phi[\{x\} : 1] \vee (\neg x \wedge \phi[\{x\} : 0]) \text{ si } L = \{x\} \text{ pour une variable } x \in Var(\phi) \\ fg(\phi, L) = \phi[\{x\} : 0] \vee (x \wedge \phi[\{x\} : 1]) \text{ si } L = \{\neg x\} \text{ pour une variable } x \in Var(\phi) \\ fg(\phi, L) = fg(fg(\phi, L'), \{\ell\}) \text{ si } L = L' \cup \{\ell\} \text{ pour un littéral } \ell \text{ formé sur } Var(\phi) \end{cases}$$

Nous sommes donc en mesure de montrer l'équivalence suivante entre les deux notions. Intuitivement, projeter une formule sur un ensemble de littéraux revient à oublier tous les autres.

Proposition 11.1 (projection vs. oubli) *Soient ϕ une formule propositionnelle et L un ensemble de littéraux formés sur $Var(\phi)$. Soit L_ϕ l'ensemble de tous les littéraux formés sur $Var(\phi)$. Alors pour tout projeté ϕ' de ϕ sur L on a $\phi' \equiv fg(\phi, L_\phi \backslash L)$.*

Preuve Nous montrons le résultat par induction sur $L_\phi \backslash L$. Si $L = L_\phi$, tout d'abord, on a $\forall m \in \mathcal{M}(\phi), m_{|L} = m$ et donc pour tout projeté ϕ' de ϕ, on a $\phi' \equiv \phi$; or on a $fg(\phi, L_\phi \backslash L) = fg(\phi, \emptyset) = \phi$.

Supposons ensuite $L = L_\phi \backslash \{x\}$ pour une variable $x \in Var(\phi)$, et soit ϕ' un projeté de ϕ sur L. Si m est une affectation à $Var(\phi)$ satisfaisant ϕ', alors il existe $m' \in \mathcal{M}(\phi)$ avec $m \in ext(m'_{|L})$; si on a $m'[x] = 1$ alors m' satisfait $\phi[\{x\} : 1]$ et on a $\forall x' \in Var(\phi), x' \neq x \Longrightarrow m[x] = m'[x]$ par définition du projeté d'une affectation, donc m satisfait $\phi[\{x\} : 1]$; si à l'inverse on a $m'[x] = 0$, alors on a $m = m'$ puisque $\neg x$ est dans L, donc m satisfait $\neg x \wedge \phi[\{x\} : 0]$. Finalement, m satisfait $\phi[\{x\} : 1]$ ou $\neg x \wedge \phi[\{x\} : 0]$, donc satisfait $fg(L, \phi)$. Réciproquement, soit m une affectation à $Var(\phi)$ satisfaisant $fg(L, \phi)$; si m satisfait $\phi[x : 1]$, alors l'affectation m' à $Var(\phi)$ définie par $m'[x] = 1$ et $\forall x' \in Var(\phi), x' \neq x \Longrightarrow m'[x'] = m[x']$ vérifie $m'_{|L} = m$ (puisque l'on a $x \notin L$) et $m' \models \phi$ (puisque l'on a $m \models \phi[\{x\} : 1]$) ; si à l'inverse m satisfait $\neg x \wedge \phi[\{x\} : 0]$, alors m satisfait ϕ, donc *a fortiori* ϕ'. Finalement, m satisfait ϕ'.

Le cas $L = \{\neg x\}$, pour une variable $x \in Var(\phi)$, est symétrique. Enfin, le cas $L = L_\phi \backslash (L' \cup \{\ell\})$, pour un littéral $\ell \in L_\phi$, se montre de même que le cas $L = L_\phi \backslash \{\ell\}$. □

Notons que toutes les notions présentées dans ce paragraphe s'étendent aisément au cas où l'ensemble de littéraux sur lequel on projette une formule ϕ contient un littéral formé sur une variable $x \notin Var(\phi)$: il suffit alors d'ignorer ce littéral.

11.2.2 Modèle d'algorithme

Nous donnons maintenant le principe de notre modèle d'algorithme. Celui-ci a notamment pour but de mettre en évidence les calculs difficiles, indépendamment des classes de formules.

Etant donné un problème d'abduction $\Pi = (\phi, \alpha, L)$ pour deux classes de formules quelconques, nous caractérisons tout d'abord les explications pour Π en fonction des modèles de ϕ et de ceux de la formule $\phi \wedge \neg\alpha$. Afin de simplifier l'exposé, pour un ensemble de variables V nous définissons le *terme associé* à une affectation partielle p à V, noté $T(p)$ (rappelons que nous voyons les termes comme des ensembles de littéraux) :

$$T(p) = \{x \in V \mid p[x] = 1\} \cup \{\neg x \mid x \in V \text{ et } p[x] = 0\}$$

Ainsi, le terme associé à l'affectation partielle 0?1?1 de l'exemple de la thèse est $T(0?1?1) = (\neg x_{me} \wedge x_{ba} \wedge x_{ch})$.

Lemme 11.1 *Soit $\Pi = (\phi, \alpha, L)$ un problème d'abduction pour deux classes de formules quelconques, et soit ϕ' un projeté de la formule $\phi \wedge \neg\alpha$ sur L. Alors pour toute explication E pour Π, il existe une affectation m à $Var(\phi) \cup Var(\alpha)$ telle que :*

- *m satisfait la formule $\phi \wedge \neg\phi'$*
- *E est un sous-terme du terme associé au projeté $m_{|L}$ de m sur L.*

Réciproquement, si m est une affectation à $Var(\phi) \cup Var(\alpha)$ satisfaisant $\phi \wedge \neg\phi'$, alors le terme $T(m_{|L})$ est une explication pour Π.

Preuve Dans toute la preuve, ϕ' désigne un projeté de la formule $\phi \wedge \neg\alpha$ sur L, fixé.

[sens direct] Soit E une explication pour Π, et soit m une affectation à $Var(\phi) \cup Var(\alpha)$ satisfaisant $\phi \wedge E \wedge \alpha$; par définition d'une explication une telle affectation existe bien. En

particulier m satisfait E, donc E est bien un sous-terme de $T(m_{|L})$, et m satisfait ϕ. Il ne reste donc qu'à montrer que m satisfait $\neg\phi'$, c'est-à-dire que m ne satisfait pas ϕ'. Mais puisque E est une explication pour Π, on a $\phi \wedge E \models \alpha$, donc la formule $\phi \wedge E \wedge \neg\alpha$ est insatisfaisable ; en particulier, aucune affectation m' à $Var(\phi) \cup Var(\alpha)$ avec $m' \models \phi \wedge \neg\alpha$ ne vérifie $E \subseteq T(m')$, et donc aucune affectation m' à $Var(\phi) \cup Var(\alpha)$ qui satisfait $\phi \wedge \neg\alpha$ ne vérifie $m_{|L} = m'_{|L}$, donc par définition de la projection m ne satisfait pas $(\phi \wedge \neg\alpha)_{|L}$.

[réciproque] Réciproquement, soit m une affectation à $Var(\phi) \cup Var(\alpha)$ satisfaisant $\phi \wedge \neg\phi'$, et notons E le terme $T(m_{|L})$. Puisque m satisfait $\phi \wedge \neg\phi'$, la formule $\phi \wedge E$ est satisfaisable. D'autre part, puisque m ne satisfait pas ϕ', il n'existe aucune affectation m' à $Var(\phi) \cup Var(\alpha)$ avec $m' \models \phi \wedge \neg\alpha$ et $m_{|L} = m'_{|L}$, donc aucun m' satisfaisant à la fois $\phi \wedge \neg\alpha$ et E, et on en déduit que la formule $\phi \wedge \neg\alpha \wedge E$ est insatisfaisable, c'est-à-dire que la formule $\phi \wedge E$ implique α. □

A titre d'illustration, considérons à nouveau l'exemple du carrefour, et le problème d'abduction $\Pi_C = (\phi_C, \alpha_C, L_C)$ introduit dans le paragraphe 11.1.1, où le but α_C est réduit à la variable x_{DT} et avec $L_C = \{x_{VP}, \neg x_{VP}, x_{VV}, \neg x_{VV}\}$. Nous avons vu que les explications pour Π_C étaient les termes (x_{VP}) et $(x_{VP} \wedge \neg x_{VV})$. Rappelons également que l'ensemble des modèles de ϕ_C est l'ensemble d'affectations à $V_C = \{x_{PV}, x_{VP}, x_{VV}, x_{DT}\}$:

$$M_C = \{0000, 0010, 0101, 1000, 1010, 1101\}$$

De fait, l'ensemble des modèles de la formule $\phi_C \wedge \neg\alpha_C = \phi_C \wedge \neg x_{DT}$ est l'ensemble :

$$\{0000, 0010, 1000, 1010\}$$

L'ensemble des projetés sur L_C des affectations de cet ensemble est donc l'ensemble d'affectations partielles à V_C $\{?00?, ?01?\}$; intuitivement, ce sont donc les affectations à $\{x_{VP}, x_{VV}\}$ qui sont cohérentes avec $\phi_C \wedge \neg\alpha_C$, et de fait les termes associés ne peuvent expliquer α_C. Ces «non-explications» sont donc résumées par l'un quelconque des projetés de $\phi_C \wedge \neg\alpha$ sur L_C, par exemple la formule réduite à un littéral $\neg x_{VP}$, satisfaite exactement par les extensions des affectations généralisées ci-dessus. On remarque alors que l'affectation 0101 à V_C satisfait ϕ_C mais pas $\neg x_{VP}$, et on déduit donc du lemme 11.1 que le terme $T(0101_{|L_C}) = T(?10?) = (x_{VP} \wedge \neg x_{VV})$ est une explication pour Π_C.

Etant donné un problème d'abduction $\Pi = (\phi, \alpha, L)$, nous avons donc une caractérisation des explications pour Π, mais pas des explications *minimales* ; nous déléguons le problème de minimiser une explication E donnée pour Π à notre modèle d'algorithme, en lui intégrant la phase de minimisation suivante, qui reprend le principe glouton donné dans [SL90]. Soit E une explication pour Π.

$E' \leftarrow E$;
Pour tout littéral $\ell \in E'$ **faire**
  Si $\phi \wedge E' \backslash \{\ell\} \models \alpha$ **Alors** $E' \leftarrow E' \backslash \{\ell\}$ **Fin Si** ;
Fin Pour ;

Ainsi, si l'on reprend l'explication $(x_{VP} \wedge \neg x_{VV})$ calculée plus haut pour Π_C, cette phase teste tout d'abord si on a $\phi_C \wedge \neg x_{VV} \models x_{DT}$, ce qui est contredit par le modèle 0000 de ϕ_C, puis si on a $\phi_C \wedge x_{VP} \models x_{DT}$, ce qui est vrai ; l'explication minimisée est donc le terme (x_{VP}).

La correction de cette phase est évidente ; il suffit des remarques suivantes concernant l'hypothèse E' retournée par la procédure :

Entrée : Un problème d'abduction $\Pi = (\phi, \alpha, L)$ pour deux classes de formules quelconques
Sortie : Une explication minimale pour Π, s'il en existe une.

Début
  $\phi' \leftarrow$ un projeté de la formule $\phi \wedge \neg\alpha$ sur L ;
  $\phi'' \leftarrow$ une formule logiquement équivalente à la formule $\phi \wedge \neg\phi'$;
  Si ϕ'' est insatisfaisable **Alors**
    retourner «Pas d'explication» ;
  Sinon
    $m \leftarrow$ un modèle de ϕ'' ;
    $E \leftarrow T(m_{|L})$;
    minimiser E ;
  Fin Si ;
  retourner E ;
Fin

FIG. 11.1 – Algorithme `Expliquer` (modèle)

- E' étant un sous-terme de E et la formule $\phi \wedge E$ étant par hypothèse satisfaisable, $\phi \wedge E'$ est satisfaisable
- puisque la formule $\phi \wedge E$ implique α, par construction $\phi \wedge E'$ l'implique également
- si on avait $\phi \wedge E' \backslash \{\ell\} \models \alpha$ pour un certain littéral $\ell \in E'$, alors on aurait $\phi \wedge E'' \backslash \{\ell\} \models \alpha$ pour tout sur-ensemble E'' de E', donc ℓ aurait été supprimé de E' par l'algorithme ; donc E' est minimale.

Remarquons que, selon l'ordre dans lequel les littéraux de E sont considérés, l'explication minimale retournée peut être différente ; néanmoins toutes sont bien des explications minimales pour le problème d'abduction considéré.

Nous pouvons donc finalement donner le modèle d'algorithme de la figure 11.1, qui recherche une explication minimale pour un problème d'abduction donné. Sa correction est garantie par le lemme 11.1 et par la correction de la procédure de minimisation. Nous considérons ce modèle comme un algorithme pour donner le résultat, même s'il n'explicite pas tous les calculs.

Proposition 11.2 (`Expliquer`) *Soit Π un problème d'abduction pour deux classes de formules quelconques. Sur la donnée de Π, l'algorithme* `Expliquer` *retourne une explication minimale pour Π s'il en existe une, et retourne «Pas d'explication» sinon.*

11.3 Classes polynomiales

Nous étudions dans ce paragraphe les classes $\mathcal{C}_\phi$ et $\mathcal{C}_\alpha$ de formules telles que le problème `Abduction[`$\mathcal{C}_\phi$`,`$\mathcal{C}_\alpha$`]` admette un algorithme polynomial sur le modèle de `Expliquer`. L'opération de projection étant essentielle, notre étude s'appuie sur les classes $\mathcal{C}$ de formules telles que l'on peut calculer efficacement, pour toute formule $\phi \in \mathcal{C}$ et tout ensemble de littéraux L, un projeté ϕ' de ϕ sur L avec $\phi' \in \mathcal{C}$; remarquons que cela suppose en particulier qu'un tel projeté existe toujours. Rappelons que nous avons choisi de ne pas détailler la complexité des algorithmes, le but principal étant d'identifier des restrictions traitables des problèmes.

11.3.1 Formules FND

Nous nous intéressons tout d'abord aux classes de formules FND présentées dans la partie I. Remarquons tout d'abord que si le problème `SAT` est trivial pour la classe de toutes les formules FND, ce n'est pas le cas pour le problème d'abduction, qui est difficile pour cette classe même en se restreignant à des buts donnés par une variable et à des ensembles de littéraux abductibles très particuliers, comme le montre le résultat suivant (nouveau à notre connaissance).

Proposition 11.3 (FND) *Le problème* `D-Abduction[FND,VARS]` *est NP-difficile, même si l'on n'autorise que les problèmes de la forme* (ϕ, α, L) *où* L *est l'ensemble de tous les littéraux formés sur* $Var(\phi)\backslash Var(\alpha)$.

Preuve Nous réduisons le problème `SAT` pour les formules FNC à ce problème. Soit ϕ une formule FNC, et soit $x \in \mathbb{V}\backslash Var(\phi)$ une nouvelle variable. Nous montrons que ϕ est satisfaisable si et seulement si le problème d'abduction $\Pi = (Neg(\phi) \vee (x), x, Var(\phi) \cup \{\neg x' \mid x' \in Var(\phi)\})$ admet au moins une explication ; notons que la formule $Neg(\phi) \vee (x)$ est bien en FND, et peut être construite en temps linéaire en la taille de ϕ. Supposons tout d'abord ϕ insatisfaisable ; alors pour toute hypothèse E pour Π, E satisfait $Neg(\phi)$ et donc l'hypothèse $E \cup \{\neg x\}$ satisfait $Neg(\phi) \vee (x)$; on en déduit que E n'est pas une explication pour Π. Réciproquement, si ϕ est satisfaisable on peut trouver une affectation m à $Var(\phi)$ avec $m \not\models Neg(\phi)$. En considérant m comme une affectation partielle et en notant $E = T(m)$, on voit alors que la formule $(Neg(\phi) \vee (x)) \wedge E \wedge \neg x$ est insatisfaisable, donc que $Neg(\phi) \wedge E$ implique x, et que la formule $(Neg(\phi) \vee (x)) \wedge E$ est satisfaisable (affecter 1 à x) ; finalement, E est une explication pour Π. □

Il est donc important de déterminer des classes $\mathcal{C}_\phi$ de formules FND pour lesquelles ce problème devient traitable avec des classes C_α intéressantes ; nous montrons que c'est possible pour toutes les classes de formules FND présentées dans la partie I.

