Université d'Artois Faculté des sciences Jean Perrin

Raisonnement en présence d'incohérence : de la compilation de bases de croyances stratifiées à l'inférence à partir de bases de croyances partiellement préordonnées

THÈSE

pour l'obtention du grade de

Docteur de l'Université d'Artois
(spécialité informatique)

par

Safa Yahi-Mechouche

devant le jury composé de

Pascal Nicolas	Professeur des Universités, Université d'Angers	(rapporteur)
Henri Prade	Directeur de recherche, Université Paul Sabatier	(rapporteur)
Salem Benferhat	Professeur des Universités, Université d'Artois	(directeur de thèse)
Habiba Drias	Professeur des Universités, USTHB	(co-directrice de thèse)
Sylvain Lagrue	Maître de Conférences, Université d'Artois	(examinateur)

Centre de Recherche en Informatique de Lens (CRIL)

Table des matières

Introduction générale

Contexte et motivations

La représentation des connaissances et le raisonnement de sens commun à partir de celles-ci constituent l'épine dorsale de l'Intelligence Artificielle. En supposant que ces connaissances sont certaines, complètes et donc cohérentes, la logique classique constitue un outil très satisfaisant d'un point de vue expressivité. Néanmoins, en réalité, les connaissances dont on dispose ne sont pas toujours aussi parfaites. En effet, l'incohérence survient dans de nombreux contextes, à l'image du raisonnement tolérant les exceptions, le raisonnement avec incertitude, la révision de croyances, la fusion d'informations issues de différentes sources qui peuvent être conflictuelles, etc. Dans ce cas, la logique classique devient très vite obsolète car l'application de l'inférence classique en présence d'incohérence conduit à une trivialisation, dans le sens où elle permet de déduire n'importe quoi (*principe d'ex falso quodlibet sequitur*).

Afin de permettre de raisonner en présence d'incohérence sans trivialisation, différentes approches ont été mises au point. D'une manière générale, ces approches peuvent être scindées en deux grandes familles. D'une part, on a celles qui affaiblissent la relation d'inférence classique, à l'instar des logiques paraconsistantes [32] et des approches argumentatives [10, 11]. D'autres part, on a les approches qui affaiblissent les croyances elles mêmes telles que les approches basées sur la restauration de la cohérence.

Les approches basées sur la restauration, auxquelles nous nous intéressons dans cette thèse, consistent à générer dans un premier temps des sous-bases cohérentes préférées, ensuite à appliquer l'inférence classique sur ces sous-bases. Ainsi, de telles approches, de par leur principe, sont qualifiées de très naturelles. En outre, elles permettent de tirer profit de toute la machinerie développée dans le cadre de la logique classique. D'où la popularité des approches basées sur la restauration de la cohérence en termes de raisonnement en présence d'incohérence. Maintenant, parmi ces approches, on peut distinguer celles qui sont définies par rapport à des bases de croyances totalement préordonnées ou stratifiées, et celles qui sont dédiées à des bases de croyances partiellement préordonnées.

Dans le cadre de bases de croyances totalement préordonnées, les croyances, c'est-à-dire les connaissances incertaines, sont munies d'un préordre total qui est une relation réflexive et transitive, via laquelle toutes les croyances sont deux à deux comparables. Les relations d'inférence les plus fréquemment utilisées dans ce cas sont l'inférence possibiliste [45], l'inférence linéaire [79], l'inférence lexicographique [9, 67] ainsi que l'inférence relative à la préférence pour l'inclusion [21, 28]. Sur le plan pratique, ces relations d'inférence suscitent un intérêt grandissant. En effet, elles trouvent de plus en plus d'applications dans de nombreux domaines, à l'image de la sécurité informatique. Dans ce contexte, citons à titre d'exemple, les modélisations logiques des politiques de sécurité informatique, où les règles de contrôle d'accès présentent souvent des exceptions ce qui génère des conflits [6]. Ces modélisations s'appuient sur des approches basées sur la restauration de la cohérence à

partir de bases de croyances stratifiées. .

Toutefois, d'un point de vue calculatoire, l'expressivité additionnelle offerte par ces relations d'inférence s'accompagne d'un saut de complexité. En effet, elles sont plus coûteuses que l'inférence classique, qui est déjà un problème calculatoirement épineux. Plus précisément, dans le cas où les croyances sont décrites en logique propositionnelle (et non pas en logique du premier ordre), ces relations d'inférence nécessitent au moins un nombre d'appels logarithmique à un oracle NP, et au pire des cas un nombre d'appels exponentiel à un oracle NP. Clairement, cet aspect est très limitatif en pratique, notamment dans des contextes où le temps de réponse constitue un facteur prépondérant. Par exemple, pour la modélisation des politiques de contrôle d'accès dans un contexte médical, il est crucial qu'un médecin puisse avoir rapidement le droit d'accès au dossier médical d'un de ses patients en cas d'urgence.

Par ailleurs, la compilation de connaissances [23, 37, 88, 74] est parmi les approches les plus prometteuses qui permettent d'appréhender les problèmes liés à la complexité du raisonnement logique. En effet, cette approche, qu'est en plein expansion, s'est avérée assez efficace pour traiter de nombreuses instances de problèmes en pratique. Le principe sous-jacent est d'effectuer un prétraitement sur la base de connaissances en vue de générer hors ligne une base compilée à partir de laquelle le raisonnement en ligne devient plus efficace par rapport au raisonnement depuis la base initiale, voire même polynomial. Donc, l'idée directrice est de déplacer la surcharge de travail de la phase en ligne vers la phase hors ligne. Comme la phase hors ligne n'est appelée qu'une seule fois, l'effort déployé lors de la génération de la base compilée sera amorti par les multiples phases en ligne.

Actuellement, on dispose d'un répertoire considérable de langages cibles de compilation dans le cadre de la logique propositionnelle, qui sont dérivés du langage NNF (pour Negation Normal Form), et qui sont issus de travaux en Intelligence Artificielle, en vérification formelle et en informatique de manière générale [37]. Ces langages diffèrent d'abord relativement aux opérations logiques (telles que la déduction de clauses, l'oubli de variables, le test d'équivalence, etc) qu'ils permettent d'accomplir en temps polynomial. Ils se distinguent aussi en termes de leur efficacité spatiale connue sous le nom de *concision*. Ainsi, les langages DNNF (pour Decomposable Negation Normal Form) et les impliqués premier PI (pour Prime Implicates) revêtent une importance capitale parmi les langages cibles de compilation. En effet, ces langages permettent d'effectuer un bon nombre d'opérations logiques suffisantes pour de nombreuses applications. De plus, il est connu que DNNF est plus concis que tous les autres langages cibles de compilation excepté PI. En revanche, on sait que PI n'est pas plus concis que DNNF mais on ignore jusqu'à présent l'inverse.

Or, l'application directe de ces langages sur des bases incohérentes mène, tout comme l'inférence classique, à une trivialisation. Par exemple, la compilation DNNF d'une base incohérente se résume à la contradiction. Par conséquent, d'autres approches de compilation en présence d'incohérence, notamment pour l'inférence à partir de bases de croyances

stratifiées, ont été proposées tout en exploitant un des langages de compilation classique. Ces approches comprennent essentiellement la compilation proposée par Coste-Marquis et Marquis [31] et celle introduite par Benferhat et Prade [15] qui se basent sur le langage DNF, ainsi que celle développé par Cayrol, Lagasquie-Schiex et Schiex [27] qui s'appuie sur le langage OBDD. Néanmoins, en compilation classique, les langages DNF et OBDD sont loins d'être en tête de liste par rapport au facteur concision. D'où la pertinence de développer de nouvelles approches de compilation à partir de bases de croyances stratifiées qui tirent profit de langages plus concis, notamment DNNF et PI, qui sont complémentaires dans ce sens.

Par ailleurs, la deuxième catégorie des relations d'inférence basées sur la restauration de la cohérence, concerne les bases de croyances partiellement pré-ordonnées. En effet, dans ceratines applications, les informations proviennent de plusieurs sources hétérogènes, indépendantes et souvent entachées d'incertitude. Alors, préordonner totalement ces informations dans ce cas n'est pas naturel, voire même contre intuitif, et de plus mène souvent à des résultats qui sont trop aventureux ou risqués, et donc qui ne sont pas souhaitables.

Ainsi, les bases de croyances partiellement préordonnées offrent plus de flexibilité que les bases stratifiées dans de nombreuses situations, afin de représenter des informations incomplètes, et ce en permettant d'éviter de comparer des informations incomparables, ou des informations dont on ignore le degré de priorité. Le besoin de raisonner en présence d'incohérence à partir de bases de croyances partiellement préordonnées est d'autant plus important si l'on considère la dynamique des croyances. En effet, même si les informations disponibles sont totalement préordonnées, il se peut que lors de l'utilisation de certains opérateurs de mise à jour [63], l'intégration d'une nouvelle information mène à préordonner partiellement la base de croyances en question. Enfin, de telles bases peuvent être utilisées dans une situation où l'on a plusieurs agents qui partagent les mêmes croyances mais où chacun définit son propre préordre total sur ces mêmes croyances, et donc la fusion de toutes ces bases de croyances totalement préordonnées génère une base de croyances partiellement préordonnées.