Projection Nous remarquons tout d'abord que, de façon générale, un projeté en FND d'une formule ϕ donnée en FND sur un ensemble L de littéraux peut être calculé en temps linéaire en la taille de ϕ, en supprimant simplement dans ϕ toutes les occurrences de littéraux qui ne sont pas dans L ; notons $GS(L, \phi)$ («Garder Seulement L dans ϕ») la formule ainsi obtenue.

Le résultat suivant, très simple, est notamment donné par Marquis [Mar00].

Lemme 11.2 *Soient* ϕ *une formule FND et* L *un ensemble de littéraux. Alors la formule FND* $GS(L, \phi)$ *est un projeté de* ϕ *sur* L.

Preuve Puisque la projection est distributive sur la disjonction, il suffit de montrer l'énoncé pour les formules réduites à un terme. Mais puisque tous les littéraux d'un terme sont formés sur des variables différentes, la projection est encore distributive sur la conjonction à l'intérieur d'un terme, et on a le résultat. □

Considérons par exemple la formule FND de l'exemple du carrefour :

$$\phi'_C = (\neg x_{PV} \wedge \neg x_{VP} \wedge \neg x_{DT}) \vee (\neg x_{VP} \wedge x_{VV} \wedge \neg x_{DT}) \vee (x_{VP} \wedge \neg x_{VV} \wedge x_{DT}) \vee (\neg x_{VP} \wedge \neg x_{VV} \wedge \neg x_{DT})$$

La formule FND suivante est donc l'un de ses projetés sur l'ensemble $\{x_{PV}, \neg x_{VP}, x_{DT}, \neg x_{DT}\}$ de littéraux :

$$(\neg x_{VP} \wedge \neg x_{DT}) \vee (x_{DT})$$

Abduction Nous pouvons maintenant donner un premier résultat pour l'abduction. Nous le formulons de façon générale, mais discutons ensuite les classes de formules FND qui nous intéressent et qui en satisfont les hypothèses.

Proposition 11.4 (classes de formules FND) *Soit $\mathcal{C}$ une classe de formules FND telle que :*
- *$\mathcal{C}$ est stable par propagation d'affectations*
- *pour toute formule $\phi \in \mathcal{C}$ et tout ensemble L de littéraux, la formule $GS(L, \phi)$ est dans $\mathcal{C}$*
- *le problème* `TAUT` *restreint aux formules de $\mathcal{C}$ est traitable.*

Alors le modèle d'algorithme `Expliquer` *fournit un algorithme polynomial pour le problème* `Abduction[`$\mathcal{C}$`,`CLAUSES`]`.

Preuve Nous détaillons l'exécution de l'algorithme donné par le modèle `Expliquer` dans ce cadre. Soit (ϕ, α, L) un problème d'abduction pour $\mathcal{C}$ et CLAUSES.

[projection] Tout d'abord, on peut calculer calculer la formule $\phi_1 = \phi \wedge Neg(\alpha)$, qui est logiquement équivalente à $\phi \wedge \neg\alpha$, en temps linéaire en les tailles de ϕ et de α. Cette formule est encore logiquement équivalente à la formule $\phi_2 = \phi'_2 \wedge Neg(\alpha)$, où ϕ'_2 est la formule obtenue en propageant dans ϕ l'affectation m à $Var(\alpha)$, définie pour toute variable $x \in Var(\alpha)$ par $m[x] = 0 \iff x \in \alpha$; cette formule peut toujours être calculée efficacement. Par construction on a de plus $Var(\phi'_2) \cap Var(Neg(\alpha)) = \emptyset$, et donc la formule $\phi' = GS(L, \phi'_2) \wedge GS(L, Neg(\alpha))$, qui peut être calculée efficacement, est un projeté de ϕ_2 sur L, c'est-à-dire, de $\phi \wedge \neg\alpha$ sur L. Enfin, on a par hypothèse $GS(L, \phi'_2) \in \mathcal{C}$.

[calcul d'une explication] Nous devons donc maintenant calculer une formule ϕ'' logiquement équivalente à la formule $\phi \wedge \neg\phi'$. Nous posons $\phi'' = \phi \wedge (Neg(GS(L, \phi'_2)) \vee Neg(GS(L, Neg(\alpha))))$. On a donc $\phi'' \equiv (\phi \wedge Neg(GS(L, \phi'_2))) \vee (\phi \wedge Neg(GS(L, Neg(\alpha))))$. En notant $\phi = (T_1 \vee T_2 \vee \cdots \vee T_k)$, où pour tout $i \in \{1, 2, \ldots, k\}$, T_i est un terme, cette formule est encore logiquement équivalente à la formule :

$$(T_1 \wedge Neg(GS(L, \phi'_2))) \vee \cdots \vee (T_k \wedge Neg(GS(L, \phi'_2))) \vee (\phi \wedge Neg(GS(L, Neg(\alpha))))$$

Puisque la classe $\mathcal{C}$ est traitable pour `TAUT`, la classe des formules ϕ avec $Neg(\phi) \in \mathcal{C}$ est traitable pour `SAT`. De fait, on peut décider efficacement la satisfaisabilité de la formule $Neg(GS(L, \phi'_2))$ (qui est dans $\mathcal{C}$ par construction) ; comme T_1 est un terme, et comme $\mathcal{C}$ est stable par propagation d'affectations, on peut décider efficacement la satisfaisabilité de la formule FNC $(T_i \wedge Neg(GS(L, \phi'_2)))$. D'autre part, on voit que $Neg(GS(L, Neg(\alpha)))$ est une clause, et donc on peut déterminer efficacement la satisfaisabilité de la formule $(\phi \wedge Neg(GS(L, Neg(\alpha))))$, qui est logiquement équivalente à la formule $(\phi \wedge \ell_1) \vee (\phi \wedge \ell_2) \vee \cdots \vee (\phi \wedge \ell_{k'})$, où $Neg(GS(L, Neg(\alpha))) = (\ell_1 \vee \ell_2 \vee \cdots \vee \ell_{k'})$. Finalement, on peut déterminer efficacement la satisfaisabilité de la formule ϕ'', et on voit facilement que si elle est satisfaisable, on peut dans le même temps en calculer un modèle. Soit donc m un tel modèle ; il est facile de calculer $E = T(m_{|L})$. Il ne reste donc qu'à minimiser E, mais comme vérifier $\phi \wedge E' \backslash \{\ell\} \models \alpha$ revient à vérifier que $\phi \wedge E' \backslash \{\ell\} \wedge Neg(\alpha)$ est insatisfaisable, que ϕ est en FND et que $E' \backslash \{\ell\} \wedge Neg(\alpha)$ est un terme, on peut effectuer ce test efficacement. □

Reprenons l'exemple du carrefour, et considérons par exemple le problème d'abduction du paragraphe 11.1.1, en prenant la formule FND ϕ'_C comme base de connaissances de référence. Rappelons que ϕ'_C est la formule :

$$\phi'_C = (\neg x_{PV} \wedge \neg x_{VP} \wedge \neg x_{DT}) \vee (\neg x_{VP} \wedge x_{VV} \wedge \neg x_{DT}) \vee (x_{VP} \wedge \neg x_{VV} \wedge x_{DT}) \vee (\neg x_{VP} \wedge \neg x_{VV} \wedge \neg x_{DT})$$

et que la requête α_C et l'ensemble d'abductibles L_C du problème sont, respectivement, la variable x_{DT} et l'ensemble $\{x_{VP}, \neg x_{VP}, x_{VV}, \neg x_{VV}\}$. Pour résoudre le problème, nous calculons donc tout d'abord la formule $\phi_1 = \phi'_C \wedge \neg x_{DT}$ (nous conservons les notations de la preuve de la proposition 11.4), puis nous propageons l'affectation de 0 à x_{DT} dans la formule ϕ'_C, obtenant ainsi la formule :

$$\phi_2 = ((\neg x_{PV} \wedge \neg x_{VP}) \vee (\neg x_{VP} \wedge x_{VV}) \vee (\neg x_{VP} \wedge \neg x_{VV})) \wedge \neg x_{DT}$$

Nous pouvons alors calculer un projeté ϕ' de ϕ_2 sur L_C, et nous obtenons la formule :

$$\phi' = (\neg x_{VP}) \vee (\neg x_{VP} \wedge x_{VV}) \vee (\neg x_{VP} \wedge \neg x_{VV})$$

Enfin, nous pouvons calculer la formule $\phi'' = \phi'_C \wedge \neg\phi'$, qui est logiquement équivalente à la disjonction :

$$\begin{array}{lllll}
 & ((\neg x_{PV} \wedge \neg x_{VP} \wedge \neg x_{DT}) & \wedge & ((x_{VP}) \wedge (x_{VP} \vee \neg x_{VV}) \wedge (x_{VP} \vee x_{VV}))) \\
\vee & ((\neg x_{VP} \wedge x_{VV} \wedge \neg x_{DT}) & \wedge & ((x_{VP}) \wedge (x_{VP} \vee \neg x_{VV}) \wedge (x_{VP} \vee x_{VV}))) \\
\vee & ((x_{VP} \wedge \neg x_{VV} \wedge x_{DT}) & \wedge & ((x_{VP}) \wedge (x_{VP} \vee \neg x_{VV}) \wedge (x_{VP} \vee x_{VV}))) \\
\vee & ((\neg x_{VP} \wedge \neg x_{VV} \wedge \neg x_{DT}) & \wedge & ((x_{VP}) \wedge (x_{VP} \vee \neg x_{VV}) \wedge (x_{VP} \vee x_{VV})))
\end{array}$$

On voit alors que la troisième formule de cette disjonction est satisfaisable, puisqu'en affectant 1 à x_{VP}, 0 à x_{VV} et 1 à x_{DT} (pour satisfaire le terme $(x_{VP} \wedge \neg x_{VV} \wedge x_{DT})$) on satisfait toute la formule ; on peut ensuite affecter une valeur quelconque à x_{PV}. On projette alors cette affectation sur l'ensemble d'abductibles, obtenant l'explication $(x_{VP} \wedge \neg x_{VV})$ pour le problème, et finalement, après minimisation on obtient l'explication (x_{VP}), ce qui termine le calcul.

Classes concernées La dernière hypothèse de la proposition 11.4, la traitabilité du problème `TAUT`, est vérifiée par toutes les classes de formules FND que nous considérons dans ce mémoire (cf. chapitre 4), puisque c'est l'un des critères qui nous a fait choisir ces classes. Remarquons également que toutes ces classes vérifient les deux premières hypothèses de la proposition, sauf les classes HDP-FND et HDP-Ren-FND. Nous pouvons donc établir la traitabilité de l'abduction lorsque la base de connaissances est donnée par une formule de l'une des classes :

Horn-FND, Anti-Horn-FND, Nég-FND, Pos-FND, Bij-FND
Horn-Ren-FND, Monot-FND

et que le but est donné par une clause. Notons encore que, les classes HDP-FND et HDP-Ren-FND étant des sous-classes des classes Horn-FND et Horn-Ren-FND, respectivement, le résultat est le même pour elles ; rappelons même que, plus généralement, toutes les classes mentionnées sont des sous-classes de la classe Horn-Ren-FND (si l'on omet les formules 2FND tautologiques). Il ne reste donc que le cas de la classe DisjAffine, qui n'est pas une classe de formules FND ; nous le traitons à la fin de ce paragraphe.

Dans la lignée des résultats précédents, nous sommes également en mesure d'en donner d'autres pour les formules FND, en imposant des restrictions supplémentaires sur la classe de la base de connaissances mais en autorisant des requêtes plus générales.

Proposition 11.5 (Horn-FND, Nég-FND) *Les problèmes*

`Abduction[Horn-FND,Pos]`, `Abduction[Anti-Horn-FND,Nég]`,
`Abduction[Nég-FND,Anti-Horn]`, `Abduction[Pos-FND,Horn]`

sont traitables.

Preuve Nous ne détaillons pas la preuve, qui est très similaire à celle de la proposition 11.4. Il s'agit seulement de remarquer (pour le cas HORN-FND) que la formule $\phi \wedge Neg(\alpha)$, où $\phi \in$ HORN-FND et $\alpha \in$ POS, est logiquement équivalente à la formule obtenue en distribuant la conjonction $\wedge$ dans ϕ et dans $Neg(\alpha)$, donc à une conjonction de termes de la forme $T \cup T'$ avec $T \in \phi$ et $T' \in Neg(\alpha)$; or, si T est un terme de Horn et T' un terme négatif, on voit que $T \cup T'$ est un terme de Horn. □

Disjonctions d'équations linéaires Nous considérons enfin la classe DISJAFFINE, et montrons que l'on peut projeter les formules de cette classe en temps linéaire.

Lemme 11.3 *Soient ϕ une disjonction d'équations linéaires et L un ensemble de littéraux formés sur $Var(\phi)$. On peut calculer un projeté $\phi' \in$ DISJAFFINE de ϕ sur L en temps linéaire en la taille de ϕ.*

Preuve L'opération de projection étant distributive sur la disjonction, il suffit de savoir projeter une équation linéaire. Le cas $L = Var(\phi)$ est trivial. Nous traitons le cas $L = Var(\phi)\backslash\{x\}$, où x est une variable de $Var(\phi)$. Soit E une équation linéaire ; si E ne contient pas x alors E elle-même est un projeté convenable, et nous supposons donc que E est de la forme $(x \oplus x_{i_1} \oplus x_{i_2} \oplus \cdots \oplus x_{i_k} = a)$ avec $x_{i_1}, x_{i_2}, \ldots, x_{i_k} \in Var(\phi)$ et $a \in \{0, 1\}$. En reprenant la définition de Marquis [Mar00], on a :

$$\begin{aligned} E_{|L} &= fg(E, \{x\}) \\ &= E[x:1] \vee (\neg x \wedge E[x:0]) \\ &= (x_{i_1} \oplus x_{i_2} \oplus \cdots \oplus x_{i_k} = a \oplus 1) \vee (\neg x \wedge (x_{i_1} \oplus x_{i_2} \oplus \cdots \oplus x_{i_k} = a)) \end{aligned}$$

Il est facile de voir que cette disjonction est équivalente à la disjonction $(x = 0) \vee (x_{i_1} \oplus x_{i_2} \oplus \cdots \oplus x_{i_k} = a \oplus 1)$, et que la construction est symétrique pour le cas $L = Var(\phi)\backslash\{\neg x\}$. Nous obtenons donc une disjonction d'équations linéaires logiquement équivalente à $E_{|L}$, et dans le cas général on peut donc itérer la construction. On voit alors facilement que l'on peut obtenir un algorithme linéaire (voir l'exemple ci-dessous). □

Ainsi, si l'on considère la «négation» de la base de connaissances ϕ''_t de l'exemple de la thèse (chapitre 3, paragraphe 3.2) :

$$(x_{qu} \oplus x_{ba} = 0) \vee (x_{me} \oplus x_{ba} \oplus x_{in} = 1) \vee (x_{in} \oplus x_{ch} = 1)$$

alors la formule suivante est l'un de ses projetés sur l'ensemble de littéraux $V_t \cup \{\neg x \mid x \in V_t\}\backslash\{x_{ba}\}$:

$$(x_{ba} = 0) \vee (x_{qu} = 1) \vee (x_{me} \oplus x_{in} = 0) \vee (x_{in} \oplus x_{ch} = 1)$$

Si l'on projette ϕ''_t sur l'ensemble de littéraux $V_t \cup \{\neg x \mid x \in V_t\}\backslash\{x_{ba}, \neg x_{me}\}$, on obtient donc par exemple la formule :

$$(x_{ba} = 0) \vee (x_{qu} = 1) \vee (x_{me} = 1) \vee (x_{in} = 0) \vee (x_{in} \oplus x_{ch} = 1)$$

Nous obtenons finalement le résultat suivant, dont la preuve est similaire à celle de la proposition 11.4.