Plusieurs relations d'inférence à partir de bases de croyances partiellement préordonnées ont été proposées. En général, ce sont des extensions des relations d'inférence à partir de bases de croyances stratifiées, telles que l'inférence de l'inclusion basée-compatibles [21, 62] et l'inférence démocratique [28] qui étendent l'inférence de l'inclusion. On a aussi les inférences possibilistes forte et faible [5] qui généralisent l'inférence possibiliste.

Néanmoins, aucune extension n'a été définie vis-à-vis de l'inférence lexicographique en dépit des caractéristiques intéressantes dont elle jouit. Par exemple, une analyse psychologique conduite conjointement par un chercheur en Intelligence Artificielle et des psychologues [8] a révélé que l'inférence lexicographique présente de fortes similarités avec le raisonnement humain non-monotone. En particulier, les participants à ce test psychologique ont clairement manifesté une sensibilité à la majorité qui est au cœur de l'in-

férence lexicographique.

Par ailleurs, étant donnée une application où les croyances sont partiellement préordonnées, on se demande légitimement quelle relation d'inférence choisir. En fait, une réponse naturelle à une telle question consiste à prendre en compte trois critères clés, à savoir la complexité qui reflète le coût induit par cette relation d'inférence, ses propriétés logiques qui décrivent à quel point le comportement de cette relation est acceptable, en plus du facteur prudence qui désigne le nombre de conclusions déductibles via cette même relation d'inférence. Néanmoins, les relations d'inférence à partir de bases de croyances partiellement préordonnées existantes, à l'inverse des relations d'inférence à partir de bases de croyances stratifiées, n'ont pas été suffisamment analysée relativement à ces trois critères prépondérants.

Contributions

Dans cette thèse, nous nous intéressons donc aux approches basées sur la restauration de la cohérence de manière générale, c'est-à-dire à la fois aux relations d'inférence à partir de bases de croyances stratifiées et à partir de bases de croyances partiellement préordonnées.

Dans le cadre de relations d'inférence à partir de bases de croyances stratifiées, nous proposons trois nouvelles approches de compilation que nous qualifions de flexibles dans le sens où elles peuvent être paramétrées par n'importe quel langage cible de compilation. La première concerne l'inférence en logique possibiliste (inférence simple, pondérée et conditionnement possibiliste) et s'adapte facilement à l'inférence linéaire. Cette approche est proche de celle proposée par Benferhat et Prade [15] relativement au codage propositionnel qu'elle utilise. Néanmoins, la nôtre se trouve paramétrée par n'importe quel langage cible de compilation, entre autres DNNF et PI contrairement à l'autre approche qui est basée uniquement sur DNF.

La seconde approche que nous proposons se rapporte à l'inférence lexicographique. Elle se base sur la notion de contraintes de cardinalité Booléennes, et elle est complémentaire par rapport aux approches existantes en étant aussi paramétrée par n'importe quel langage cible de compilation.

Enfin, nous introduisons une nouvelle méthode de compilation par rapport à l'inférence MSP (pour Minimum de Specificity Principle) qui est parmi les approches naturelles de raisonnement en présence d'exceptions. Cette compilation est particulièrement adaptée à des théories des défauts fréquemment mises à jour. Cette compilation nous permet de calculer efficacement à la fois la base stratifiée associée ainsi que l'inférence possibiliste à partir de cette dernière. Par conséquent, une telle compilation permet d'effectuer l'inférence MSP en temps polynomial.

Notons que les approches que nous introduisons, comme les approches existantes, supposent que les croyances sont décrites en logique propositionnelle, sans aller à la logique du premier ordre. En effet, cette logique bien que simple, elle se trouve suffisamment expressifve pour de nombreuses applications telles que le diagnostic. De plus, le processus d'inférence en logique propositionnelle est un problème décidable contrairement au fragment complet de la logique des prédicats du premier ordre.

En ce qui concerne le raisonnement à partir de bases de croyances partiellement préordonnées, notre première contribution consiste en l'introduction d'une extension de l'inférence lexicographique classique. En particulier, nous proposons deux nouvelles relations d'inférence basées sur la cardinalité à partir de bases de croyances partiellement préordonnées. Ces deux relations sont assez naturelles, et étendent l'inférence lexicographique classique. La première, consiste à appliquer l'inférence lexicographique classique sur toutes les bases totalement préordonnées compatibles. Quant à la seconde, qui se trouve plus prudente, elle revient à appliquer l'inférence classique sur les sous-bases cohérentes qui sont préférées par rapport à une nouvelle relation de préférence lexicographique. L'intuition derrière cette nouvelle relation lexicographique est qu'une sous-base cohérente est préférée à une autre sous-base cohérente par rapport à une base de croyances partiellement préordonnées, si et seulement si elle lui est préférée lexicographiquement de manière classique relativement à toute base de croyances stratifiées compatible. En outre, cette relation admet une définition équivalente qui ne nécessite pas le calcul de toutes les sous-bases stratifiées compatibles.

La seconde contribution dans ce même cadre, est l'étude comparative des différentes relations d'inférence à partir de bases de croyances partiellement préordonnées relativement aux facteurs de complexité, de propriétés logiques et de prudence. Cette étude comparative gravite autour des deux relations d'inférence lexicographiques que nous introduisons, ainsi que les extensions de l'inférence possibiliste à savoir l'inférence possibiliste forte et l'inférence possibiliste faible, et les extensions de l'inférence de l'inclusion telles que l'inférence démocratique et l'inférence relative à la préférence pour l'inclusion basée-compatibles.

Une autre contribution dans cette thèse se rapporte à l'application du raisonnement en présence d'incohérence, en particulier à partir de bases de croyances partiellement préordonnées dans le cadre de la détection d'intrusions coopérative. L'idée derrière la détection d'intrusion coopérative est de faire coopérer plusieurs dispositifs (systèmes de détection d'intrusions (IDSs), analyseurs réseaux, etc) afin d'avoir une vision globale du réseau, en vue d'améliorer le processus de surveillance.

Cependant, ces dispositifs ne sont pas complètement fiables. Par exemple, les IDSs souffrent souvent des problèmes des fausses alertes (faux positifs) qui correspondent à des attaques qui n'ont pas eu lieu et qui occupent l'administrateur au détriment des vraies attaques. Les IDSs souffrent également des problèmes des attaques qui passent inaperçues (faux négatifs). En ce qui concerne les analyseurs réseaux, on a souvent affaire à des

informations incomplètes, voire même incohérentes parfois par rapport à la réalité car elles sont récoltées en général hors ligne étant donné le coût de cette opération. Ceci implique que la dynamique du réseau n'est pas correctement prise en compte à tout moment. Par conséquent, on peut facilement être confronté à des situations incohérentes.

D'autre part, la relation de fiabilité entre ces dispositifs est partiellement définie. En effet, certaines d'entre eux sont comparables. Par exemple, certaines méthodes expérimentales permettent d'évaluer et de comparer des IDSs [50]. Par contre, d'autres dispositifs sont incomparables : comparer un IDS et un scanner réseau n'a pas de sens, surtout si on prend en compte la dynamique du système.

Nous proposons alors une nouvelle approche de corrélation d'alertes, en vue d'améliorer le processus de la détection d'intrusions. Cette approche se situe dans un contexte coopératif, et se base sur le raisonnement à partir de bases de croyances partiellement préordonnées. Par ailleurs, en détection d'intrusion, les informations que l'on manipule sont de nature structurée, la logique propositionnelle n'est pas vraiment propice pour exprimer de telles informations. D'où la nécessite d'aller au delà de la logique propositionnelle en termes d'expressivité pour une telle application. Ainsi, nous proposons d'utiliser les logiques de description [3] qui sont en revanche bien adaptées à la représentation de telles connaissances, tout en restant décidables comparées au fragment entier de la logique du premier ordre.