Proposition 11.6 (DISJAFFINE) *Le problème* `Abduction[DISJAFFINE,AFFINE]` *est traitable.*

Entrée : Une formule affine ϕ et un sous-ensemble V de $Var(\phi)$
Sortie : Un projeté $\phi' \in$ AFFINE de ϕ sur V.

Début
$<\leftarrow$ un ordre total sur $Var(\phi)$ tel que $\forall x \in Var(\phi)\backslash V, x' \in V, x < x'$;
$\phi_{tr} \leftarrow$ `PivotGauss`(ϕ) selon $<$;
$\phi' \leftarrow \{E \in \phi_{tr} \mid Var(E) \subseteq V\}$;
retourner ϕ' ;
Fin

FIG. 11.2 – Algorithme `ProjAffine`

11.3.2 Formules affines

Nous montrons maintenant que l'on peut trouver un algorithme polynomial sur le modèle de `Expliquer` pour le problème `Abduction[`AFFINE`,`DISJAFFINE`]` si on impose une restriction sur l'ensemble des littéraux abductibles : nous demandons que si un littéral en fait partie, alors le littéral opposé en fasse également partie ; nous dirons alors que l'ensemble est *clos par négation*, et nous définissons donc le problème d'abduction suivant.

Problème 11.3 (`Abduction-C[`$\mathcal{C}_\phi$`,`$\mathcal{C}_\alpha$`]`)

Donnée : *Un problème d'abduction* $\Pi = (\phi, \alpha, L)$ *pour* $\mathcal{C}_\phi$ *et* $\mathcal{C}_\alpha$, *avec* L *clos par négation*
Sortie : *Une explication minimale pour* Π.

Avec cette restriction, les opérations de projection de formules deviennent donc des opérations de projection sur des ensembles de *variables* ; par abus de langage, nous parlons donc d'un projeté d'une formule sur un ensemble de variables V au lieu du projeté sur l'ensemble de littéraux clos par négation $\{x \mid x \in V\} \cup \{\neg x \mid x \in V\}$.

Nous montrons en effet qu'en supposant cette restriction, on peut calculer efficacement un projeté affine d'une formule affine donnée. L'algorithme que nous proposons consiste à appliquer le pivot de Gauss (voir le chapitre 3) à la formule donnée en respectant un certain ordre sur ses variables ; plus précisément, nous les ordonnons de sorte que celles sur lesquelles on projette soient plus grandes que toutes les autres (voir chapitre 3, paragraphe 3.2). L'algorithme est reporté sur la figure 11.2. Ainsi, nous avons vu (chapitre 3, paragraphe 3.2) que la formule :

$$\phi''_t = (x_{qu} \oplus x_{ba} = 1) \wedge (x_{me} \oplus x_{ba} \oplus x_{in} = 0) \wedge (x_{in} \oplus x_{ch} = 0)$$

de l'exemple de la thèse était triangulaire selon l'ordre $x_{qu} < x_{me} < x_{in} < x_{ba} < x_{ch}$ sur ses variables. Pour calculer un projeté de cette formule sur l'ensemble de variables $\{x_{me}, x_{in}, x_{ba}, x_{ch}\}$, l'algorithme calcule donc simplement la formule $(x_{me} \oplus x_{ba} \oplus x_{in} = 0) \wedge (x_{in} \oplus x_{ch} = 0)$.

Lemme 11.4 *Soient* ϕ *une formule affine et* V *un sous-ensemble de* $Var(\phi)$. *L'algorithme* `ProjAffine` *calcule en temps* $O(|\phi|^2|Var(\phi)|)$, *sur la donnée* (ϕ, V), *un projeté* $\phi' \in$ AFFINE *de* ϕ *sur* V.

Preuve Le fait que ϕ' est une formule affine est évident. Soit m une affectation à $Var(\phi)$. Si m satisfait ϕ, alors m satisfait la formule triangulaire ϕ_{tr} calculée par l'algorithme (puisque l'on a $\phi_{tr} \equiv \phi$ par construction), et donc en particulier toutes les équations de ϕ_{tr} ; ces dernières

équations n'étant formées que sur V, on a bien $m' \models \phi_{tr}$, et donc $m' \models \phi'$ pour toute affectation m' à $Var(\phi)$ avec $m'_{|L} = m_{|L}$. Réciproquement, si m satisfait ϕ', soit m' l'affectation partielle $m_{|V \cup \{\neg x \,|\, x \in V\}}$; il suffit donc de montrer qu'il existe une affectation $m'' \in ext(m')$ avec $m'' \models \phi$; or ϕ_{tr} est triangulaire sur un ordre des variables pour lequel les variables de V sont les plus grandes, donc toute affectation à V satisfaisant les équations de ϕ_{tr} formées sur V (intuitivement, les «dernières» équations de ϕ_{tr}) peut être prolongée en une affectation m'' à $Var(\phi)$ satisfaisant ϕ (pour plus de détails sur ce prolongement, voir le chapitre 3, paragraphe 3.2). □

Nous sommes donc en mesure de montrer le résultat suivant ; nous donnons une forme générale pour la requête α, mais remarquons qu'en particulier, une variable, un littéral ou une clause peuvent être donnés comme des disjonctions d'équations linéaires. La preuve de la proposition est assez similaire à celle de la proposition 11.4, via la dualité entre la disjonction et la conjonction.

Proposition 11.7 (AFFINE) *Le problème* `Abduction-C[AFFINE,DISJAFFINE]` *est traitable.*

Preuve Soit $\Pi = (\phi, \alpha, L)$ un problème d'abduction pour AFFINE et DISJAFFINE, et soit L un ensemble de littéraux clos par négation. Nous explicitons rapidement les calculs de l'algorithme `Expliquer` sur ce problème.

[projection] Notons $\alpha = E_1 \vee E_2 \vee \cdots \vee E_k$, où E_i est une équation linéaire pour tout $i \in \{1, 2, \ldots, k\}$, et pour tout $i \in \{1, 2, \ldots, k\}$, notons $\overline{E_i}$ l'équation linéaire obtenue de E_i en inversant son membre droit. Alors la formule $\phi \wedge \overline{E_1} \wedge \overline{E_2} \wedge \cdots \wedge \overline{E_k}$ est une formule affine logiquement équivalente à $\phi \wedge \neg\alpha$; on peut donc calculer efficacement un projeté $\phi' \in$ AFFINE de cette formule sur L (lemme 11.4).

[calcul d'une explication] Notons $\phi' = E'_1 \wedge E'_2 \wedge \cdots \wedge E'_{k'}$, où E'_i est une équation linéaire pour tout $i \in \{1, 2, \ldots, k'\}$. Alors la formule $\phi'' = \phi \wedge (\overline{E'_1} \vee \overline{E'_2} \vee \cdots \vee \overline{E'_{k'}})$ est logiquement équivalente à $\phi \wedge \neg\phi'$, et en distribuant la conjonction sur la disjonction, décider si elle est satisfaisable revient à décider si l'une au moins de k' formules affines est satisfaisable, ce qui peut être fait efficacement ainsi que, le cas échéant, le calcul d'un modèle. Pour terminer, la minimisation d'une explication peut être menée efficacement, puisque la formule $\phi \wedge E' \backslash \{\ell\} \wedge \neg\alpha$ est logiquement équivalente à la conjonction de la formule affine $\phi \wedge \overline{E_1} \wedge \overline{E_2} \wedge \cdots \wedge \overline{E_k}$ et du terme $E' \backslash \{\ell\}$. □

11.3.3 Classes retrouvées

Nous signalons finalement que notre cadre permet de retrouver des restrictions polynomiales déjà connues pour la recherche d'une explication minimale. Dans les deux cas, ceux des formules FNC monotones et bijonctives, l'algorithme fonctionne de la même manière que le fait la méthode présentée dans [Mar00]. Il s'agit principalement de remarquer qu'il est possible de lister en temps polynomial tous les impliqués premiers d'une formule appartenant à l'une de ces deux classes, et donc de construire un projeté de la base de connaissances en sélectionnant les impliqués premiers formés sur les littéraux concernés. Nous renvoyons à [Mar00] pour plus de détails sur ce point, et nous rappelons simplement les résultats.

Proposition 11.8 (MONOT, 2FNC) *Les problèmes*

`Abduction[MONOT,CLAUSES]`, `Abduction[2FNC,2FND]`

sont traitables.

11.4 Discussion et résumé des résultats connus

Nous terminons ce chapitre en discutant rapidement le modèle d'algorithme `Expliquer` que nous avons exposé.

Tout d'abord, nous montrons que cet algorithme ne permet pas de retrouver *toutes* les classes polynomiales connues pour l'abduction, tout du moins si l'on s'intéresse aux classes de formules $\mathcal{C}$ telles que l'on puisse calculer efficacement un projeté $\phi' \in \mathcal{C}$ d'une formule $\phi \in \mathcal{C}$, comme nous l'avons fait dans le paragraphe 11.3. Rappelons en effet (voir paragraphe 11.1.3) que le problème `Abduction[HDP,`TERMES∩POS`]` est traitable. Nous montrons en revanche que le calcul d'un projeté comme ci-dessus n'est pas traitable pour cette classe. Remarquons toutefois que la traitabilité du problème `Abduction[HDP,`TERMES∩POS`]` tire parti d'une propriété très particulière de cette classe, contrairement aux classes «retrouvées» du paragraphe 11.3.3, qui s'inscrivent dans un cadre plus général. Notons également que la proposition suivante montre également l'intraitabilité de la projection à l'intérieur de la classe HORN (voir la preuve).

Proposition 11.9 (projection dans HORN/HDP) *Il existe deux familles de formules FNC de Horn définies positives, $(\phi_n)_{n\in\mathbb{N}}$ et $(\phi'_n)_{n\in\mathbb{N}}$, et une famille $(L_n)_{n\in\mathbb{N}}$ d'ensembles de littéraux telles que :*

- *pour tout $n \in \mathbb{N}$, ϕ'_n est un projeté de ϕ_n sur L_n*
- *pour tout $n \in \mathbb{N}$, il n'existe aucun projeté $\phi''_n \in$ HDP de ϕ_n sur L_n dont la taille est inférieure à celle de ϕ'_n*
- *la taille de ϕ'_n, exprimée en fonction de n, est exponentiellement plus grande que celle de ϕ_n.*

Preuve Pour $n \in \mathbb{N}$, considérons l'ensemble de variables $\{x_1, x_2, \ldots, x_n\} \cup \{x'_1, x'_2, \ldots, x'_n\} \cup \{x''_1, x''_2, \ldots, x''_n\} \cup \{x\}$, et la formule de HDP :

$$\begin{array}{rcl} & & (\neg x_1 \vee x''_1) \wedge (\neg x'_1 \vee x''_1) \\ & \wedge & (\neg x_2 \vee x''_2) \wedge (\neg x'_2 \vee x''_2) \\ \phi_n = & \wedge & \cdots \\ & \wedge & (\neg x_n \vee x''_n) \wedge (\neg x'_n \vee x''_n) \\ & \wedge & (\neg x''_1 \vee \neg x''_2 \vee \cdots \vee \neg x''_n \vee x) \end{array}$$

Intuitivement, en regardant les clauses comme des règles on peut voir la variable x comme un but que la conjonction des (sous-buts) x''_i permet d'atteindre, tandis que chaque x''_i peut être atteint, soit à partir de x_i, soit à partir de x'_i. Considérons alors l'ensemble de littéraux

$$L_n = \{x_1, \neg x_1, x_2, \neg x_2, \ldots, x_n, \neg x_n\} \cup \{x'_1, \neg x'_1, x'_2, \neg x'_2, \ldots, x'_n, \neg x'_n\} \cup \{x, \neg x\}$$

Et la formule de HDP :

$$\phi'_n = \bigwedge_{\forall i \in \{1,2,\ldots,n\}, y_i \in \{x_i, x'_i\}} (\neg y_1 \vee \neg y_2 \vee \cdots \vee \neg y_n \vee x)$$

Il est facile de voir que ϕ'_n est un projeté de ϕ_n sur L_n ; intuitivement, on a supprimé les sous-buts x''_i. D'autre part, on ne peut trouver deux de ses clauses qui soient résolubles, et comme aucune de ses clauses n'en subsume une autre, on en déduit que ϕ'_n est de taille minimale parmi toutes les formules FNC (de HDP ou non) logiquement équivalentes. Or, on voit que ϕ_n est de taille $5n + 1$, tandis que ϕ'_n est de taille $(n + 1)2^n$.

Puisque ϕ_n et ϕ'_n sont des formules FNC de Horn et que ϕ'_n est de taille minimale parmi *toutes* les formules FNC logiquement équivalentes, l'énoncé de la proposition reste vrai pour la classe HORN. □

Un deuxième point que nous tenons à souligner est le fait que le modèle d'algorithme `Expliquer` présenté dans ce chapitre est très général, en ce sens qu'il fournit un algorithme pour toute représentation de ses entrées si l'on suppose disponibles les opérateurs suivants sur ces représentations : projection, négation, conjonction. On peut raisonnablement supposer d'un système de gestion de connaissances qu'il fournit ces opérateurs comme des *boîtes noires* pour les classes de formules qu'il propose de manipuler, même s'ils ne sont pas efficaces. Bien sûr, il n'est pas garanti que l'algorithme ainsi défini soit aussi efficace qu'un algorithme *ad hoc* pour une classe de formules donnée, comme nous l'avons vu plus haut pour le cas de la classe HDP. Cet algorithme peut cependant être vu comme un moyen général de résoudre le problème, à utiliser si aucun algorithme plus spécialisé pour l'instance du problème rencontrée n'est connu. Notons également que ce modèle d'algorithme est tout aussi correct pour des bases de connaissances *multivaluées*, comme les Problèmes de Satisfaction de Contraintes utilisés par Amilhastre *et al.* [AFM02], même s'il n'y a toujours aucune garantie de complexité.

Notons enfin que ce modèle d'algorithme fournit en intension toutes les explications pour un problème d'abduction donné, sous la forme de la formule ϕ'' (voir figure 11.1). Enumérer toutes ces explications revient alors à énumérer les impliquants premiers de cette formule formés de littéraux abductibles. Notons cependant que cet algorithme ne permet pas d'énumérer les explications *minimales* du problème au sens de la complexité d'énumération, puisque plusieurs explications sont susceptibles d'être minimisées en la même telle explication minimale. Une solution, dans le cas général, pourrait consister à ajouter à la base de connaissances des contraintes empêchant de générer une explication qui ne pourrait être réduite qu'en une explication minimale déjà calculée, mais on ne conserverait pas, en procédant ainsi, la classe de formules de la base de connaissances, et donc non plus, en général, la complexité du processus. Pour plus de détails sur la génération de plusieurs explications, nous renvoyons le lecteur à l'article d'Eiter et Makino [EM02].

Nous terminons enfin par un résumé des résultats connus pour les classes de formules qui nous intéressent ; ce résumé est présenté dans la table 11.1.

$\mathcal{C}_\phi$	$\mathcal{C}_\alpha$	complexité	Référence
HORN, HORN-REN	VARS	NP-complet	[SL90]
ANTI-HORN	littéraux négatifs	NP-complet	[SL90]
HDP	terme positif	traitable	[EG95a]
ANTI-HDP	terme négatif	traitable	[EG95a]
MONOT	CLAUSES	traitable	[Mar00]
BIJ	2FND	traitable	[Mar00]
AFFINE	DISJAFFINE	traitable	* (L clos par négation)
HORN-REN-FND	CLAUSES	traitable	*
HORN-FND	POS	traitable	*
ANTI-HORN-FND	NÉG	traitable	*
POS-FND	HORN	traitable	*
NÉG-FND	ANTI-HORN	traitable	*
DISJAFFINE	AFFINE	traitable	*

TAB. 11.1 – Complexité de l'abduction

Chapitre 12

Raisonnement et approximation

Sommaire

12.1 Présentation 163
12.2 Déduction et abduction avec des approximations fortes 165
12.3 Abduction 167
12.3.1 Vision «méfiante» 167
12.3.2 Vision «restrictive» avec des approximations faibles minimales . . . 167
12.4 Discussion et résumé des résultats connus 170

Ce dernier chapitre est consacré au raisonnement, vu sous l'angle «approximation». Nous nous intéressons donc au raisonnement à partir de bases de connaissances *approximatives*, dans la même acception qu'au chapitre 8. De fait, nous n'étudions plus l'*algorithmique* des processus, mais leur *sémantique*. Avec ce point de vue, nous nous intéressons encore à la vérification de cohérence, à la déduction et à l'abduction. Ce dernier processus n'avait à notre connaissance jamais été réellement étudié sous ce point de vue, et nous montrons quelques premiers résultats intéressants.

12.1 Présentation

Nous rappelons dans ce paragraphe, rapidement, les problématiques qui nous intéressent. Néanmoins, nous renvoyons au chapitre 8 pour des motivations plus précises et pour un état de l'art sur la question.

Problématiques Nous nous intéressons donc aux bases de connaissances représentant des approximations faibles minimales ou fortes maximales de l'ensemble des situations envisageables dans un petit monde donné. Comme nous l'avons déjà discuté dans le chapitre 8, on peut envisager au moins trois manières de raisonner avec de telles approximations. La première, la vision «confiante», consiste à oublier que notre base de connaissances n'est qu'approximative, et donc à considérer les conclusions que l'on peut en tirer comme des vérités sur le petit monde concerné. Quelle que soit la tâche de raisonnement à laquelle on ait affaire, la sémantique de cette manière de raisonner est principalement résumée dans l'idée que les éventuelles erreurs ainsi commises sont d'importance inversement proportionnelle à la qualité de l'approximation, du moment que celle-ci est mesurée de la même manière que les erreurs.

Ce sont les deux autres manières de considérer le raisonnement dans ce cadre qui nous intéressent dans ce chapitre. Rappelons que l'on peut envisager un tel raisonnement de manière

«méfiante» ou «restrictive». Dans le premier cas, il s'agit de décider de la signification du résultat obtenu avec l'approximation après l'avoir calculé, et dans le deuxième cas il s'agit de restreindre les tâches de raisonnement, via des restrictions sur les requêtes, à celles qui sont compatibles avec l'approximation ; par là, nous entendons des tâches pour lesquelles par construction, la réponse obtenue avec la base de connaissances approximative est nécessairement la même que celle qui aurait été obtenue avec une représentation exacte de la réalité.

Nous étudions ici ces deux dernières visions, en nous intéressant de façon générale à la question :

> Soient $\mathcal{C}$ une classe de formules, M un ensemble d'affectations et $\phi_{(l)ub}$ une approximation faible (minimale) de M. Soit α une requête pour une certaine tâche de raisonnement, et soit $\rho_{(l)ub}$ une réponse à cette requête obtenue à partir de la base de connaissances $\phi_{(l)ub}$. Comment peut-on déduire de $\rho_{(l)ub}$ une réponse correcte pour le même problème avec la connaissance M ?

Nous nous intéresserons également, bien entendu, à la même question pour les approximations fortes.

Motivations Rappelons que l'approximation a principalement pour but, dans le contexte de ce mémoire, de permettre d'intégrer des connaissances quelconques à des systèmes de gestion de connaissances n'acceptant que des formules de certaines classes. Rappelons également que l'intérêt de restreindre ainsi les formules qu'un tel système peut manipuler est pour nous la traitabilité des tâches demandées aux bases de connaissances. Il est alors naturel de se demander dans quelle mesure ce compromis expressivité/traitabilité est intéressant.

Nous avons vu que, selon la vision choisie du raisonnement avec des bases de connaissances approximatives, on pouvait espérer un processus de raisonnement, soit cohérent, incomplet mais efficace, soit cohérent, complet mais pas toujours efficace (dans les deux cas pour la vision «méfiante»), soit enfin un raisonnement cohérent, complet et efficace, mais en restreignant les requêtes possibles (vision «restrictive»). Bien entendu, selon les contraintes de l'application réelle de ces processus (temps réel, besoin de réponses exactes etc.), l'une ou l'autre approche pourra être privilégiée, mais dans tous les cas il est nécessaire de déterminer la sémantique exacte de ces processus ; par exemple, déterminer quelles requêtes on peut autoriser, en fonction de la classe de l'approximation et de sa nature (forte ou faible, minimale etc.), pour la vision «restrictive».

Enfin, ces problématiques s'inscrivent directement dans celle de la *compilation* de bases de connaissances ; nous renvoyons le lecteur aux travaux déjà cités concernant ce sujet [SK96, CD97, Lib98, DM02] ainsi qu'au chapitre 8 de ce mémoire.

Résultats connus Nous renvoyons au chapitre 8 pour un état de l'art. Néammoins, rappelons que Selman et Kautz [SK96] montrent que le problème `SAT` peut être résolu avec une Horn-LUB ou une Horn-GLB. Pour les LUBs, ce résultat peut être généralisé à toute classe de formules dans laquelle il existe une formule insatisfaisable (car alors toute LUB de l'ensemble d'affectations vide est insatisfaisable) ; c'est donc le cas pour toutes les classes de formules FNC (ou affines) qui nous intéressent, sauf pour les classes HDP, Anti-HDP et HDP-Ren. Pour les GLBs, le résultat peut être généralisé à toute classe dans laquelle il existe, pour toute affectation à un ensemble de variables, une formule admettant cette seule affectation pour modèle (car alors il existe une GLB satisfaisable de tout ensemble d'affectations non vide) ; c'est le cas pour les classes qui nous intéressent, sauf pour les classes HDP, Anti-HDP, Nég et Pos. Néammoins, pour les classes HDP et Anti-HDP la question ne se pose pas, puisqu'il n'existe pas de formule insatisfaisable dans ces classes.

Pour la déduction, rappelons que del Val [dV95] montre que la $\mathcal{C}$-LUB d'un ensemble d'affectations supporte une déduction restrictive pour les requêtes de $\mathcal{C}$ dès que $\mathcal{C}$ (une classe de formules *définie en nature*) est close par subsomption. C'est donc le cas pour toutes les classes de formules FNC définies en nature qui nous intéressent, sauf pour les classes HDP et ANTI-HDP. Néammoins, pour le cas HDP, étant donné qu'une HDP-LUB d'un ensemble d'affectations ne diffère d'une HORN-LUB que par un modèle, l'affectation à 1 de toutes les variables, et que ce modèle satisfait toute clause de Horn définie positive, on voit que la vision «restrictive» fonctionne encore. Enfin, pour ce qui est de la classe AFFINE on voit aisément que le même résultat est valable, c'est-à-dire que les AFFINE-LUBs supportent une déduction sans perte d'information pour des requêtes données par des formules affines.

Notre apport La problématique de la déduction avec la vision «restrictive» n'a à notre connaissance été étudiée que pour des approximations *faibles minimales*. S'il ne semble pas raisonnable d'espérer des résultats similaires pour les approximations faibles quelconques, il semble moins absurde de se poser cette question pour des approximations *fortes maximales*. Afin de confirmer l'intuition selon laquelle la sémantique de ces approximations est moins «claire» que celle des approximations faibles, nous donnons une liste d'exemples pour lesquels la réponse à un requête obtenue avec une telle base est sans lien avec la «vraie» réponse. Pour ce qui est de l'abduction, la problématique n'ayant à notre connaissance jamais été étudiée, sinon pour la classe HORN et dans un cadre différent (nous préciserons ce point dans le paragraphe 12.3), nous donnons de premiers résultats qui montrent que la vision «restrictive» est intéressante pour ce processus également. Ces résultats sont publiés partiellement [Zan02b, Zan02c].

12.2 Déduction et abduction avec des approximations fortes

Comme nous l'avons dit, nous donnons dans ce paragraphe des exemples d'ensembles d'affectations et d'approximations fortes maximales de ces ensembles à l'appui de l'intuition selon laquelle la sémantique de la déduction et de l'abduction avec de telles approximations n'est pas claire. Notons qu'un résultat dans ce sens est donné dans la littérature, qui s'applique dans notre cadre pour le cas HORN : Khardon et Roth [KR96, théorème 5.8] montrent en effet que la vision restrictive de la déduction (avec des requêtes de la classe HORN) demande au moins les modèles HORN-*caractéristiques* de l'ensemble d'affectations M que l'on approxime ; or, il est facile de voir qu'une HORN-approximation forte ϕ de M n'est en général pas satisfaite par tous les modèles HORN-caractéristiques de M (sinon ϕ serait, par clôture, une approximation *faible* de M) ; la seule exception est le cas où M est descriptible dans HORN, mais il est alors évident que raisonner avec une de ses HORN-GLBs est correct.

Déduction Nous donnons des exemples pour la déduction ; ils sont résumés dans la table 12.1. Pour chaque classe de formules $\mathcal{C}$ qui nous intéresse, nous y donnons un ensemble d'affectations M (à un ensemble de variables $\{x_1, x_2, \ldots\}$), l'ensemble de modèles d'une $\mathcal{C}$-approximation forte maximale ϕ_{glb} de M, et une requête α tels que ϕ_{glb} implique α alors qu'il existe une affectation $m \in M$ ne satisfaisant pas α.

Tous les exemples de requêtes donnés dans cette table sont des littéraux, qui sont des cas particuliers de clauses, de termes, d'équations linéaires etc. Nous pensons donc que ces exemples montrent qu'aucune restriction naturelle sur les requêtes ne permet de raisonner avec des approximations fortes, même maximales, de façon restrictive. Les cas des classes NÉG et HDP, qui sont «à part», sont traités après la proposition suivante.

Classe $\mathcal{C}$	M	$\mathcal{M}(\phi_{glb})$	α
HORN	$\{01, 10\}$	$\{01\}$	$\neg x_1$
HORN	$\{01, 10\}$	$\{01\}$	x_2
HDP	$\{01, 10, 11\}$	$\{01, 11\}$	x_2
NÉG	$\{00, 11\}$	$\{00\}$	$\neg x_1$
BIJ, AFFINE, HORN-REN, HDP-REN	$\{001, 010, 100\}$	$\{001, 010\}$	$\neg x_1$
BIJ, AFFINE, HORN-REN, HDP-REN	$\{011, 110, 101\}$	$\{011, 110\}$	x_2
MONOT	$\{00, 11\}$	$\{00\}$	$\neg x_1$
MONOT	$\{00, 11\}$	$\{11\}$	x_2

TAB. 12.1 – Contre-exemples pour la déduction avec des GLBs

Proposition 12.1 (déduction et GLBs) *Pour chaque ligne de la table 12.1, la colonne intitulée «$\mathcal{M}(\phi_{glb})$» donne l'ensemble de modèles d'une $\mathcal{C}$-GLB de M, et on a $\phi_{glb} \models \alpha$ mais $\exists m \in M, m \not\models \alpha$.*

Preuve Les assertions $\phi_{glb} \models \alpha$ et $\exists m \in M, m \not\models \alpha$ peuvent être vérifiées sur les ensembles d'affectations donnés dans la table. Pour vérifier que la colonne «$\mathcal{M}(\phi_{glb})$» donne bien toujours l'ensemble de modèles d'une $\mathcal{C}$-GLB de M, on peut tout d'abord vérifier avec les opérations de $\mathcal{C}$-clôture et les restrictions sur les renommages candidats (voir chapitre 4, paragraphe 4.2) qu'elle donne bien toujours l'ensemble de modèles d'une $\mathcal{C}$-LB de M. Pour la maximalité, on peut vérifier que dans tous les cas, M n'est pas descriptible dans $\mathcal{C}$, et il suffit alors de remarquer que l'ensemble de modèles donné est toujours de cardinal $|M| - 1$. □

Pour les seules classes NÉG et HDP, la table 12.1 ne donne pas un littéral négatif *et* un littéral positif pour la requête α. Ceci est dû aux raisons suivantes, qui rendent le problème trivial.

Lemme 12.1 *Soient ϕ une formule FNC négative et x une variable. Alors on a $\phi \models x$ si et seulement si ϕ est insatisfaisable.*

Preuve Si ϕ est insatisfaisable, alors on a bien $\phi \models x$. Réciproquement, si ϕ est satisfaisable, alors $\mathcal{M}(\phi)$ vérifie $\mathcal{M}(\phi) = Cl_{\text{NÉG}}(\mathcal{M}(\phi))$, et on a donc $\overline{0_{Var(\phi)}} \in \mathcal{M}(\phi)$. Si x est dans $Var(\phi)$, on a donc $\overline{0_{Var(\phi)}} \models \phi \wedge \neg x$, ce qui contredit $\phi \models x$, et si x n'est pas dans $Var(\phi)$ le prolongement de $\overline{0_{Var(\phi)}}$ à $Var(\phi) \cup \{x\}$ qui affecte 0 à x satisfait $\phi \wedge \neg x$, d'où le résultat. □

Ainsi, pour la formule ϕ_{tennis} du chapitre 2, qui représente intuitivement le concept de la défaite au tennis en fonction des sets gagnés, ce lemme correspond à l'intuition que pour perdre une partie, le gain d'aucun set n'est obligatoire...

Pour des approximations en FNC négative et des requêtes données par un littéral positif, la situation de la proposition 12.1 ne peut donc jamais se produire, sinon avec une approximation insatisfaisable.

Lemme 12.2 *Soient ϕ une formule FNC de Horn définie positive et x une variable. Alors on a $\phi \not\models \neg x$.*

Preuve La preuve est similaire à celle du lemme 12.1, en considérant l'affectation $\overline{1_{Var(\phi)}}$. □

Abduction Nous terminons ce paragraphe en remarquant qu'on peut encore moins espérer une approche «restrictive» avec des approximations fortes pour l'abduction que pour la déduction. En effet, remarquons que la table 12.1 fournit des exemples pour lesquels cette approche ne fonctionne pas, même avec des restrictions très importantes ; il suffit de considérer les problèmes de déduction représentés dans cette table comme des problèmes d'abduction de la forme $\Pi = (\phi, \alpha, L)$, où ϕ est une description de M et L un ensemble de littéraux quelconques. Alors dans tous les cas listés dans la table, le problème (ϕ_{glb}, α, L) admet le terme vide () comme explication minimale, alors que ce terme n'est pas une explication pour Π. De fait, aucune restriction sur les hypothèses autorisant le terme vide ne peut fonctionner dans tous ces cas.