Guide de lecture

Ce mémoire se veut structuré en deux parties. La première partie est consacré à l'état de l'art. Plus précisément, dans le chapitre 1, nous rappelons brièvement les aspects syntaxiques ensuite sémantiques de la logiques propositionnelles, qui sont nécessaires pour la compréhension de la suite du mémoire. Le chapitre 2 fournit une brève introduction à la théorie de la complexité. Dans ce chapitre, nous présentons d'abord les notions de base de la complexité ainsi que le modèle des machines de Turing. Ensuite, nous esquissons les classes de complexité classiques avant d'aborder la hiérarchie polynomiale. Nous rappelons dans le chapitre 3 les systèmes de propriétés logiques selon Kraus, Lehmann et Magidor [64, 69] tels que le système cumulatif, le système cumulatif avec boucle, le système préférentiel et le système rationnel. Ensuite, dans le chapitre 4, nous passons en revue un bon nombre d'approches basées sur la restauration de la cohérence par rapport à des bases de croyances totalement préordonnées, à savoir la logique possibiliste, l'inférence linéaire, l'inférence lexicographique ainsi que l'inférence relative à la préférence pour l'inclusion. Nous rappelons ensuite les extensions de certaines de ces approches dans le cadre de bases de croyances partiellement préordonnées, notamment les inférences possibilistes forte et faible, l'inférence démocratique et l'inférence relative à la préférence pour l'inclusion basée-compatibles. Quant au chapitre 5, il est consacré à une introduction à

la compilation de connaissances. Après la présentation du principe correspondant, nous esquissons les méthodes classiques de compilation exactes, en particulier, celles basées sur les impliqués premiers ainsi que les méthodes classiques de compilation approximative. Ensuite, nous explorons la carte de compilation proposée par Darwiche et Marquis dans [37] qui rassemble un bon nombre de langages cibles de compilation, et les compare par rapport à leur concision et aux opérations logiques qu'ils permettent d'effectuer en temps polynomial. Enfin, nous rappelons les principales approches de compilation destinées à l'inférence à partir de bases de croyances stratifiées.

La deuxième partie de ce mémoire englobe les contributions de cette thèse en les décrivant à travers six chapitres. Dans le chapitre 6, nous présentons notre approche de compilation de bases de croyances stratifiées relativement à l'inférence possibiliste et à l'inférence linéaire. Ensuite, dans le chapitre 7, nous décrivons notre approche de compilation de l'inférence lexicographique basée sur les contraintes de cardinalité Booléennes, et ce après avoir rappelé l'algorithme proposé dans [27] pour le calcul de l'inférence lexicographique ainsi que les contraintes de cardinalité Booléennes. Dans le chapitre 8, nous présentons une nouvelle méthode de compilation de théories des défauts par rapport à l'inférence MSP à la suite d'un rappel concernant le traitement possibiliste des théories des défauts. Le chapitre 9 est consacré aux deux extensions de l'inférence lexicographique à partir de bases de croyances partiellement préordonnées que nous proposons. Nous commençons alors par la définition d'une première relation d'inférence qui revient à appliquer l'inférence lexicographique classique à partir de toutes les bases totalement préordonnées compatibles. Par la suite, nous définissons la seconde relation d'inférence qui consiste à appliquer l'inférence classique sur les sous-bases cohérentes qui sont préférées par rapport à une nouvelle relation de préférence lexicographique. Le chapitre suivant se rapporte à l'étude des relations d'inférence à partir de bases de croyances partiellement préordonnées par rapport aux critères de complexité, propriétés logiques et prudence. Le dernier chapitre des contribution se rapporte à l'application de la gestion de l'incohérence en détection d'intrusions coopérative. Nous donnons dans un premier temps les notions de base en détection d'intrusions. Par ailleurs, comme nous avons choisi les logiques de description comme outil de représentation des différentes informations, nous fournissons une brève introduction aux logiques de description.

Enfin, le dernier chapitre de ce mémoire fait l'objet d'une conclusion générale où nous synthétisons les différents résultats obtenus dans cette thèse. Nous décrivons également les perspectives ouvertes et les nouvelles pistes à explorer.

Première partie

État de l'art

Chapitre 1

Logique propositionnelle

1.1 Introduction

La logique propositionnelle constitue un formalisme de représentation de connaissances et de raisonnement simple mais qui se trouve suffisamment expressif pour de nombreuses applications. En effet, de multiples problèmes pratiques peuvent se représenter de manière simple et se résoudre en logique propositionnelle, à l'image du diagnostic par exemple. Par ailleurs, la logique propositionnelle est la pierre angulaire de nombreuses logiques plus sophistiquées. En outre, le processus d'inférence en logique propositionnelle est un problème décidable contrairement au fragment complet de la logique des prédicats du premier ordre. Autrement dit, la logique propositionnelle garantit d'effectuer le raisonnement en un temps fini, ce qui est prépondérant d'un point de vue pratique pour de nombreuses applications.

Notre but dans ce chapitre n'est pas de faire un état de l'art complet sur la logique propositionnelle, mais plutôt de présenter succinctement ses aspects syntaxiques ensuite sémantiques qui sont nécessaires pour la compréhension de ce mémoire.

1.2 Syntaxe

D'un point de vue syntaxique, l'élément atomique est la notion de variable propositionnelle.

Définition 1 *Une **variable propositionnelle** (appelée également **proposition** ou **atome**) est une variable booléenne susceptible de prendre deux valeurs de vérité : Vrai ou Faux.*

Soient les **connecteurs logiques** suivants :
- la négation : $\neg$,
- la conjonction : $\wedge$,
- la disjonction : $\vee$,
- l'implication matérielle : $\Rightarrow$,
- l'équivalence : $\Leftrightarrow$.

Les connecteurs $\wedge$, $\vee$, $\Rightarrow$ et $\Leftrightarrow$ sont de même priorité, l'ordre d'évaluation étant ainsi déterminé par les parenthèses. Le connecteur de négation est considéré, quant à lui, comme prioritaire sur tous les autres connecteurs.

Soit V un ensemble fini de variables propositionnelles notées par les lettres romaines minuscules $a, b, \ldots$ Alors, $PROP_V$ désigne le langage propositionnel construit à partir des variables de V, des constantes booléennes $\top$ et $\bot$, des connecteurs logiques et des parenthèses. Formellement :

Définition 2 *Soit V un ensemble fini de variables propositionnelles. $PROP_V$ est le langage propositionnel défini inductivement par :*
- *$\top$ et $\bot \in PROP_V$,*
- *si $x \in V$ alors $x \in PROP_V$,*
- *si $\varphi \in PROPV$ alors (φ), $\neg\varphi \in PROP_V$,*
- *si φ et $\psi \in PROP_V$ alors $\varphi \wedge \psi$, $\varphi \vee \psi$, $\varphi \Rightarrow \psi$ et $\varphi \Leftrightarrow \psi \in PROP_V$.*

*Les éléments de $PROP_V$ sont appelés **formules** (ou **formules propositionnelles**).*

Exemple 1 *Soient $a, b, c, d \in V$. Alors, $(a \vee b) \wedge ((c \Rightarrow d) \wedge \neg a)$ est une formule appartenant à $PROP_V$.*

Définition 3 *Un **littéral** l est soit une variable propositionnelle x (on parle de **littéral positif**), soit sa négation $\neg x$ (on parle alors de **littéral négatif**). De plus, $\neg l$ représente le **littéral complémentaire** de l.*

L_V (resp. L_V^+, L_V^-) dénote l'ensemble des littéraux (resp. positifs, négatifs) construits à partir de V.

Définition 4 *Un littéral l est dit **pur** (ou **monotone**) pour une formule φ si l apparaît dans φ et $\neg l$ n'y apparaît pas.*

En se basant sur la notion de littéral, les clauses et les termes sont définis comme suit :

Définition 5 *Une **clause** est une disjonction finie de littéraux (en particulier la constante $\bot$, lorsque l'ensemble des littéraux est vide).*

Définition 6 *Un **terme** (**monôme** ou encore **cube**) est une conjonction finie de littéraux (en particulier la constante $\top$, lorsque l'ensemble des littéraux est vide).*

Exemple 2 *Soient a et $b \in V$.*
- *La formule $a \vee \neg b$ est une clause.*
- *La formule $\neg a \wedge b$ est un terme.*

Plusieurs types de clauses existent. Par exemple, on peut distinguer les clauses unaires, positives, négatives, de Horn et reverse-Horn.

Définition 7
- *Une **clause unitaire** ou **unaire** est une clause qui contient un seul littéral.*
- *Une **clause de Krom** est une clause qui contient au plus deux littéraux.*

- *On appelle* ***clause positive*** (*resp.* ***négative***) *une clause qui ne contient que des littéraux positifs* (*resp. négatifs*).
- *Une* ***clause fondamentale*** (*resp.* ***terme fondamental***) *est une clause* (*resp. terme*) *qui ne contient pas de littéraux complémentaires.*
- *Une* ***clause de Horn*** *est une clause qui contient au plus un littéral positif.*
- *Une* ***clause reverse-Horn*** *est une clause qui contient au plus un littéral négatif.*

Exemple 3 *Soient* $a, b, c \in V$.