12.3 Abduction

Nous étudions maintenant le processus d'*abduction* à partir de connaissances approximatives. A notre connaissance, le seul autre travail à aborder ce sujet est celui de Khardon et Roth [KR96], où y est étudiée la possibilité de rechercher des explications positives à un littéral positif avec l'ensemble des modèles «caractéristiques» d'une base de connaissances, vis-à-vis d'une certaine base (nous renvoyons à l'article pour les définitions).

12.3.1 Vision «méfiante»

Les résultats que l'on peut énoncer à propos de la recherche d'une explication avec une base de connaissances approximative sont plus restrictifs que leurs pendants pour la vérification de cohérence ou la déduction. Ceci est principalement lié aux conditions «opposées» que doit satisfaire une explication, la justification de la requête et la cohérence avec la base de connaissances.

Nous résumons simplement ce que l'on peut dire de l'abduction et de l'approche «méfiante» dans la proposition suivante ; toutes les assertions sont faciles.

Proposition 12.2 (abduction méfiante) *Soient $\mathcal{C}$ une classe de formules, M un ensemble d'affectations à un ensemble de variables V et ϕ (resp. ϕ_{ub}, ϕ_{lb}) une description (resp. une $\mathcal{C}$-approximation faible, une $\mathcal{C}$-approximation forte) de M. Soient α une formule propositionnelle et E un terme sur $V \cup Var(\alpha)$. Alors on a :*

- *Si $\phi_{lb} \wedge E$ est satisfaisable, alors $\phi \wedge E$ est satisfaisable*
- *Si $\phi_{lb} \wedge E$ n'implique pas α, alors $\phi \wedge E$ n'implique pas α*
- *Si $\phi_{ub} \wedge E$ n'est pas satisfaisable, alors $\phi \wedge E$ n'est pas satisfaisable*
- *Si $\phi_{ub} \wedge E$ implique α, alors $\phi \wedge E$ implique α.*

12.3.2 Vision «restrictive» avec des approximations faibles minimales

Nous étudions enfin, dans ce paragraphe, la vision «restrictive» de l'abduction avec des approximations faibles minimales. Rappelons que l'approche restrictive consiste à imposer des restrictions sur les requêtes afin d'obtenir un raisonnement cohérent, complet et efficace.

Horn-Approximations Comme nous l'avons annoncé, nous commençons par étudier le cas des Horn-approximations faibles minimales. Nous devons avant tout imposer une restriction sur le but α.

Lemme 12.3 (voir [RdK87, corollaire 4]) *Soit $\Pi = (\phi, \alpha, L)$ un problème d'abduction pour* HORN *et* TERMES*, avec α un terme* positif*. Alors toute explication minimale pour Π est un terme positif.*

Preuve Soit E une explication pour Π, et x une variable avec $\neg x \in E$; nous montrons que $E\backslash\{\neg x\}$ est une explication pour Π. En effet, par définition d'une explication il existe une affectation m à $Var(\phi) \cup Var(\alpha)$ avec $m \models E$, donc en particulier $m[x] = 0$. Soit alors m' une affectation à $Var(\phi) \cup Var(\alpha)$ avec $m' \models \phi \wedge E\backslash\{\neg x\}$; nous montrons que m' satisfait α. En effet, puisque ϕ est de Horn, on a $m \wedge m' \models \phi$; par construction, on a $m \wedge m' \models E\backslash\{\neg x\}$ et $(m \wedge m')[x] = 0$, donc $m \wedge m' \models E$, et donc $m \wedge m' \models \alpha$. Puisque α est un terme positif, on doit donc avoir $(m \wedge m')[x'] = 1$, et donc en particulier $m'[x'] = 1$, pour tout $x' \in Var(\alpha)$. Finalement, m' satisfait α, donc on a bien $\phi \wedge E\backslash\{\neg x\} \models \alpha$. Puisque m satisfait $\phi \wedge E\backslash\{\neg x\}$, $E\backslash\{\neg x\}$ est bien une explication pour Π. □

En utilisant ce lemme, nous pouvons montrer que rechercher une explication minimale avec une base de connaissances ou avec une HORN-approximation faible minimale de cette base revient au même, du moment que l'on n'autorise que des buts donnés par des termes positifs et que l'on restreint les hypothèses aux conjonctions de littéraux *positifs*.

Proposition 12.3 (abduction avec une HORN-LUB) *Soit $\Pi = (\phi, \alpha, L)$ un problème d'abduction, avec α un terme positif. Soit ϕ_{lub} une* HORN*-LUB de $\mathcal{M}(\phi)$. Si un terme $E \subseteq L$ est une explication minimale pour $\Pi_{lub} = (\phi_{lub}, \alpha, L)$, alors E est une explication minimale pour Π. Réciproquement, si E est une explication minimale pour Π* constituée uniquement de littéraux positifs*, alors E est une explication minimale pour Π_{lub}.*

Preuve Dans toute la preuve, nous supposons fixée la formule ϕ_{lub} de l'énoncé.

[sens direct] Soit E une explication minimale pour Π_{lub}. On sait d'après la proposition 12.2 que $\phi \wedge E$ implique α. Soit m une affectation à $Var(\phi) \cup Var(\alpha)$ satisfaisant $\phi_{lub} \wedge E$; alors par la proposition 8.1 (chapitre 8), soit m satisfait ϕ, soit m est le produit ($\wedge$) d'affectations satisfaisant ϕ, qui affectent donc 1 à toutes les variables auxquelles m affectent 1 ; puisque E est un terme positif (lemme 12.3), on en déduit que toutes le satisfont, et donc que $\phi \wedge E$ est satisfaisable. Il reste seulement à montrer que E est minimale. Or, si ce n'était pas le cas, alors on pourrait trouver une explication E' pour Π avec $E' \subset E$. Mais alors $\phi_{lub} \wedge E'$ serait satisfaisable (m en serait un modèle), mais par minimalité de E pour Π_{lub} on n'aurait pas $\phi_{lub} \wedge E' \models \alpha$, donc on aurait un modèle m' de $\phi_{lub} \wedge E' \wedge \neg\alpha$; à nouveau, on aurait donc un modèle de ϕ affectant 1 à toutes les variables auxquelles m' affecte 1, et parmi ceux-ci au moins un affectant 0 à une variable de α ; ce modèle satisferait donc $\phi \wedge E' \wedge \neg\alpha$, contredisant le fait que E' est une explication pour Π.

[réciproque] Soit E une explication minimale *positive* pour Π. D'après la proposition 12.2, $\phi_{lub} \wedge E$ est satisfaisable. La même construction que pour E' dans le paragraphe précédent montre que $\phi_{lub} \wedge E$ implique α, et finalement E est minimale pour ϕ_{lub} car sinon il existerait une explication $E' \subset E$ pour Π_{lub}, et E ne serait pas minimale pour Π par la première partie de l'énoncé. □

Donc si l'on sait que les buts seront tous des termes positifs et que l'on n'aura besoin que d'explications positives, ou si l'on est prêt à se restreindre à se cadre, on ne perd pas d'information à utiliser une HORN-approximation faible minimale de notre connaissance plutôt qu'une description de cette dernière. Notons que le jeu peut en valoir la chandelle, puisque alors la complexité du problème passe du deuxième au premier niveau de la hiérarchie polynomiale.

Ainsi, si l'on reprend l'exemple de la thèse (avec la base de connaissances ϕ_t), il est facile de voir sur l'ensemble d'affectations $M_t = \mathcal{M}(\phi_t)$ que les explications *positives minimales* pour le but x_{me} et l'ensemble d'abductibles constitué de tous les littéraux formés sur V_t sont exactement les trois termes :

$$(x_{me}), (x_{qu} \wedge x_{in}), (x_{qu} \wedge x_{ch})$$

Si l'on reprend alors la HORN-LUB ϕ_t''' de M_t (voir le chapitre 8, paragraphe 8.1.1), dont l'ensemble de modèles est :

$$M_t''' = \{00111, 01000, 10100, 11011, 00000, 00100, 00011, 10000\}$$

il est encore facile de voir que les explications positives minimales pour x_{me} avec ϕ_t''' comme base de connaissances de référence (et toujours le même ensemble d'abductibles) sont les mêmes. En revanche, le terme *négatif* $(\neg x_{qu} \wedge \neg x_{in})$ est une explication minimale lorsque ϕ_t est la base de référence, mais n'est même pas une explication lorsque l'on se réfère à ϕ_t''' (considérer le modèle 00000 de ϕ_t''').

NÉG-approximations Les formules *négatives*, quant à elles, admettent un résultat très similaire. Nous établissons à nouveau un lemme restreignant les explications, puis nous donnons le parallèle de la proposition 12.3 pour cette classe.

Lemme 12.4 *Soit* $\Pi = (\phi, \alpha, L)$ *un problème d'abduction pour* NÉG *et* TERMES, *avec* α *un terme* négatif. *Alors toute explication minimale pour* Π *est un terme positif.*

Preuve Soit E une explication pour Π. Si E contient un littéral négatif $\neg x$ nous considérons l'hypothèse $E' = E \backslash \{\neg x\}$ pour Π. Si E' n'est pas une explication pour Π, alors il existe une affectation m à $Var(\phi) \cup Var(\alpha)$ satisfaisant $\phi \wedge E' \wedge \neg\alpha$, et puisque $\phi \wedge E$ implique α on déduit que m satisfait $\phi \wedge E' \wedge \{x\} \wedge \neg\alpha$. Donc m affecte 1 à x, et de la proposition 2.10 (chapitre 2) on déduit que l'affectation m' obtenue de m en posant $m'[x] = 0$ et $\forall x' \in Var(\phi) \cup Var(\alpha), x' \neq x \Longrightarrow m'[x'] = m[x']$, qui vérifie $m' \prec m$, satisfait $\phi \wedge E \wedge \neg\alpha$, ce qui est absurde puisque E est une explication pour Π. □

Proposition 12.4 (abduction avec une NÉG-LUB) *Soit* $\Pi = (\phi, \alpha, L)$ *un problème d'abduction, avec* α *un terme négatif. Soit* ϕ_{lub} *une* NÉG-*LUB de* $\mathcal{M}(\phi)$. *Si un terme* $E \subseteq L$ *est une explication minimale pour* $\Pi_{lub} = (\phi_{lub}, \alpha, L)$, *alors* E *est une explication minimale pour* Π. *Réciproquement, si* E *est une explication minimale pour* Π constituée uniquement de littéraux positifs, *alors* E *est une explication minimale pour* Π_{lub}.

Preuve La preuve est similaire à celle de la proposition 12.3, en rappelant que pour une affectation donnée satisfaisant ϕ_{lub}, il en existe par construction une *supérieure ou égale*, pour l'ordre $\prec$, satisfaisant ϕ (proposition 2.10 du chapitre 2). □

BIJ-approximations Nous terminons ce chapitre par un résultat, toujours similaire, pour les approximations faibles minimales $2FNC$. Comme pour les autres classes, nous devons tout d'abord imposer des restrictions sur le but α et sur les hypothèses afin d'utiliser un lemme similaire au lemme 12.3. Notons que ces restrictions sont nettement plus importantes, puisque le but doit être donné par une 2clause, et que les hypothèses sont restreintes à celles constituées d'au plus un littéral ; mais il semble difficile de relâcher ces contraintes.

Lemme 12.5 *Soit* $\Pi = (\phi, \alpha, L)$ *un problème d'abduction pour* BIJ *et* TERMES, *avec* α *une* 2*clause. Alors toute explication minimale pour* Π *est constituée d'au plus un littéral.*

Preuve Soit E une explication minimale pour Π, et notons $E = (\ell_{i_1} \wedge \ell_{i_2} \wedge \cdots \wedge \ell_{i_k})$, où pour tout $p \in \{1, 2, \ldots, k\}$ ℓ_{i_p} est un littéral, et de façon similaire notons $\alpha = (\ell_{j_1} \vee \ell_{j_2})$. Puisque E est une explication pour Π, ϕ implique la formule $Neg(E) \vee \alpha$ (logiquement équivalente à la formule $E \rightarrow \alpha$), qui est une clause. Il existe donc un impliqué premier de ϕ, donc une 2clause (proposition 2.14 du chapitre 2), incluse dans $Neg(E) \vee \alpha$; notons C une telle clause. Alors nous distinguons trois cas. Tout d'abord, si C ne contient que des littéraux de $Neg(E)$, alors cela signifie que l'on a $\phi \models Neg(E)$, ce qui contredit le fait que $\phi \wedge E$ est satisfaisable. Ensuite, si C ne contient que des littéraux de α, alors on a $\phi \models \alpha$, et donc, soit E est vide, soit E n'est pas minimale. Enfin, si C contient un littéral $Neg(\ell_{i_p}) \in Neg(E)$ et un littéral $\ell_{j_q} \in \alpha$, alors on a $\phi \models (Neg(\ell_{i_p}) \vee \ell_{j_q})$, donc on a $\phi \wedge \ell_{i_p} \models \ell_{j_q}$, et comme $\phi \wedge E$, donc $\phi \wedge \ell_{i_p}$, est satisfaisable, cela signifie que E est réduit à (ℓ_{i_p}) (E n'est pas vide, car sinon la clause (ℓ_{j_q}) serait impliquée par ϕ, donc C ne serait pas première). □

On en déduit, de même que pour les cas Horn et négatif, le résultat suivant.

Proposition 12.5 (abduction avec une BIJ-LUB) *Soit* $\Pi = (\phi, \alpha, L)$ *un problème d'abduction, avec* α *une* 2*clause. Soit* ϕ_{lub} *une* BIJ-*LUB de* $\mathcal{M}(\phi)$. *Si un terme* $E \subseteq L$ *est une explication minimale pour* $\Pi_{lub} = (\phi_{lub}, \alpha, L)$, *alors* E *est une explication minimale pour* Π. *Réciproquement, si* E *est une explication minimale pour* Π constituée d'aucun ou d'un seul littéral, *alors* E *est une explication minimale pour* Π_{lub}.

Preuve Soit E une explication minimale pour Π_{lub} ; on sait (proposition 12.2) que $\phi \wedge E$ implique α, et (lemme 12.5) que E contient au plus un littéral. Puisque ϕ est satisfaisable, si E est vide alors $\phi \wedge E$ est bien satisfaisable ; supposons donc $E = (\ell)$. Puisque $\phi_{lub} \wedge E$ est satisfaisable, la clause $Neg(E)$ n'est pas impliquée par ϕ_{lub}, et comme ϕ_{lub} est logiquement équivalente à la conjonction de toutes les 2clauses impliquées par ϕ [dV96], ϕ n'implique pas $Neg(E)$, et donc $\phi \wedge E$ est satisfaisable. Enfin, si E n'est pas minimale pour Π cela signifie que ϕ implique α ; on en déduit, puisque α est une 2clause, que ϕ_{lub} l'implique également, et donc que E n'est pas minimale pour Π_{lub}, ce qui est absurde.

La réciproque, quant à elle, se montre exactement comme pour la proposition 12.3. □

12.4 Discussion et résumé des résultats connus

Nous avons posé dans ce chapitre de premières pierres en direction de l'étude du raisonnement abductif avec une approximation d'une base de connaissances. Nous avons vu que les restrictions nécessaires étaient importantes, mais pouvaient permettre, comme pour la déduction et la vérification de cohérence, de gagner en complexité.

Ces résultats peuvent également être vus comme une reformulation de la caractérisation des $\mathcal{C}$-LUBs par la notion d'impliqué premier [SK96, dV96] (voir la preuve de la proposition 12.5 par exemple), lorsqu'il s'agit des BIJ-LUBs ou des buts donnés par un littéral pour les classes HORN et NÉG. Néanmoins, il s'agissait ici de donner ces résultats dans un cadre sémantique, d'une part, et d'autre part de montrer que d'autres tâches que la déduction supportaient une vision «restrictive» pour le raisonnement à partir d'approximations. Enfin, nos résultats sont plus généraux, puisque la caractérisation par les impliqués premiers ne permet pas de les établir lorsque les buts sont donnés par des termes (cas HORN et NÉG).

En revanche, nous ne sommes pas en mesure de donner des résultats positifs pour la classe des formules *affines*. L'exemple suivant laisse même peu d'espoir.

Exemple 12.1 (abduction avec une AFFINE-LUB) *Soit M l'ensemble d'affectations*

$$\{001, 010, 100\}$$

à l'ensemble de variables $\{x_1, x_2, x_3\}$. On voit aisément que la formule $\phi = (x_1 \oplus x_2 \oplus x_3 = 1)$ est une AFFINE-*LUB de M. Or, si l'on considère l'ensemble d'abductibles $\{x_1, \neg x_1, x_2, \neg x_2\}$, on voit que si la base de connaissances de référence est la formule ϕ, alors le terme $(x_1 \wedge x_2)$ est une explication (minimale) du but x_3 ; en revanche, il n'existe aucune affectation satisfaisant $(x_1 \wedge x_2)$ dans M.*

Nous considérons cet exemple comme particulièrement décourageant, car il n'y a en général aucune influence des signes des littéraux lorsque l'on considère la classe AFFINE. De fait, il semble peu probable que l'on puisse obtenir des résultats sur la vision «restrictive» de l'abduction avec des AFFINE-approximations faibles en imposant des restrictions liées aux signes des variables. L'exemple impliquant un but, un ensemble d'abductibles et une explication par ailleurs très simples, il ne semble plus rester beaucoup de place pour envisager des restrictions. Nous pensons que ces difficultés proviennent du fait que, intuitivement, les approximations affines préservent les liens de «parité» entre les variables ; or, les explications pour un problème d'abduction sont définies comme des termes, soit un lien «conjonctif» entre des littéraux.

Pour terminer, nous résumons tous les résultats cités dans ce chapitre à propos de l'approche «restrictive» dans la table 12.2. Cette table donne, pour une classe de formules et un type d'approximation donnés, les restrictions sur les requêtes supportant cette approche (pour l'abduction, nous notons E une explication quelconque pour le problème en question).

SAT			
Classe $\mathcal{C}$	Approx.	Restrictions	Référence
Horn, Anti-Horn, Bij, Affine	LUB, GLB	—	[SK96]
HDP, Anti-HDP	GLB (si existe)	—	*
Nég, Pos	LUB	—	[SK96], *
Horn-Ren, HDP-Ren, Monot	LUB, GLB	—	[SK96]

Déduction			
Classe $\mathcal{C}$	Approx.	Restrictions	Référence
Horn, Anti-Horn, Nég, Pos, Bij	LUB	$\alpha \in \mathcal{C}$	[dV95]
HDP, Anti-HDP	LUB	$\alpha \in \mathcal{C}$	*
Affine	LUB	$\alpha \in$ Affine	*

Abduction			
Classe $\mathcal{C}$	Approx.	Restrictions	Référence
Horn	LUB	α terme pos., E pos.	* ([KR96])
Anti-Horn	LUB	α terme nég., E nég.	* ([KR96])
Nég	LUB	α terme nég., E pos.	*
Bij	LUB	α 2clause, $\|E\| < 2$	*

Tab. 12.2 – Résumé des classes et restrictions permettant un raisonnement avec approximations

Conclusion générale

Pour terminer le mémoire, nous présentons un résumé des résultats ainsi que certains problèmes laissés ouverts par ce travail. Puis nous discutons les résultats obtenus, et enfin nous proposons des perspectives de recherche qui prolongent les problématiques traitées.

Résultats et problèmes ouverts

Nous avons étudié les principaux problèmes algorithmiques relatifs à l'acquisition de connaissances et au raisonnement pour des bases de connaissances représentées en logique propositionnelle.

Résumé des résultats En raison de la diversité des formules de cette logique, nous avons présenté et motivé des critères pour choisir des formes normales (partie I). Nous avons ainsi choisi de nous intéresser aux formules en Forme Normale Conjonctive, dont les principaux atouts sont la stabilité par conjonction et la lisibilité, et aux formules en Forme Normale Disjonctive, qui permettent notamment de représenter facilement leurs négations. Puis, la plupart des problèmes algorithmiques étant difficiles dans le cas général, nous avons présenté des sous-classes de formules et discuté leur intérêt pour la représentation de connaissances. Nous avons ainsi été amenés à considérer comme intéressantes les classes des formules de Horn, négatives et de Horn définies positives, ainsi que leurs classes duales et les classes des formules renommables dans ces classes, et les classes des formules 2FNC et affines. Nous avons également introduit les classes de formules FND qui permettent d'exprimer les négations des formules de ces classes ; nous ne les avons pas considérées à proprement parler comme des langages de représentation, mais comme des langages susceptibles d'être utiles pour le raisonnement.

Une fois ces langages de représentation choisis, nous nous sommes ensuite intéressés à l'acquisition de connaissances à partir d'exemples (partie II). Pour toutes les problématiques étudiées, il s'agit donc de représenter un ensemble d'exemples, formalisé par un ensemble d'affectations à des variables, par une base de connaissances formulée dans l'un des langages choisis. Nous avons ainsi étudié les problèmes de description et d'identification, c'est-à-dire d'acquisition exacte, et avons présenté un algorithme efficace et unifié pour ces problématiques ; nous avons ensuite présenté des optimisations de cet algorithme pour la classe des formules négatives, monotones et de Horn, obtenant des complexités meilleures que celles connues auparavant. Puis nous avons étudié la problématique de l'acquisition par approximation ; nous avons complété l'état de l'art sur le sujet, en montrant notamment que pour les classes des formules HORN-renommables, Horn définies positives renommables et monotones, il existe en général plusieurs approximations faibles minimales d'un même ensemble d'affectations, non logiquement équivalentes et possédant éventuellement des nombres exponentiellement différents de modèles ; puis nous avons montré le même résultat pour les approximations faibles maximales, pour toutes les classes qui nous inté-

ressent ; nous avons alors donné un algorithme glouton efficace pour en calculer une quelconque, algorithme qui fonctionne même pour les classes de formules renommables ; enfin, pour la classe des formules affines nous avons montré qu'une approximation faible minimale, ou forte maximale avec un nombre maximal de modèles pouvaient être calculées relativement efficacement. Enfin, nous avons étudié le problème du PAC-apprentissage d'une classe de formules, et avons montré que la classe des formules affines était PAC-apprenable à partir d'exemples et de façon dynamique.

Enfin, nous nous sommes intéressé aux problèmes de raisonnement (partie III), toujours en supposant des bases de connaissances représentées dans les classes de formules choisies. Nous avons considéré les processus de vérification de cohérence, de déduction et d'abduction, et avons présenté pour ce dernier problème un modèle d'algorithme général, dont nous avons dérivé des algorithmes efficaces, principalement pour des bases de connaissances affines ou représentées dans les classes de formules FND qui nous intéressent. Enfin, nous avons traité ces mêmes problématiques pour le cas où la base de connaissances utilisée a été acquise par approximation, et avons montré que l'abduction était possible dans ce cadre, en imposant toutefois des restrictions importantes sur les requêtes.

Tout au long du mémoire, nous avons accordé une place particulière aux formules *affines* : à notre connaissance, cette étude est en effet la première réellement menée pour cette classe dans un contexte de représentation de connaissances. Nous avons vu que cette classe possédait de nombreuses bonnes propriétés, parmi lesquelles une définition en nature, des algorithmes efficaces pour l'approximation et un problème d'abduction traitable pour une large classe de buts. A notre connaissance, le seul problème lié à l'Intelligence Artificielle pour lequel on a montré que les formules affines n'admettaient pas d'algorithme efficace est le problème d'inférence par *circonscription* [DH03].

Problèmes ouverts Quelques problèmes sont malgré tout laissés ouverts dans ce mémoire, concernant principalement l'approximation. Nous en discutons brièvement deux, qui nous paraissent particulièrement intéressants.

Nous avons tout d'abord laissé ouvert le problème du calcul d'une HORN-REN-, HDP-REN- ou MONOT-approximation faible d'un ensemble d'affectations donné. Nous avons en effet montré que pour ces trois classes, l'approximation faible minimale n'est pas unique, même à l'équivalence logique près, mais nous ne sommes pas en mesure de discuter le calcul de l'une d'elles, quelconque ou avec le nombre minimal de modèles, comme nous l'avons fait pour les questions duales concernant les approximations fortes. Une question préliminaire, pour les trois classes, est celle du nombre de modèles de ces approximations. En effet, ce nombre étant en général exponentiel en le nombre d'affectations de départ pour les classes HORN, HDP et NÉG, on peut penser que c'est également le cas pour ces classes, même en considérant l'approximation faible minimale avec le nombre minimal de modèles.

Nous ne sommes néanmoins pas en mesure de donner des exemples prouvant cette intuition, et laissons de fait cette question, très intéressante, ouverte. De même, nous laissons ouverte la question de la taille des approximations elles-mêmes ; nous avons en effet vu que le nombre de modèles d'une BIJ-approximation faible minimale ϕ d'un ensemble d'affectations M pouvait être exponentiellement plus grand que le nombre d'affectations dans M, mais qu'il existait toujours une *formule* $2FNC$ décrivant cet ensemble et de taille polynomiale en la taille de M. Quand une classe de formules admet un algorithme polynomial pour la description, on sait que si les *formules* sont potentiellement de taille au mieux exponentielle, alors c'est aussi le cas pour les *ensembles de modèles* de ces formules, mais cet exemple montre que la réciproque n'est pas vraie.

Questions ouvertes (approximations faibles renommables) *Soit $\mathcal{C}$ l'une des classes de formules :*

HORN-REN, HDP-REN, MONOT

et soit M un ensemble d'affectations.
- *Existe-t-il nécessairement une $\mathcal{C}$-approximation faible minimale de M dont l'ensemble de modèles est de taille polynomiale en la taille de M ?*
- *Existe-t-il nécessairement une $\mathcal{C}$-approximation faible minimale de M (une* formule de $\mathcal{C}$*) de taille polynomiale en la taille de M ?*
- *Si oui, peut-on calculer une telle formule efficacement ?*

A notre connaissance, le seul travail consacré à l'approximation dans une classe de formules renommables est l'article de Boufkhad [Bou98]. L'auteur y traite le calcul d'une HORN-REN-approximation forte d'une *formule FNC*, et y aborde expérimentalement la question de la qualité de telles approximations.

Concernant toujours ces classes «renommables», ainsi que la classe des formules affines, nous avons laissé ouvert le problème du raisonnement abductif avec de telles approximations. Nous avons en effet vu que l'on pouvait résoudre des problèmes d'abductions particuliers avec des approximations dans les classes HORN, NÉG et BIJ (et dans leurs classes duales), mais nous ne savons pas étendre ces résultats aux autres classes de formules. Les résultats pour les classes HORN, NÉG et BIJ laissent penser qu'on ne peut espérer des résultats que pour les approximations faibles minimales ; pour les classes «renommables», l'étude de ces approximations doit donc certainement être menée auparavant.

Question ouverte (abduction et approximation) *Existe-t-il des restrictions naturelles permettant de résoudre des problèmes d'abduction avec des $\mathcal{C}$-approximations faibles minimales d'une base de connaissances, lorsque $\mathcal{C}$ est l'une des classes suivantes :*

AFFINE, HORN-REN, HDP-REN, MONOT *?*

Discussion

Nous replaçons maintenant le travail présenté dans ce mémoire dans les contextes plus généraux de l'Intelligence Artificielle et de la théorie de la Complexité.

De retour sur Mars Les résultats donnés ou rappelés dans ce mémoire (voir les résumés à la fin de chaque chapitre) peuvent être vus comme constituant un catalogue des possibilités offertes par chaque langage considéré, la figure 5.1 (chapitre 5, page 69) comparant leur expressivité. Le langage de représentation d'un système de gestion de connaissances peut alors être choisi en fonction de ce catalogue, comme celui qui présente l'expressivité maximale tout en permettant d'effectuer efficacement les tâches auxquelles est destiné le système.

Revenons ainsi sur le robot explorateur de l'introduction. Si son but principal n'est pas d'acquérir de nombreuses informations, il peut choisir de représenter ses connaissances par des

formules de Horn, bien que cette classe ne permette pas, par exemple, d'approximer efficacement des observations. S'il doit néanmoins raisonner très souvent avec sa base de connaissances, et notamment chercher à expliquer, il a tout intérêt à se limiter à des formules négatives ou de Horn définies positives, car pour ces classes la plupart des tâches de raisonnement peuvent être menées efficacement. En revanche, si son but est simplement de déduire, il n'a pas besoin de se limiter à ces classes, et peut utiliser la plus grande expressivité de la classe de toutes les formules de Horn, et même celle de la classe de toutes les formules HORN-renommables.

Si le robot, en revanche, est censé ramener sur Terre le maximum d'informations nouvelles, et les présenter de façon lisible, on a tout intérêt à le munir d'un langage de représentation de connaissances qui supporte une acquisition efficace, par exemple la classe des formules 2FNC ou affines.

Intelligence Artificielle Cependant, comme nous l'avons dit en introduction, le langage de la logique propositionnelle est souvent trop peu expressif pour suffire à des applications concrètes de l'Intelligence Artificielle. Nous avons néanmoins vu au fil des chapitres que ce formalisme permettait d'encoder un certain nombre de connaissances et de problèmes naturels. De plus, il peut être considéré comme un formalisme «minimal», en ce sens qu'un formalisme logique utilisé pour la représentation de connaissances doit posséder *au moins* son expressivité. Son importance est également visible par exemple dans le fait que l'on peut obtenir directement une formule propositionnelle à partir d'une formule du premier ordre lorsque les domaines des variables sont finis. En un autre sens, la logique propositionnelle peut également être vue comme une logique centrale pour l'informatique car les propriétés de ses formules sont *décidables*, ce qui n'est pas le cas en général des formalismes logiques plus expressifs. De fait, de nombreux problèmes, tels que celui consistant à décider si une formule est ou non toujours vraie, ne peuvent trouver de solution algorithmique dans des formalismes logiques réellement plus expressifs.

Pour toutes ces raisons, nous considérons la compréhension des divers aspects de la logique propositionnelle comme une question très importante, ce qu'atteste par ailleurs la très importante activité de recherche qui la concerne à travers le monde. Nous pensons que ce mémoire contribue à cette compréhension. Nous espérons également qu'au-delà des classes de formules étudiées, les résultats anciens et nouveaux présentés seront une nouvelle étape vers la compréhension profonde d'autres classes connues de formules propositionnelles, qui sont la plupart du temps des extensions des classes étudiées dans ce mémoire, mais aussi vers celle des restrictions de certains formalismes plus expressifs ; cet espoir est notamment fondé sur le fait que l'on retrouve la classe des formules de Horn, très importante en logique propositionnelle, avec la même importance en logique du premier ordre.

Théorie de la complexité La majorité des questions que nous avons étudiées étant de nature algorithmique, ce travail s'inscrit également dans une lignée de travaux s'intéressant à la *complexité* (en temps) des problèmes. Le plus représentatif de ces travaux est certainement celui de Schaefer [Sch78], qui a établi le premier résultat de *dichotomie* en montrant que le problème `SAT-GEN[.]` est soit facile, soit très difficile (voir le chapitre 5). De tels résultats sont très importants pour la compréhension des classes de complexité (voir [CKS01] par exemple). Nous pensons que les travaux présentés dans ce mémoire, en exhibant par exemple de nouvelles classes traitables pour l'abduction, apportent une pierre à cette compréhension ; ainsi, il est notamment conjecturé que l'abduction propositionnelle admet un théorème de «trichotomie», c'est-à-dire que, selon les classes de formules qui le paramètrent, il est de complexité polynomiale, NP-complète ou Σ_2^P-complète.