- a *est une clause unitaire.*
- $a \vee b$ *est une clause de Krom.*
- $a \vee \neg b \vee \neg c$ *est une clause de Horn.*
- $a \vee b \vee \neg c$ *est une clause reverse-Horn.*
- *Les clauses négatives sont des clauses de Horn.*
- *Les clauses positives sont des clauses reverse-Horn.*
- $a \vee b \vee \neg c$ *est une clause fondamentale.*

Définition 8 *Deux clauses* c_1 *et* c_2 ***se résolvent*** *si et seulement s'il existe un littéral* l *tel que* $l \in c_1$ *et* $\neg l \in c_2$. *La clause* $((c_1 \setminus l) \cup (c2 \setminus \neg l))$ *est appelée* ***résolvante****, plus précisément résolvante en* l *de* c_1 *et* c_2.

Exemple 4 *Les clauses* $(a \vee b)$ *et* $(\neg b \vee c)$ *se résolvent en la clause* $a \vee c$.

Étant donnée une formule propositionnelle φ, $Var(\varphi)$ (resp. $Lit(\varphi)$) désigne l'ensemble des variables (resp. littéraux) qui apparaissent dans φ.

Définition 9 *Une clause* c_1 ***subsume*** *ou* ***sous-somme*** *une clause* c_2 *si et seulement si* $Lit(c_1) \subseteq Lit(c_2)$.

Exemple 5 $a \vee b$ *subsume* $a \vee b \vee \neg c$.

Définition 10

- *Une formule* φ *est sous* ***forme normale conjonctive*** (***CNF***) *si et seulement si* φ *est une conjonction de clauses.*
- *Une formule* φ *est sous* ***forme normale disjonctive*** (***DNF***) *si et seulement si* φ *est une disjonction de termes.*
- *Une formule de* ***Horn CNF*** (*resp.* ***reverse-Horn CNF***) *est une formule CNF composée uniquement de clauses de Horn* (*resp. reverse-Horn*).

Exemple 6 *Soient* $a, b, c, d \in V$.

- $(a \vee b) \wedge (\neg c \vee d)$ *est une formule CNF.*
- $(a \wedge b) \vee (\neg c \wedge d)$ *est une formule DNF.*

– $(\neg a \vee \neg b) \wedge (\neg c \vee d)$ *est une formule de Horn CNF.*
– $(a \vee b) \wedge (\neg c \vee d)$ *est une formule reverse-Horn CNF.*

Une base de connaissances est un ensemble fini de formules logiques que l'on assimile généralement à la conjonction de ses formules lorsqu'elle est cohérente.

1.3 Sémantique

D'un point de vue sémantique, la notion de base en logique propositionnelle est celle d'une interprétation.

Définition 11 *Une* ***interprétation*** *est une application de l'ensemble des formules propositionnelles considérées dans l'ensemble des valeurs de vérité* $\{Vrai, Faux\}$*. Étant données deux formules* φ *et* ψ*, une interprétation* I *vérifie ce qui suit :*

– $I(\top) = Vrai$;
– $I(\bot) = Faux$;
– $I(\neg\varphi) = Vrai$ *si et seulement si* $I(\varphi) = Faux$;
– $I(\varphi \wedge \psi) = Vrai$ *si et seulement si* $I(\varphi) = I(\psi) = Vrai$;
– $I(\varphi \vee \psi) = Faux$ *si et seulement si* $I(\varphi) = I(\psi) = Faux$;
– $I(\varphi \Rightarrow \psi) = Faux$ *si et seulement si* $I(\varphi) = Vrai$ *et* $I(\psi) = Faux$;
– $I(\varphi \Leftrightarrow \psi) = Vrai$ *si et seulement si* $I(\varphi) = I(\psi)$.

La table suivante synthétise les valeurs de vérité des formules $\neg\varphi$, $\varphi \wedge \psi$, $\varphi \vee \psi$, $\varphi \Rightarrow \psi$ et $\varphi \Leftrightarrow \psi$ en fonction de celles de φ et ψ.

$I(\varphi)$	$I(\psi)$	$I(\neg\varphi)$	$I(\varphi \wedge \psi)$	$I(\varphi \vee \psi)$	$I(\varphi \Rightarrow \psi)$	$I(\varphi \Leftrightarrow \psi)$
$Faux$	$Faux$	$Vrai$	$Faux$	$Faux$	$Vrai$	$Vrai$
$Faux$	$Vrai$	$Vrai$	$Faux$	$Vrai$	$Vrai$	$Faux$
$Vrai$	$Faux$	$Faux$	$Faux$	$Vrai$	$Faux$	$Faux$
$Vrai$	$Vrai$	$Faux$	$Vrai$	$Vrai$	$Vrai$	$Vrai$

Définition 12 *Soit* φ *une formule propositionnelle.*
– *On dit qu'une interprétation* I *est un* ***modèle*** *de* φ *si et seulement si* $I(\varphi) = Vrai$*. On dit aussi que* I ***satisfait*** φ*.*
– *On dit qu'une interprétation* I *est un* ***contre-modèle*** *de* φ *si et seulement si* $I(\varphi) = Faux$*. On dit aussi que* I ***falsifie*** φ*.*

Les concepts de cohérence, incohérence, validité et invalidité d'une formule sont donnés par la définition suivante :

Définition 13 – *Une formule est dite* ***cohérente*** (*ou* ***satisfiable***) *si et seulement si elle admet au moins un modèle.*
– *Une formule est dite* ***incohérente*** (*ou* ***insatisfiable***) *si et seulement si elle n'admet pas de modèle.*
– *Une formule est dite* ***valide*** (***universellement valide*** *ou* ***tautologie***) *si et seulement si elle n'admet pas de contre-modèle.*
– *Une formule est dite* ***invalide*** *si et seulement si elle admet un contre-modèle.*

Exemple 7 *Soient a et b deux propositions.*
– *La formule $a \vee \neg a \vee b$ est valide.*
– *La formule $a \wedge b$ est invalide mais elle est cohérente.*
– *La formule $a \wedge b \wedge (\neg a \vee \neg b)$ est incohérente.*

La relation d'inférence ou de déduction en logique propositionnelle est définie comme suit :

Définition 14 *Une formule φ* ***implique sémantiquement*** *une formule ψ, noté par $\varphi \models \psi$, si et seulement si tout modèle de φ est un modèle de ψ. On dit aussi que ψ est une* ***conséquence logique*** *de φ.*

Exemple 8 *Par exemple, on a $a \wedge b \models a$ et $a \models a \vee b$. Par contre, $a \vee b \not\models a$.*

Définition 15 *Deux formules φ et ψ sont dites* ***logiquement équivalentes****, noté par $\varphi \equiv \psi$, si et seulement si $\varphi \models \psi$ et $\psi \models \varphi$, c'est-à-dire elles admettent exactement les mêmes modèles.*

Étant données deux formules φ et ψ, un résultat fondamental en démonstration automatique (preuve par l'absurde) stipule que :

$$\varphi \models \psi \text{ si et seuelemnt si } \varphi \wedge \neg\psi \text{ est incohérente.}$$

Enfin, nous rappelons les notions d'impliqués, d'impliquants, d'impliqués premiers et d'impliquants premiers.

Définition 16 *Une clause γ est un* ***impliqué*** *d'une formule φ si et seulement si $\varphi \models \gamma$. Cette clause γ est dite* ***impliqué premier*** *de φ si et seulement si γ est un impliqué de φ et pour tout autre impliqué γ' de φ, si $\gamma' \models \gamma$ alors $\gamma \models \gamma'$.*

Définition 17 *Un terme δ est un* ***impliquant*** *d'une formule φ si et seulement si $\delta \models \varphi$. Le terme δ est un* ***impliquant premier*** *de φ si et seulement si δ est un impliquant de φ et pour tout autre impliqué δ' de φ, si $\delta \models \delta'$ alors $\delta' \models \delta$.*

Donc, les impliqués premiers et les impliquants premiers sont minimaux par rapport à l'incluion ensembliste lorsqu'on considère les clauses et les termes comme étant des ensembles de littéraux.

Exemple 9 *Soit* $\varphi = (a \vee \neg b \vee c) \wedge (\neg c \vee d) \wedge (\neg a \vee \neg d) \wedge (b \vee c)$ *une formule propositionnelle. Alors, on a :*

- *la formule* $\neg a \vee \neg c \vee d$ *est impliqué de* φ*,*
- *la formule* $\neg a \vee \neg c$ *est un impliqué premier de* φ*,*
- *la formule* $\neg a \wedge b \wedge c \wedge d$ *est un impliquant de* φ*,*
- *la formule* $\neg a \wedge c \wedge d$ *est impliquant premier de* φ*.*

1.4 Conclusion

La logique propositionnelle est une logique simple mais qui retrouve toute sa place en termes de représentation de connaissances et de raisonnement. En plus, elle est au cœur des autres logiques qui sont plus expressives. Par ailleurs, elle suscite un intérêt particulier vis-à-vis de son caractère décidable qui n'est pas garanti par la logique des prédicats du premier ordre. Dans ce chapitre nous avons présenté brièvement les éléments de base de la logique propositionnelle. Plus précisément, nous avons esquissé ses aspects syntaxiques et sémantiques qui se trouvent utiles pour la compréhension de ce mémoire.

Chapitre 2

Complexité

2.1 Introduction

La théorie de la complexité revêt une importance capitale en Informatique. Elle s'attache à discriminer les problèmes par rapport au coût nécessaire à leur résolution en fonction de la taille de l'entrée. Ce coût peut être soit d'ordre temporel, soit d'ordre spatial. Cependant, le temps de calcul est considéré comme la ressource la plus significative, l'espace utilisé lui étant très lié.

Ce chapitre constitue une introduction succincte à la théorie de la complexité. Pour de plus amples détails, le lecteur pourra se référer utilement aux ouvrages de référence, par exemple [53] et [80]. Dans ce chapitre, nous présentons d'abord les notions de base de la complexité. La définition des différentes classes de complexité se basant sur un modèle de machine appelé machine de Turing, il convient de définir ce modèle de calcul. Ensuite, nous esquissons les classes de complexité classiques. Enfin, nous abordons la notion de la hiérarchie polynomiale.

2.2 Notions de base

En théorie de la complexité, un **problème** est donné par une entrée (ensemble de données) générique et une question sur cette entrée. Une **instance** d'un problème est une version de ce dernier où son entrée générique a été instanciée

Exemple 10 *Ci-dessous deux exemples de problèmes et quelques instances associées.*
- *Le problème "Soit x un entier naturel, x est-il premier ?" admet comme instances "2 est-il premier ?", "9 est-il premier ?", etc.*
- *Le problème "Étant données deux formules logiques ψ et ϕ, ϕ est-elle conséquence logique de ψ ?" admet comme instances "a est-elle conséquence logique de $(a \vee b) \wedge \neg b$?, "b est-elle conséquence logique de $(a \vee b)$?", etc.*

La théorie de la complexité se focalise principalement sur les problèmes de décision.

Définition 18 *Un **problème de décision** est un problème qui ne peut avoir que deux réponses possibles : "oui" ou "non".*

Un problème de décision peut être considéré comme un langage formel constitué de toutes ses instances positives.

Définition 19 *Une **instance positive** (resp. **négative**) d'un problème de décision est une instance pour laquelle la réponse est "oui" (resp. "non").*

La notion de problème **complémentaire** d'un problème de décision est donnée comme suit :

Définition 20 *Étant donné un problème de décision P, le problème complémentaire de P, noté coP, est le problème dont la réponse est "oui" quand celle de P est "non" et vice-versa. Autrement dit, les instances positives de coP coïncident avec les instances négatives de P et inversement.*

En outre, en théorie de la complexité, on n'étudie que les problèmes de décision décidables.

Définition 21 *Un problème de décision est dit **décidable** s'il existe un algorithme permettant de le résoudre en un temps fini.*

De prime abord, le fait que la théorie de la complexité s'intéresse principalement aux problèmes de décision peut sembler limitatif. Toutefois, il faut savoir que très souvent, un algorithme polynomial pour un problème de décision peut être facilement transformé en un algorithme polynomial pour un problème général (à réponses quelconques) qui lui correspond. Prenons l'exemple du voyageur de commerce.

Exemple 11 *Étant donné un ensemble de villes séparées par des distances connues, le problème du voyageur de commerce consiste à trouver le plus court chemin qui relie toutes les villes, en passant une et une seule fois par chaque ville.*

Plus formellement, ce problème peut être défini par la donnée d'un graphe complet pondéré $G = (V, A, \omega)$ où V est un ensemble de sommets (villes), A est un ensemble d'arêtes et $\omega : A \rightarrow N$ est une fonction de coût sur les arcs (distances entre les villes). La question est alors de trouver le cycle passant une et une seule fois par chaque sommet et qui minimise la somme des coûts des arcs utilisés.

A ce problème correspond le problème de décision suivant : Étant donnés un graphe complet pondéré $G = (V, A, \omega)$ et un entier naturel B, existe-t-il un cycle passant une et une seule fois par chaque sommet tel que la somme des coûts des arcs utilisés soit inférieure à B.

Clairement, si on sait résoudre efficacement l'un, on sait aussi résoudre efficacement l'autre.

Maintenant, étant donné un problème P, soit n la taille de son entrée. On dit qu'un algorithme qui résout P est en $O(f(n))$ si et seulement si à partir d'un certain rang n_0, sa complexité temporelle, c'est-à-dire le nombre d'opérations élémentaires nécessaires à son déroulement, noté par C_T, est majoré par $c * f(n)$ où c est une constante réelle positive.

Définition 22 *Un algorithme est en $O(f(n))$ où n est la taille de l'entrée du problème qu'il résout si et seulement si :*

$$\exists c, \exists n_0 : \forall n \geq n_0 : C_T \leq c * f(n).$$

Asymptotiquement, un algorithme en $O(n\ log\ n)$ est plus performant qu'un algorithme en $O(n^2)$, lequel est beaucoup plus performant qu'un algorithme en $O(2^n)$.

Exemple 12 *Considérons l'algorithme classique qui effectue la multiplication de deux matrices carrées d'ordre n, et prenons comme opérations élémentaires l'addition et la multiplication. Cet algorithme effectue n multiplications et $(n-1)$ additions n^2 fois. Par conséquent, cet algorithme est en $O(n^3)$.*

Définition 23 *Un algorithme est dit **polynomial** s'il est en $O(n^k)$ pour un entier naturel k où n est la taille de l'entrée. Un algorithme est dit **exponentiel** s'il n'est pas polynomial.*

Un problème de décision est dit efficacement solvable si et seulement s'il existe un algorithme polynomial qui permet de le résoudre. Tous les autres problèmes requièrent un temps exponentiel.

Afin de déterminer si un problème est efficacement solvable ou non, il suffit donc d'exhiber un algorithme polynomial qui le résout ou, au contraire, de prouver qu'un tel algorithme ne peut exister.

Néanmoins, pour beaucoup de problèmes, aucun algorithme polynomial n'est connu, bien qu'il semble impossible de prouver qu'un temps exponentiel soit nécessaire à leur résolution. Pour tous ces problèmes, la séparation "efficacement solvables" versus "non efficacement solvables" n'est pas assez fine pour les classifier d'une manière signifiante d'un point de vue calculatoire, d'où la nécessité d'associer des classes plus fines à de tels problèmes. Ceci est l'un des objectifs majeurs de la théorie de la complexité.

2.3 Machines de Turing

La définition des différentes classes de complexité se basant sur un modèle de calcul appelée machine de Turing, il est pertinent de rappeler les fondements de celle-ci. Une machine de Turing est un modèle abstrait du fonctionnement des appareils mécaniques de calcul, par exemple un ordinateur, qui a été introduit par Alan Turing dans son article monumental [96], en vue de donner une définition précise au concept d'algorithme. Ce modèle est utilisé en Informatique théorique, en particulier pour les problèmes de complexité et de calculabilité.

Pour le modèle des machines de Turing, il existe plusieurs définitions proches les unes des autres. L'une d'elles, relativement courante, est la suivante :

Définition 24 *Une machine de Turing est définie par la donnée des éléments suivants :*

1. *Un "ruban" divisé en cases consécutives où chaque case contient un symbole d'un alphabet fini. L'alphabet contient un symbole spécial "blanc" et éventuellement d'autres symboles. Le ruban est supposé être de longueur infinie vers la gauche ou vers la droite. Les cases non encore écrites du ruban contiennent le symbole "blanc".*
2. *Une tête de lecture/écriture pouvant lire et écrire des symboles sur la bande et se déplacer d'une case à la fois, vers la gauche ou vers la droite du ruban.*
3. *Un registre d'état qui mémorise l'état courant de la machine de Turing. Le nombre d'états possibles est toujours fini. On distingue des états spéciaux : l'états initial et le(s) état(s) d'arrêt.*
4. *Une table de transitions qui indique quel symbole écrire, comment déplacer la tête (vers la gauche ou vers la droite) et quel est le nouvel état en fonction de l'état courant de la machine et du symbole lu sur le ruban.*

Brièvement, le calcul sur une machine de Turing s'effectue de la manière suivante :
- la donnée d'entrée est placée sur le ruban, la tête pointant sur son début,
- le calcul se déroule selon la table de transitions et se termine lorsqu'une transition conduit à un état d'arrêt,
- le résultat est le mot figurant sur le ruban lorsque la machine s'arrête.

La thèse de Church-Turing postule que tout procédé de calcul de type algorithmique peut être résolu par une machine de Turing, ce qui montre l'intérêt de cet outil mathématique qui semble assez simple.