Perspectives

Pour terminer, nous présentons quelques perspectives de recherche qui pourraient permettre d'étendre le travail présenté.

Problèmes Tout d'abord, concernant les classes de formules étudiées dans ce mémoire, plusieurs *problèmes* non abordés par ce travail sont particulièrement pertinents. En premier lieu, il serait intéressant de connaître les «tailles» des différentes classes de formules étudiées, c'est-à-dire le nombre d'ensembles d'affectations à un ensemble de variables donné qui y sont descriptibles. Ces dénombrements permettraient en effet de mieux comprendre l'expressivité de chaque classe de formules, et notamment la proportion des ensembles d'affectations à un ensemble de variables fixé V qu'elle permet de représenter parmi les $2^{2^{|V|}}$ existant. Un problème moins général, mais tout aussi intéressant, consiste à comparer les nombres d'ensembles d'affectations descriptibles dans chacune des classes, par exemple en estimant leurs quotients. A notre connaissance, on ne sait dénombrer que les ensembles d'affectations descriptibles dans la classe AFFINE (il suffit de dénombrer les bases). Il serait également intéressant d'étudier ces problèmes de dénombrement en se restreignant aux ensembles d'affectations qui peuvent être décrits par des formules de taille polynomiale ; on sait en effet que la proportion des tels ensembles tend vers 0 lorsque le nombre de variables tend vers l'infini.

Ensuite, il serait intéressant d'étudier d'éventuels résultats de dichotomie pour les problèmes de raisonnement que nous avons considérés, montrant par exemple que parmi toutes les classes de requêtes envisageables, pour un langage de représentation de connaissances donné il existe des classes supportant un résultat polynomial, mais que toutes les autres donnent des résultats de difficulté. Pour étudier de tels problèmes, les auteurs se restreignent généralement au cadre des S-formules de Schaefer [Sch78] (voir le chapitre 5) ; ces résultats concernent les différentes classes de formules auxquelles appartiennent les bases de connaissances, mais il serait également intéressant, pour la déduction et l'abduction, de fixer une telle classe et de faire varier les classes des requêtes.

Enfin, concernant les problèmes d'*acquisition de connaissances*, il serait très pertinent d'étudier les versions *dynamiques* des problèmes. Nous avons déjà abordé ce point dans le chapitre 9, concernant le PAC-apprentissage des formules affines. Etudier les problématiques de description et d'approximation dans ce cadre, pour les classes de formules que nous avons traitées, serait du plus haut intérêt. Une discussion de cet aspect, ainsi qu'un panel d'opérateurs de révision (indiquant la sémantique du changement à apporter à la base) peuvent être trouvés dans l'article de Gogic *et al.* [GPS98].

Langages Il serait également intéressant d'essayer d'étudier les mêmes problématiques que celles traitées dans ce mémoire, de façon systématique, mais en s'intéressant à d'autres *langages* de représentation des connaissances.

Tout d'abord, il serait pertinent d'étudier ces problématiques pour d'autres classes de formules propositionnelles que celles étudiées dans ce mémoire ; nous nous sommes en effet restreint à des classes de formules FNC définies en nature (ou renommables dans une telle classe), et avons motivé ce choix, mais d'autres classes de formules peuvent être intéressantes pour le raisonnement. En imaginant en effet une base de connaissances en FNC, mais à laquelle on a ajouté des clauses de natures différentes, une fois confronté à un problème de raisonnement on peut imaginer que la base de connaissances ne fait partie d'aucune «bonne» classe de formules définie en nature (ou de formules renommables dans une telle classe). Il est alors intéressant d'avoir à disposition un certain nombre d'autres classes de formules auxquelles on peut tester de façon

«opportuniste» (c'est-à-dire pour ce problème de raisonnement précis et jusqu'à la prochaine mise à jour «sauvage» de la base) l'appartenance de la base courante. Dans le cas où cette dernière appartient à une telle classe, on peut alors utiliser un éventuel algorithme de raisonnement efficace, et ce même si on ne peut garantir qu'après l'ajout d'une nouvelle clause à la base elle appartiendra toujours à cette classe. Plusieurs classes de formules définies en structure ont ainsi été définies : ordonnées [Ben00], de largeur bornée [dV00a] etc. Toutes sont polynomiales pour `SAT` (c'est ce qui motive en général l'introduction d'une nouvelle classe de formules), mais il serait intéressant d'étudier leurs propriétés algorithmiques pour différentes tâches de raisonnement. De même, il serait intéressant d'étudier d'autres représentations des fonctions booléennes, comme par exemple la représentation par un Diagramme de Décision Binaire [Bry86] ou la représentation par un ensemble de modèles caractéristiques [KR96]. Mentionnons à ce sujet le récent travail d'Horiyama et Ibaraki [HI02] sur des problèmes algorithmiques liés à ces représentations.

Un autre prolongement de ce travail concernerait les classes de formules *multivaluées*, c'est-à-dire pour lesquelles le domaine des variables n'est pas restreint à $\{0, 1\}$. Plusieurs formalismes existent pour étudier de telles formules, ou des représentations équivalentes. Notons par exemple qu'un Problème de Satisfaction de Contraintes peut être vu comme une base de connaissances, l'ensemble des situations envisageables étant codé par l'ensemble de ses solutions (voir le travail d'Amilhastre *et al.* [AFM02] par exemple). Citons également les formules *régulières* [BHM99, BHM00], qui sont une généralisation de la Forme Normale Conjonctive à ce cadre.

Même si les bases de connaissances multivaluées, lorsque les domaines des variables sont finis, ne sont dans l'absolu pas plus expressives que les formules de la logique propositionnelle, elles permettent toutefois de représenter des connaissances de façon plus concise, en incluant par exemple dans la *sémantique* des formules (pour les formules régulières) le fait que plusieurs valeurs peuvent être mutuellement exclusives (si elles font partie du domaine de la même variable), alors qu'une base de connaissances propositionnelle doit exprimer elle-même de tels liens. Ainsi, des formules régulières pourraient permettre à un robot d'affecter simplement la valeur «Immense» à une variable représentant la profondeur d'un exemple du concept «Ravin», plutôt que d'avoir à affecter la valeur «Non» aux variables «Pas-Profond», «Un-Peu-Profond» et «Très-Profond», et la valeur «Oui» à la variable... «*Trop*-Profond».

Bibliographie

[AFM02] J. Amilhastre, H. Fargier, et P. Marquis. Consistency restoration and explanations in dynamic CSPs — application to configuration. *Artificial Intelligence*, 135(1–2) :199–234, 2002.

[AFP90] D. Angluin, M. Frazier, et L. Pitt. Learning conjunctions of Horn clauses (extended abstract). In *Proc. 31st Annual Symposium on Foundations of Computer Science*, pages 186–192. IEEE Computer Society Press, 1990.

[AFP92] D. Angluin, M. Frazier, et L. Pitt. Learning conjunctions of Horn clauses. *Machine Learning*, 9 :147–164, 1992.

[AHHP98] H. Aizenstein, T. Hegedüs, L. Hellerstein, et L. Pitt. Complexity theoretic hardness results for query learning. *Computational Complexity*, 7(1) :19–53, 1998.

[AHU74] A. Aho, J. Hopcroft, et J. Ullman. *The design and analysis of computer algorithms*. Addison-Wesley, 1974.

[AHU83] A. Aho, J. Hopcroft, et J. Ullman. *Data structures and algorithms*. Addison-Wesley, 1983.

[AHV95] S. Abiteboul, R. Hull, et V. Vianu. *Foundations of databases*. Addison-Wesley, 1995.

[Ang90] D. Angluin. Negative results for equivalence queries. *Machine Learning*, 5(2) :121–150, 1990.

[Ang92] D. Angluin. Computational learning theory : survey and selected bibliography. In *Proc. 24th Annual ACM Symposium on Theory Of Computing (STOC'92)*, pages 319–342. ACM Press, 1992.

[APT79] B. Aspvall, M. Plass, et R.E. Tarjan. A linear-time algorithm for testing the truth of certain quantified Boolean formulas. *Information Processing Letters*, 8(3) :121–123, 1979.

[Asp80] B. Aspvall. Recognizing disguised nr(1) instances of the satisfiability problem. *Journal of Algorithms*, 1 :97–103, 1980.

[BATJ89] T. Bylander, D. Allemang, M.C. Tanner, et J.R. Josephson. Some results concerning the computational complexity of abduction. In *Proc. 1st International Conference on Principles of Knowledge Representation and Reasoning (KR'89)*, pages 44–54. Morgan Kaufmann, 1989.

[BCZ01] E. Bertino, B. Catania, et G.P. Zarri. *Intelligent database systems*. ACM Press, 2001.

[Ben00] E. Benoist. *Génération à délai polynomial pour le problème SAT*. Thèse de doctorat, Université de Caen, France, 2000.

[BGM+97] Y. Boufkhad, E. Grégoire, P. Marquis, B. Mazure, et L. Saïs. Tractable cover compilations. In *Proc. 15th International Joint Conference on Artificial Intelligence (IJCAI'97)*, pages 122–127. Morgan Kaufmann, 1997.

[BHM99] B. Beckert, R. Hähnle, et F. Manyà. Transformations between signed and classical clause logic. In *Proc. 29th International Symposium on Multiple-Valued Logics (ISMVL'99)*, pages 248–255. IEEE Computer Society Press, 1999.

[BHM00] B. Beckert, R. Hähnle, et F. Manyà. The 2-SAT problem of regular signed CNF formulas. In *Proc. 30th International Symposium on Multiple-Valued Logics (ISMVL'2000)*, pages 331–336. IEEE Computer Society Press, 2000.

[BHRV02] E. Böhler, E. Hemaspaandra, E. Reith, et H. Vollmer. Equivalence and isomorphism for Boolean constraint satisfaction. In *Proc. Annual Conference of the European Association for Computer Science Logic (CSL'02)*, numéro 2471 in Lecture Notes In Computer Science, pages 412–426, 2002.

[Bou98] Y. Boufkhad. Algorithms for propositional KB approximation. In *Proc. 15th National Conference on Artificial Intelligence (AAAI'98)*, pages 280–285. AAAI Press/MIT Press, 1998.

[Bry86] R.E. Bryant. Graph-based algorithms for Boolean function manipulation. *IEEE Transactions on Computers*, C-35-8 :677–691, 1986.

[Bsh95] N. Bshouty. Exact learning Boolean functions via the monotone theory. *Information and Computation*, 123(1) :146–153, 1995.

[CD97] M. Cadoli et F.M. Donini. A survey on knowledge compilation. *AI Communications*, 10 :137–150, 1997.

[CG99] C. Carlet et P. Guillot. A new representation of Boolean functions. In *Proc. 13th International Symposium on Applied Algebra, Algebraic Algorithms and Error-Correcting Codes (AAECC'13)*, volume 1719 of *Lecture Notes in Computer Science*, pages 94–103. Springer, 1999.

[CH96] N. Creignou et M. Hermann. Complexity of generalized satisfiability counting problems. *Information and Computation*, 125(1) :1–12, 1996.

[CKS01] N. Creignou, S. Khanna, et M. Sudan. *Complexity classifications of Boolean constraint satisfaction problems.* SIAM Monographs on Discrete Mathematics and Applications, 2001.

[CL73] C.-L. Chang et R. Lee. *Symbolic logic and mechanical theorem proving.* Academic Press, 1973.

[CMM98] S. Coste-Marquis et P. Marquis. Characterizing consistency-based diagnoses. In *Proc. 5th International Symposium on Artificial Intelligence and Mathematics (AI-MATH'98)*, 1998.

[Coo71] S.A. Cook. The complexity of theorem-proving procedures. In *Proc. 3rd Annual ACM Symposium on the Theory Of Computing (STOC'71)*, pages 151–158. ACM Press, 1971.

[Cur84] C.W. Curtis. *Linear algebra. An introductory approach.* Springer Verlag, 1984.

[Dec90] R. Dechter. Decomposing a relation into a tree of binary relations. *Journal of Computer and System Sciences*, 41(1) :2–24, 1990.

[DG84] W.F. Dowling et J.H. Gallier. Linear-time algorithms for testing the satisfiability of propositional Horn formulae. *Journal of Logic Programming*, 3 :267–284, 1984.

[DH03] A. Durand et M. Hermann. The inference problem for propositional circumscription of affine formulas is coNP-complete. In *Proc. 20th Annual Symposium on Theoretical Aspects of Computer Science (STACS'03)*, numéro 2607 in Lecture Notes in Computer Science, pages 451–462, 2003.

[dK86] J. de Kleer. An assumption-based TMS. *Artificial Intelligence*, 28 :127–162, 1986.

[DL01] S. Džeroski et N. Lavrač. An introduction to inductive logic programming. In *Relational Data Mining*, pages 48–73. Springer, 2001.

[DM02] A. Darwiche et P. Marquis. A knowledge compilation map. *Journal of Artificial Intelligence Research*, 17 :229–264, 2002.

[DMP99] C. Domingo, N. Mishra, et L. Pitt. Efficient read-restricted monotone CNF/DNF dualization by learning with membership queries. *Machine Learning*, 37(1) :89–110, 1999.

[DP92] R. Dechter et J. Pearl. Structure identification in relational data. *Artificial Intelligence*, 58 :237–270, 1992.

[DS94] R. Dechter et E. Schwalb. Compiling relational data into disjunctive structure : empirical evaluation. In *Proc. 10th Biennial Conference on Artificial Intelligence (AI'94)*. University of Calgary Press, 1994.

[dV95] A. del Val. An analysis of approximate knowledge compilation. In *Proc. 14th International Joint Conference on Artificial Intelligence (IJCAI'95)*, pages 830–836. Morgan Kaufmann, 1995.

[dV96] A. del Val. Approximate knowledge compilation : the first-order case. In *Proc. 13th National Conference on Artificial Intelligence (AAAI'96)*, pages 498–503. AAAI Press/MIT Press, 1996.

[dV00a] A. del Val. The complexity of restricted consequence finding and abduction. In *Proc. 17th National Conference on Artificial Intelligence (AAAI'00)*, pages 337–342. AAAI Press/MIT Press, 2000.

[dV00b] A. del Val. On some tractable classes in deduction and abduction. *Artificial Intelligence*, 116(1-2) :297–313, 2000.

[EG95a] T. Eiter et G. Gottlob. The complexity of logic-based abduction. *Journal of the ACM*, 42(1) :3–42, 1995.

[EG95b] T. Eiter et G. Gottlob. Identifying the minimal transversals of a hypergraph and related problems. *SIAM Journal on Computing*, 24(6) :1278–1304, 1995.

[EIM98a] T. Eiter, T. Ibaraki, et K. Makino. Computing intersections of Horn theories for reasoning with models. *Discrete Applied Mathematics*, pages 39–48, 1998.

[EIM98b] T. Eiter, T. Ibaraki, et K. Makino. Double Horn functions. *Information and Computation*, 144(2) :155–190, 1998.

[EM02] T. Eiter et K. Makino. On computing all abductive explanations. In *Proc. 18th National Conference on Artificial Intelligence (AAAI'02)*, pages 62–67. AAAI Press, 2002.

[End72] H.T. Enderton. *A mathematical introduction to logic*. Academic Press, 1972.

[Esh93] K. Eshghi. A tractable class of abduction problems. In *Proc. 13th International Joint Conference on Artificial Intelligence (IJCAI'93)*, pages 3–8. Morgan Kaufmann, 1993.