Définition 25 *Une machine de Turing est dite* ***déterministe*** *si et seulement si les transitions qui lui sont associées sont déterministes dans le sens où pour tout couple (état courant, symbole lu) de la table de transitions, est associé un triplet unique (symbole écrit, mouvement de la tête, nouvel état).*

A l'inverse, une machine de Turing non déterministe est définie comme suit :

Définition 26 *Une machine de Turing* ***non déterministe*** *peut possèder, en plus des transitions déterministes, des transitions non déterministes dans sa table de transitions, c'est-à-dire elle peut admettre un couple (état courant, symbole lu) auquel plusieurs triplets (symbole écrit, mouvement de la tête, nouvel état) correspondent.*

Il s'agit donc d'une variante purement théorique des machines de Turing : on ne peut pas construire de telles machines. En quelque sorte, elles devinent toujours juste. Une autre manière de voir leur fonctionnement est de considérer qu'à chaque choix non déterministe, elles se dédoublent, les clones poursuivent le calcul en parallèle suivant les branches du

choix. Si l'un des clones accepte l'entrée, on dit que la machine accepte l'entrée.

Les ordinateurs actuels sont conçus selon un modèle de machine à accès aléatoire. Les machines à accès aléatoire possèdent une mémoire adressable d'où un accès direct à l'information, contrairement à la machine de Turing qui doit parcourir séquentiellement son ruban pour obtenir un résultat intermédiaire. Il est donc naturel de se demander pourquoi baser les définitions de complexité sur le modèle des machines de Turing. Néanmoins, il a été démonté dans [80] que tout calcul nécessitant un temps en $f(n)$ sur une machine à accès aléatoire, peut être effectué sur une machine de Turing effectuant le même calcul en $O(f^6(n))$, voire même $O(f^3(n))$ avec une machine de Turing à plusieurs rubans.

Dans la suite de ce mémoire, nous considérons les machines de Turing à états d'arrêts oui/non : on suppose que l'ensemble des états de la machine contient deux états particuliers oui et non, à partir desquels aucune transition n'est possible, et que ce sont les deux seuls états vérifiant cette propriété. On définit alors la sortie d'une telle machine sur un mot d'entrée par l'état obtenu lorsque la machine s'arrête. Ce type de machines est adapté à la résolution de problèmes de décision qui sont au cœur de la théorie de la complexité.

2.4 Classes de complexité

On dit qu'une machine de Turing résout un problème de décision Q si pour toute instance I de Q, la machine s'arrête en renvoyant la réponse "oui" quand I est une instance positive, c'est-à-dire que le mot représentant I est accepté ou reconnu par la machine. Elle s'arrête en renvoyant la réponse "non" quand I est une instance négative ce qui signifie que le mot représentant I est rejeté par la machine.

Différentes classes de complexité regroupent des problèmes de décision selon la capacité qu'a une machine de Turing d'en permettre la résolution en consommant plus au moins de ressources (temps et espace).

Définition 27 *La classe* ***P*** *est l'ensemble des problèmes de décision qui peuvent être résolus par une machine de Turing* ***déterministe*** *en* ***temps polynomial*** *par rapport à la taille d'entrée.*

Ce sont des problèmes relativement faciles, c'est-à-dire ceux pour lesquels on connaît des algorithmes efficaces.

Définition 28 *La classe* ***NP*** *est l'ensemble des problèmes de décision qui peuvent être résolus par une machine de Turing* ***non déterministe*** *en* ***temps polynomial*** *par rapport à la taille d'entrée.*

Pour savoir si un problème donné appartient ou non à la classe NP, il suffit de pouvoir vérifier en temps polynomial si une instance donnée est positive. Par exemple, le problème SAT, de la satisfiabilité d'une formule propositionnelle sous forme CNF, est dans NP car étant donnée une interprétation, il est trivial de vérifier en temps polynomial si elle satisfait toutes les clauses en questions.

Une machine de Turing déterministe est aussi une machine de Turing non déterministe d'où P $\subseteq$ NP. En revanche, la réciproque NP $\subseteq$ P, que l'on résume généralement à P = NP du fait de la trivialité de l'autre inclusion, est l'un des problèmes ouverts les plus fondamentaux et les plus intéressants en Informatique Théorique. Cette question a été posée en 1970 pour la première fois, et celui qui arrivera à prouver que P et NP sont différents ou égaux recevra le prix de Clay (plus de 1.000.000 US$).

La question P = ? NP revient à déterminer si un problème NP-complet peut être résolu par un algorithme déterministe en temps polynomial. En effet, si cela était le cas, alors on pourrait résoudre tous les problèmes de NP en un temps polynomial, par une machine déterministe. Cependant, les problèmes NP-complets sont très fréquents et personne n'a réussi à trouver un algorithme déterministe polynomial résolvant l'un de ces problèmes. En conséquence, on conjecture que P $\neq$ NP.

Définition 29 *La classe **coNP** est l'ensemble des problèmes de décision dont les problèmes complémentaires appartiennent à la classe **NP**.*

De même P $\subseteq$ coNP et on conjecture que P $\neq$ coNP.

D'autres classes de complexité sont définies par rapport au facteur espace telles que les classes : L, NL, PSPACE et NPSPACE.

Définition 30 *La classe **L** est l'ensemble des problèmes de décision qui peuvent être résolus par une machine de Turing **déterministe** en un **espace logarithmique** par rapport à la taille d'entrée.*

Définition 31 *La classe **NL** est l'ensemble des problèmes de décision qui peuvent être résolus par une machine de Turing **non déterministe** en **espace logarithmique** par rapport à la taille d'entrée.*

Concernant les classes L et NL, on ignore aussi si L = NL. Toutefois, il s'agit d'une question moins importante que P = NP car L $\subseteq$ NL $\subseteq$ P $\subseteq$ NP.

Définition 32 *La classe **PSPACE** est l'ensemble des problèmes de décision qui peuvent être résolus par une machine de Turing **déterministe** en **espace polynomial** par rapport à la taille d'entrée.*

Définition 33 *La classe **NPSPACE** est l'ensemble des problèmes de décision qui peuvent être résolus par une machine de Turing **non déterministe** en **espace polynomial** par rapport à la taille d'entrée.*

Par contre, il a été prouvé que PSPACE = NPSPACE. De plus, NL est strictement inclus dans PSPACE. Enfin, la classe EXPTIME est définie par :

Définition 34 *La classe **EXPTIME** est l'ensemble des problèmes de décision qui peuvent être résolus par une machine de Turing **déterministe** en **temps exponentiel** par rapport à la taille d'entrée.*

Par ailleurs, une notion fondamentale en théorie de la complexité et de la calculabilité est celle de réduction.

Définition 35 *On dit qu'un problème P est **réductible** en un problème Q, noté par $P \propto Q$, si et seulement s'il existe une fonction f, calculable en temps polynomial, telle que pour toute instance I de P, on a I est instance positive de P si et seulement si $f(I)$ est instance positive de Q.*

Ceci veut dire que Q est au moins aussi difficile que P. La fonction f est appelée réduction polynomiale.

Définition 36 *Soit X une classe de complexité.*

- *un problème de décision P est dit X**-difficile** (ou X**-dur**) si tout problème de la classe X lui est réductible.*
- *un problème de décision P est dit X-complet s'il est X-difficile et en plus il appartient lui même à la classe X.*

Ainsi, un problème de décision est dit NP-complet s'il est dans NP et si tout problème dans NP lui est réductible.

Les problèmes NP-complets sont très étudiés depuis que leur existence a été mise en évidence par Cook en 1971 [30]. Ceci revient au fait que de nombreux problèmes intéressants sont NP-complets et que l'on ne sait pas résoudre un problème NP-complet d'une manière efficace. Le problème Sat, qui consiste à déterminer si une formule propositionnelle sous forme CNF est satisfiable, est le problème NP-complet de référence. Il s'agit du premier problème à avoir été démontré NP-complet [30]. Son problème complémentaire UnSat appartient à la classe coNP-complet.

2.5 Hiérarchie polynomiale

La hiérarchie polynomiale est une hiérarchie conjecturée infinie de classes de complexité situées au delà de NP. La hiérarchie polynomiale est définie via la notion de machine de Turing à oracle.

Définition 37 *Soit* ***X*** *une classe de complexité. Une machine de Turing* ***à oracle X*** *est une machine de Turing pouvant appeler un sous-programme, l'oracle, capable de résoudre n'importe quel problème de* ***X*** *en temps constant. Chaque appel compte alors pour un pas d'exécution.*

Ainsi, les classes P^X et NP^X sont définies comme suit :

Définition 38 *La classe* $\boldsymbol{P}^X$ *contient les problèmes de décision pouvant être résolus en temps polynomial par une machine de Turing déterministe à oracle X.*

Définition 39 *La classe* $\boldsymbol{NP}^X$ *contient les problèmes de décision pouvant être résolus en temps polynomial par une machine de Turing non déterministe à oracle X.*

En se basant sur ces notions, les classes Δ_k^p, Σ_k^p et Π_k^p $(k \geq 0)$ sont définies par :
- $\Delta_0^p = \Sigma_0^p = \Pi_0^p = P$
- $\Delta_{k+1}^p = P^{\Sigma_k^p}$
- $\Sigma_{k+1}^p = NP^{\Sigma_k^p}$
- $\Pi_{k+1}^p = \mathrm{co}\Sigma_{k+1}^p$.

En particulier, Δ_1^P =P, Σ_1^p =NP et Π_1^p = coNP.

Quand un problème est de complexité X, son problème complémentaire est de complexité co$X$.

Définition 40 *La hiérarchie polynomiale* ***PH*** (*pour Polynomial Hierarchy*) *est l'union des* Σ_k^P *où k est un entier naturel :*

$$PH = \bigcup_{k \geq 0} \Sigma_k^P$$

La classe $\Delta_2^p[O(log\ n)]$ est l'ensemble des problèmes de $\Delta_2^p = P^{NP}$ qui peuvent être décidés via un nombre d'appels logarithmique à un oracle NP.

La définition de PH implique les inclusions suivantes :

$$\Sigma_i^P \subseteq \Delta_{i+1}^P \subseteq \Sigma_{i+1}^P,$$
$$\Pi_i^P \subseteq \Delta_{i+1}^P \subseteq \Pi_{i+1}^P.$$

Le fait de savoir si ces inclusions sont strictes est une autre question ouverte en théorie de la complexité. Si pour un entier i, $\Sigma_i^P = \Sigma_{i+1}^P$ alors la hiérarchie polynomiale s'effondre au niveau i : pour tout entier $j \geq i$, $\Sigma_j^P = \Sigma_i^P$. En particulier, si NP = P, alors la hiérarchie polynomiale s'effondre complètement.

La figure 2.5 reprise de [65] est une représentation graphique de la hiérarchie polynomiale. Cette représentation est fidèle aux conjectures précédentes. Par exemple, la classe P est incluse dans NP et dans coNP, elles mêmes incluse dans Δ_2^P, et ainsi de suite.

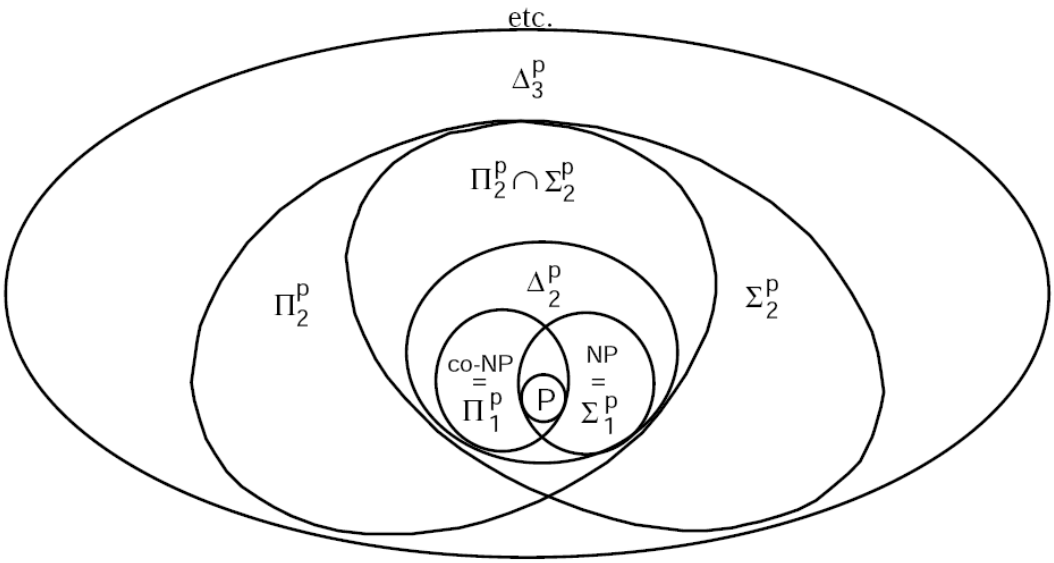

FIG. 2.1 – Hiérarchie polynomiale PH

Un problème de décision est dit au k^{ieme} niveau de la hiérarchie polynomiale si et seulement s'il appartient à Δ_{k+1}^P et est Σ_k^P-difficile ou Π_k^P-difficile.

Tous les problèmes de la hiérarchie polynomiale peuvent être résolus au pire des cas par des algorithmes déterministes en temps exponentiel. Par conséquent, la hiérarchie

polynomiale peut paraître inutile. Cependant, les niveaux de PH peuvent différer considérablement en pratique. Par exemple, les méthodes qui marchent pour les problèmes NP-complets d'une taille moyenne, ne marchent pas pour les problèmes Σ_2^P-complets.

Enfin, on sait que PH est inclus dans PSPACE mais on ne sait pas à l'heure actuelle si ces deux classes sont égales. Aussi, le problème QBF de la validité de formules booléennes quantifiées (resp. ses restrictions $k\text{QBF}_\exists$ et $k\text{QBF}_\forall$ ($k \geq 0$)) joue un rôle important en tant que problèmes complets canoniques pour PSPACE (resp. les classes Σ_k^P et Π_k^P de PH). En effet, il est connu que QBF est PSPACE-complet et que pour tout $k \geq 0$, $k\text{QBF}_\exists$ est Σ_k^P-complet alors que $k\text{QBF}_\forall$ est Π_k^P-complet.

2.6 Conclusion

La complexité est un critère capital pour évaluer le coût induit par la résolution d'un problème donné. En particulier, cette notion jouit d'une grande importance en Intelligence Artificielle, surtout en raisonnement logique.

Dans ce chapitre, nous avons rappelé des notions de base en théorie de la complexité ainsi que le modèle des machines de Turing sur lequel est basée cette théorie. En outre, nous avons passé en revue un bon nombre de classes de complexité les plus fréquemment utilisées. Enfin, nous avons abordé la notion de la hiérarchie polynomiale.

Chapitre 3

Propriétés logiques

3.1 Introduction

De nombreuses relations d'inférence ont été introduites dans le cadre du raisonnement en présence d'incohérence. Néanmoins, ces relations d'inférence, sont loin d'avoir toutes le même comportement. L'étude des propriétés logiques ou des propriétés de déduction que l'on pouvait attendre de ces différentes relations d'inférence, en vue de garantir un raisonnement "acceptable", a fait l'objet de plusieurs travaux.

Parmi les principaux travaux qui traitent de ce sujet, nous nous sommes intéressés aux travaux de référence de Kraus, Lehmann et Magidor [64, 69]. Kraus, Lehmann et Magidor proposent plusieurs systèmes de propriétés logiques tels que le système cumulatif, le système cumulatif avec boucle, le système préférentiel ainsi que le système rationnel. Le but de ce chapitre est de rappeler ces différents systèmes en considérant à la fois aussi bien les aspects syntaxiques que sémantiques.

3.2 Raisonnement cumulatif

3.2.1 Relations de conséquence cumulatives

Le système cumulatif ou le système C en abrégé est le plus faible des systèmes logiques considérés dans [64] où les auteurs pensent, à l'image de [49], qu'il englobe les propriétés de base sans lequelles un système ne peut être considéré comme un système logique.

Le système C est étroitement lié à l'inférence cumulative étudiée dans [73], et correspond exactement à ce qui a été proposé dans [49]. Le système C consiste en un axiome et un ensemble de règles d'inférence.

Définition 41 *Une relation de conséquence $\mid\sim$ est dite* ***cumulative*** *si et seulement si elle est fermée pour l'axiome de*

- ***réflexivité*** *:* $\alpha \mid\sim \alpha$

et les règles de déduction suivantes :

- ***équivalence logique gauche*** *:* $\dfrac{\alpha \equiv \beta, \alpha \mid\sim \gamma}{\beta \mid\sim \gamma}$,
- ***affaiblissement droit*** *:* $\dfrac{\alpha \models \beta, \gamma \mid\sim \alpha}{\gamma \mid\sim \beta}$,
- ***coupure*** *:* $\dfrac{\alpha \wedge \beta \mid\sim \gamma, \alpha \mid\sim \beta}{\alpha \mid\sim \gamma}$,
- ***monotonie prudente*** *:* $\dfrac{\alpha \mid\sim \beta, \alpha \mid\sim \gamma}{\alpha \wedge \beta \mid\sim \gamma}$.