[FK96] M. Friedman et L. Kachiyan. On the complexity of dualization of monotone disjunctive normal forms. *Journal of Algorithms*, 21 :618–628, 1996.

[FW95] S. Floyd et M. Warmuth. Sample compression, learnability, and the Vapnik-Chervonenkis dimension. *Machine Learning*, 21 :269–304, 1995.

[GJ79] M.R. Garey et D.S. Johnson. *Computers and intractability : a guide to the theory of NP-completeness*. W.H. Freeman and Company, 1979.

[GPS98] G. Gogic, C.H. Papadimitriou, et M. Sideri. Incremental recompilation of knowledge. *Journal of Artificial Intelligence Research*, 8 :23–37, 1998.

[Héb94] J.-J. Hébrard. A linear algorithm for renaming a set of clauses as a Horn set. *Theoretical Computer Science*, 124 :343–350, 1994.

[HI02] T. Horiyama et T. Ibaraki. Ordered binary decision diagrams as knowledge-bases. *Artificial Intelligence*, 136 :189–213, 2002.

[HK92] P.L. Hammer et A. Kogan. Horn functions and their DNFs. *Information Processing Letters*, 44 :23–29, 1992.

[HK93] P.L. Hammer et A. Kogan. Optimal compression of propositional Horn knowledge bases : complexity and approximation. *Artificial Intelligence*, 64 :131–145, 1993.

[HK94] P.L. Hammer et A. Kogan. Essential and redundant rules in Horn knowledge bases. Rapport technique 94-42, University of New Jersey, USA, 1994.

[HK95] P.L. Hammer et A. Kogan. Quasi-acyclic propositional Horn knowledge bases : optimal compression. *IEEE Transactions on Knowledge and Data Engineering*, 7(5) :751–762, 1995.

[HLM01] A. Herzig, J. Lang, et P. Marquis. Planning as abduction. In *Proc. IJCAI'01 Workshop on Planning with Uncertainty and Incomplete Information*, 2001.

[HSAM93] J. Hobbs, M. Stickel, D. Appelt, et P. Martin. Interpretation as abduction. *Artificial Intelligence*, 63 :69–142, 1993.

[HTF01] T. Hastie, R. Tibshirani, et J. Friedman. *The elements of statistical learning*. Springer, 2001.

[HZ03] J.-J. Hébrard et B. Zanuttini. An efficient algorithm for Horn description. *Information Processing Letters*, 2003. A paraître.

[IKM99] T. Ibaraki, A. Kogan, et K. Makino. Functional dependencies in Horn theories. *Artificial Intelligence*, 108(1-2) :1–30, 1999.

[JYP88] D.S. Johnson, M. Yannakakis, et C.H. Papadimitriou. On generating all maximal independent sets. *Information Processing Letters*, 27(3) :119–123, 1988.

[Kha95] R. Khardon. Translating between Horn representations and their characteristic models. *Journal of Artificial Intelligence Research*, 3 :349–372, 1995.

[KK01] L. Kirousis et P. Kolaitis. A dichotomy in the complexity of propositional circumscription. In *Proc. 16th Annual IEEE Symposium on Logic in Computer Science (LICS'01)*, pages 71–80. IEEE Computer Society Press, 2001.

[KKS95] H. Kautz, M. Kearns, et B. Selman. Horn approximations of empirical data. *Artificial Intelligence*, 74 :129–145, 1995.

[Koz92] D.C. Kozen. *The design and analysis of algorithms*. Springer Verlag, 1992.

[KPS93] D. Kavvadias, C.H. Papadimitriou, et M. Sideri. On Horn envelopes and hypergraph transversals (extended abstract). In *Proc. 4th International Symposium on Algorithms And Computation (ISAAC'93)*, numéro 762 in Springer Lecture Notes in Computer Science, pages 399–405. Springer, 1993.

[KR94] R. Khardon et D. Roth. Exploiting relevance through model-based reasoning. In *Proc. AAAI Fall Symposium on Relevance*, pages 109–114. AAAI Press, 1994.

[KR96] R. Khardon et D. Roth. Reasoning with models. *Artificial Intelligence*, 87 :187–213, 1996.

[KR97] R. Khardon et D. Roth. Learning to reason. *Journal of the ACM*, 44(5) :697–725, 1997.

[KS98] D. Kavvadias et M. Sideri. The inverse satisfiability problem. *SIAM Journal on Computing*, 28(1) :152–163, 1998.

[KS03] D. Kavvadias et E. Stavropoulos. Monotone Boolean dualization is in co-NP[$\log^2 n$]. *Information Processing Letters*, 85 :1–6, 2003.

[KSS00] D.J. Kavvadias, M. Sideri, et E.C. Stavropoulos. Generating all maximal models of a Boolean expression. *Information Processing Letters*, 74 :157–162, 2000.

[KV00] P. Kolaitis et M. Vardi. Conjunctive-query containment and constraint satisfaction. *Journal of Computer and System Sciences*, 61(2) :302–332, 2000.

[Lib98] P. Liberatore. *Compilation of intractable problems and its application to Artificial Intelligence.* Thèse de doctorat, Université de Rome, Italie, 1998.

[LLM02] J. Lang, P. Liberatore, et P. Marquis. Conditional independence in propositional logic. *Artificial Intelligence*, 141 :79–121, 2002.

[LLM03] J. Lang, P. Liberatore, et P. Marquis. Propositional independence — formula-variable independence and forgetting. *Journal of Artificial Intelligence Research*, 18 :391–443, 2003.

[Mar00] P. Marquis. Consequence finding algorithms. In *Handbook of Defeasible Reasoning and Uncertainty Management Systems (DRUMS)*, volume 5, pages 41–145. Kluwer Academic, 2000.

[McK43] J. McKinsey. The decision problem for some classes of sentences without quantifiers. *Journal of Symbolic Logic*, 8 :61–77, 1943.

[MHI99] K. Makino, K. Hatanaka, et T. Ibaraki. Horn extensions of a partially defined Boolean function. *SIAM Journal on Computing*, 28(6) :2168–2186, 1999.

[Min88] M. Minoux. LTUR : a simplified linear-time unit resolution algorithm for Horn formulae and computer implementation. *Information Processing Letters*, 29(1) :1–12, 1988.

[Pap94] C. Papadimitriou. *Computational complexity.* Addison-Wesley, 1994.

[Pos41] E.L. Post. The two-valued iterative systems of mathematical logic. *Annals of Mathematical Studies*, 5 :1–122, 1941.

[Qui59] W.V. Quine. On cores and prime implicants of truth functions. *American Mathematical Monthly*, 66 :755–760, 1959.

[RdK87] R. Reiter et J. de Kleer. Foundations of assumption-based truth maintenance systems : preliminary report. In *Proc. 6th National Conference on Artificial Intelligence (AAAI'87)*, pages 183–188. AAAI Press/MIT Press, 1987.

[RK91] E. Rich et K. Knight. *Artificial Intelligence.* McGraw-Hill, 1991.

[RN95] S. Russel et P. Norvig. *Artificial Intelligence — A modern approach.* Prentice-Hall, 1995.

[Sch78] T.J. Schaefer. The complexity of satisfiability problems. In *Proc. 10th Annual ACM Symposium on Theory Of Computing (STOC'78)*, pages 216–226. ACM Press, 1978.

[Sch86] A. Schrijver. *Theory of linear and integer programming.* John Wiley and Sons, 1986.

[SdV01] L. Simon et A. del Val. Efficient consequence finding. In *Proc. 17th International Joint Conference on Artificial Intelligence (IJCAI'01)*, pages 359–365. Morgan Kaufman, 2001.

[Seb96] M. Sebag. Delaying the choice of bias : a disjunctive version space approach. In *Proc. 13th International Conference on Machine Learning (ICML'96)*, pages 444–452. Morgan & Kaufmann, 1996.

[SHI90] R.E. Stearns et H.B. Hunt III. Power indices and easier hard problems. *Mathematical Systems Theory*, 23 :209–225, 1990.

[SK96] B. Selman et H. Kautz. Knowledge compilation and theory approximation. *Journal of the ACM*, 43(2) :193–224, 1996.

[SL90] B. Selman et H.J. Levesque. Abductive and default reasoning : a computational core. In *Proc. 8th National Conference on Artificial Intelligence (AAAI'90)*, pages 343–348. AAAI Press, 1990.

[SW01] M Stumptner et F. Wotawa. Diagnosing tree-structured systems. *Artificial Intelligence*, 127 :1–29, 2001.

[Tar72] R.E. Tarjan. Depth first search and linear graph algorithms. *SIAM Journal on Computing*, 1 :146–160, 1972.

[UPS01] *L'Intelligence Artificielle, mais enfin, de quoi s'agit-il ?* Numéro 3 in Les livrets du service culture de l'Université Paul Sabatier. Université Paul Sabatier (Toulouse III), 2001.

[Val84] L.G. Valiant. A theory of the learnable. *Communications of the ACM*, 27(11) :1134–1142, 1984.

[Zan02a] B. Zanuttini. Approximating propositional knowledge with affine formulas. In *Proc. 15th European Conference on Artificial Intelligence (ECAI'02)*, pages 287–291. IOS Press, 2002.

[Zan02b] B. Zanuttini. Approximations of relations by propositional formulas : complexity and semantics. In *Proc. 5th Symposium on Abstraction, Reformulation and Approximation (SARA 2002)*, numéro 2371 in Lecture Notes in Artificial Intelligence, pages 242–255. Springer Verlag, 2002.

[Zan02c] B. Zanuttini. Des classes polynomiales pour l'abduction en logique propositionnelle. In *Actes des 8ièmes Journées Nationales sur la Résolution Pratique de Problèmes NP-Complets (JNPC'02)*, pages 255–268, 2002.

[Zan03a] B. Zanuttini. Approximating propositional knowledge and reasoning with affine formulas. Soumis à *Artificial Intelligence* (version longue de [Zan02a]), 2003.

[Zan03b] B. Zanuttini. New polynomial classes for logic-based abduction. *Journal of Artificial Intelligence Research*, 2003. A paraître.

[ZH02] B. Zanuttini et J.-J. Hébrard. A unified framework for structure identification. *Information Processing Letters*, 81(6) :335–339, 2002.

Index

2FNC, 38

Abductible, 144
Abduction, 143, 166, 167, 175
$aff(.)$, 49, 90
Affectation, 8
    partielle, 9, 81, 148
    représentation d'une –, 17
Affine, 41
Algorithmes
    `ApprendreAffine`, 132
    `DecrFNC`, 84
    `DecrIP`, 87
    `Expliquer`, 152
    `GLB[.]`, 116
    `GLB-Ren[.]`, 118
    `MAJAffine`, 130
    `ProjAffine`, 157
    `SIM`, 96
Anti-Horn, 40
Anti-Horn définie positive, 40
Apprentissage, 125
Approximation, 101
    avec le nombre maximal de modèles, 120
    et raisonnement, 163
    faible minimale (LUB), 102, 167, 174
    forte maximale (GLB), 102, 165
    vision confiante, 103, 128
    vision méfiante, 104, 163, 167
    vision restrictive, 105, 163, 167
Arbre de décision, 17, 82

Base (d'un espace vectoriel), 46
Base de connaissances, 18
Bijonctive, 38
But, 144

Cardinal ($|.|$), 7
Classe duale, 40
Classes de formules, 27, 175, 177
    Affine, 41
    Anti-HDP, 40
    Anti-Horn, 40
    Bij, 38
    Bij-FND, 60
    Clauses, Termes, 10
    DisjAffine, 60
    Fnc, Fnd, 10
    HDP, 34
    HDP-FND, Anti-HDP-FND, 60
    HDP-Ren, 57
    HDP-Ren-FND, 60
    Horn, 31
    Horn-FND, Anti-Horn-FND, 60
    Horn-Ren, 56
    Horn-Ren-FND, 60
    Monot, 58
    Monot-FND, 60
    multivaluées, 178
    Nég, 36
    Nég-FND, Pos-FND, 60
    Pos, 40
    Vars, Lits, 10
Clause, 10
    première, 12, 86
Clos par négation (clos par $\neg$), 157
Cohérence (vérification de –), 138
Compilation, 79, 105, 164
Complexité, 14, 176
    classes de –, 15, 146
    d'énumération, 16
Concept, 19
Connecteurs
    conjonction ($\wedge$), 7
    disjonction ($\vee$), 7
    équivalence logique ($\leftrightarrow$), 7
    implication ($\rightarrow$), 7
    négation ($\neg$), 7
    ou exclusif ($\oplus$), 7

Contraintes, 27, 63

Décrire, 8
Déduction, 138, 140, 165
Dépendance en une variable, 50, 89, 91
Descriptible, 8
Description, 77

Efficace, 14, 16
Ensemble des modèles ($\mathcal{M}(.)$), 8
Equation linéaire, 41
Equivalence logique, 9
Espace vectoriel, 45
Explication, 144
Expressivité, 28, 67, 175–177
Extension
    d'affectations partielles ($ext(.)$), 9
    en –, 18, 79

Forme Normale
    Conjonctive (FNC), 10, 26
    Disjonctive (FND), 10, 27
Formes normales, 10, 11, 25
Formule
    multivaluée, 178
    première, 12, 86
    propositionnelle, 7
    représentation d'une –, 16

Horn, 31
Horn définie positive, 34
Horn définie positive renommable, 57
Horn-renommable, 56
Hypothèse, 144

Identification, 77
Impliquant premier, 13
Impliqué premier, 12, 86
Impliquer ($\models$), 9
Intension (en –), 18, 79

Linéairement indépendant, 46
Littéral, 10
    abductible, 144
    opposé, 10, 141

Logique propositionnelle, 7, 176
Logiquement
    équivalent ($\equiv$), 9
    plus faible, 9, 102
    plus fort, 9, 102

Machine RAM, 14, 16
Modèle, 8
Monotone, 58

Nature, 28
$Neg(.)$, 11
NÉG-renommable, 58
Négative, 36

Opération élémentaire, 14
Oubli de littéraux, 148

PAC-apprenable, 127
PAC-apprentissage, 125
Petit monde, 18
Pivot de Gauss, 43
Positive, 40
Primalité, 12
Problèmes
    `Abduction[.,.]`, 145
    `Abduction-C[.,.]`, 157
    avec sortie, 14
    `D-Abduction[.,.]`, 146
    de décision, 14
    de décision associés, 16
    `Deduction[.,.]`, 140
    `Description[.]`, 78
    `D-Max-GLB[.]`, 120
    `GLB[.]`, 116
    `Identification[.]`, 78
    `Inv-SAT[.]`, 66
    `LUB[.]`, 110
    `Max-GLB[.]`, 120
    NP-complets, 15
    NP-difficiles, 15
    `PAC-apprentissage[.]`, 126
    `SAT`, 15
    `SAT[.]`, 28
    `SAT-Gen[.]`, 65
    `TAUT`, 60

Projection, 148, 157, 159
Projeté, 148
Prolongement, 8
Propagation d'une affectation, 11

Raisonnement, 137
    et approximation, 163
Reconnaissable, 28
Réduction, 15
Renommage, 53
Renommé, 54
Requête, 137, 138, 144
Résoluble, 13
Résolution, 13
    unitaire, 14
Résolvant, 13

S-formule, 63
Satisfaire ($\models$), 8
Satisfaisable, 10
Situation envisageable, 18
Stabilité par conjonction, 26
Structure, 28
Subsumer ($\subseteq$), 13
Système d'équations, 41

Tautologique, 10
Témoin, 15
Terme, 10
    premier, 13
Traitable, 14, 16

$Var(.)$, 7
Variable booléenne, 7