Intuitivement, l'axiome et les règles précédentes reflètent ce qui suit :

- La réflexivité signifie que toute formule peut être déduite d'elle même.
- L'équivalence logique gauche stipule que des formules équivalentes doivent avoir les mêmes conséquences.
- L'affaiblissement droit indique que les conséquences logiques classiques d'une conséquence plausible sont aussi des conséquences plausibles.
- La coupure exprime le fait qu'afin de déduire une conséquence plausible à partir de certaines croyances, on peut d'abord rajouter une hypothèse à ces croyances et prouver la plausibilité de la conséquence en question à partir de ce nouvel ensemble de croyances, ensuite déduire l'hypothèse à partir de l'ensemble initial des croyances. Ce type de raisonnement est valide en logique monotone et sa validité n'implique pas la monotonie.
- Enfin, la monotonie prudente qui est une forme restrictive de la monotonie reflète le fait que la présence d'une nouvelle conséquence plausible ne peut invalider ou remettre en cause les autres conséquences plausibles.
  Kraus, Lehmann et Magidor justifient leur acceptation de la monotonie prudente d'abord par le fait qu'ils soient convaincus par l'argumentation fournie par Gabbay et qui stipule que si α est suffisante pour croire β et aussi pour croire γ, alors α et β devraient suffire aussi pour permettre de déduire γ, puisque α était suffisante de toute manière, et sur cette base, β était attendue. D'autre part, ils qualifient cette règle de très importante comme elle permet de minimiser le changement des croyances. En effet, la monotonie prudente et la coupure ensemble disent que si $\alpha \mathrel{|\!\sim} \beta$ alors les conséquences plausibles de α et celles de $\alpha \wedge \beta$ coïncident.

A partir du Système C, quatre nouvelles règles sont dérivées. La première règle, dite **règle d'équivalence**, exprime le fait que deux formules où chacune est conséquence plausible de l'autre admettent exactement les mêmes conséquences plausibles.

$$\text{Équivalence : } \frac{\alpha \mathrel{|\!\sim} \beta, \beta \mathrel{|\!\sim} \alpha, \alpha \mathrel{|\!\sim} \gamma}{\beta \mathrel{|\!\sim} \gamma}$$

La seconde règle, dite **règle du ET**, indique le fait que la conjonction de deux conséquences plausibles est une conséquence plausible.

$$\text{And : } \frac{\alpha \mathrel{|\!\sim} \beta, \alpha \mathrel{|\!\sim} \gamma}{\alpha \mathrel{|\!\sim} \beta \wedge \gamma}$$

La troisième règle, dite **règle du MPC** revient à la règle modus ponens au niveau conséquence.

$$\text{MPC : } \frac{\alpha \mathrel{|\!\sim} \beta \Rightarrow \gamma, \alpha \mathrel{|\!\sim} \beta}{\alpha \mathrel{|\!\sim} \gamma}$$

La quatrième règle, **règle numéro 9**, est peut être moins attendue et montre que le système C n'est pas aussi faible qu'on l'imagine.

$$\text{Règle numéro 9 : } \frac{\alpha \vee \beta \mathrel{|\!\sim} \alpha, \alpha \mathrel{|\!\sim} \gamma}{\alpha \vee \beta \mathrel{|\!\sim} \gamma}$$

3.2.2 Monotonie

Nous décrivons maintenant quatre nouvelles règles qui ne peuvent pas être dérivées à partir du système C. La première règle est la **règle de la monotonie**.

$$\text{Règle de la monotonie : } \frac{\alpha \models \beta, \beta \mathrel{|\!\sim} \gamma}{\alpha \mathrel{|\!\sim} \gamma}$$

La règle suivante, dite **règle EHD** (pour easy half of the deduction theorem) est donnée par :

$$\text{Règle EHD : } \frac{\alpha \mathrel{|\!\sim} \beta \Rightarrow \gamma}{\alpha \wedge \beta \mathrel{|\!\sim} \gamma}$$

Enfin, nous avons les deux règles connues suivantes à savoir la **règle de la transitivité** et la **règle de la contraposition**.

$$\text{Règle de la transitivité : } \frac{\alpha \mathrel{|\!\sim} \beta, \beta \mathrel{|\!\sim} \gamma}{\alpha \mathrel{|\!\sim} \gamma}$$

$$\text{Règle de la contraposition : } \frac{\alpha \mathrel{|\!\sim} \beta}{\neg\beta \mathrel{|\!\sim} \neg\alpha}$$

Il a été démontré dans [64] qu'en présence des règles du système C, les règles de la monotonie, EHD et de la transitivité sont équivalentes. De plus, la contraposition implique la monotonie. Par ailleurs, comme la monotonie est considérée comme contre intuitive en raisonnement non monotone, il convient de renoncer aussi aux règles EHD, de la transitivité et de la contraposition.

3.2.3 Modèles cumulatifs

La contrepartie sémantique du système C et de tous les autres systèmes logiques considérés dans ce chapitre s'appuie sur la notion de modèles. Un modèle consiste en un ensemble d'états muni d'une relation binaire qui exprime une préférence entre ces états.

Un état est désigné par un ensemble de mondes (intuitivement l'ensemble de tous les mondes qu'un agent pense possibles dans cet état). On peut considérer qu'un monde est une interprétation au sens de la logique classique. Le fait que $s \prec t$ signifie que s est préféré ou plus naturel que t.

Définition 42 *Un* ***modèle*** *est défini par la donnée d'un triplet $(S, l, \prec)$ où S est un ensemble d'états, $l : S \to 2^U$ est une fonction qui assigne à chaque état un ensemble non vide de mondes et $\prec$ est une relation binaire sur S.*

Remarquons que le terme modèle est équivoque car il a un sens bien précis en logique classique, qui est différent de celui utilisé dans la définition précédente. Cependant, il s'agit du terme proposé par Kraus, Lehmann et Magidor, et donc le fait de le garder facilite la mise en correspondance de ce chapitre avec l'article cité.

Définition 43 *Étant donnés un modèle $(S, l, \prec)$ et une formule α, on dira qu'un état $s \in S$* ***satisfait*** *la formule α, noté par $s \Vdash \alpha$, si et seulement si $\forall m \in l(S), m \models \alpha$.*

L'ensemble $\{s | S \in S, s \Vdash \alpha\}$ de tous les états qui satisfont α est noté par $\hat{\alpha}$.

Définition 44 *Un modèle $(S, l, \prec)$ est dit* ***cumulatif*** *si et seulement si la relation $\prec$ satisfait la propriété de* ***régularité*** *:*

$$\forall \alpha \in L, \text{ l'ensemble } \hat{\alpha} \text{ est } \textbf{\textit{régulier}}.$$

La notion d'ensemble régulier est donnée par la définition suivante :

Définition 45 *Soient $P \subseteq U$ et $\prec$ une relation binaire sur U. On dira que P est* ***régulier*** *si et seulement si $\forall t \in P$, soit t est minimal dans P, soit $\exists s$ minimal dans P tel que $s \prec t$.*

La régularité est nécessaire afin d'assurer la validité de la monotonie prudente. Les modèles cumulatifs acceptent une assertion conditionnelle $\alpha \mathrel{|\!\sim} \beta$ si et seulement si tous les états préférés parmi ceux satisfaisant α, satisfont β.

Définition 46 *Étant donné un modèle cumulatif $W = (S, l, \prec)$. La relation de conséquence définie par ce modèle, notée par $|\!\sim_W$, est définie par : $\alpha \mathrel{|\!\sim_W} \beta$ si et seulement si pour chaque état s minimal dans $\hat{a}$, $s \Vdash \beta$.*

Il est à noter que la relation $\prec$ n'est pas requise pour être un ordre partiel et que dans les modèles où cette relation n'est pas un ordre partiel, la condition de régularité est en

général difficile à vérifier même lorsque l'ensemble des états est fini.

Le théorème de représentation des relations cumulatives proposé par Kraus, Lehmann et Magidor postule qu'une relation de conséquence est cumulative si et seulement si elle est définie par un certain modèle cumulatif.

Ce théorème signifie que :
- pour tout modèle cumulatif, la relation de conséquence qu'il définit est cumulative,
- pour toute relation cumulative $\mid\sim$, on peut construire un modèle cumulatif W définissant une relation de conséquence $\mid\sim_W$ qui coïncide avec $\mid\sim$.

Le système C reste toutefois trop faible pour être l'épine dorsale d'un raisonnement non-monotone réaliste. En outre, les systèmes cumulatifs sont lourds à manipuler. Dans les sections suivantes, nous verrons des systèmes plus puissants.

3.3 Raisonnement cumulatif avec boucle

Définition 47 *Le **système CL** (pour cumulative loop) est construit à partir des règles du système C et de la **règle de la boucle** :*

$$\frac{\alpha_0 \mid\sim \alpha_1, \alpha_1 \mid\sim \alpha_2, \ldots, \alpha_{k-1} \mid\sim \alpha_k, \alpha_k \mid\sim \alpha_0}{\alpha_0 \mid\sim \alpha_k}$$

*Une relation de conséquence qui vérifie les règles du système CL est dite **cumulative avec boucle**.*

La règle suivante, dite **règle 15**, découle du système CL :

$$\text{Règle 15 : } \frac{\alpha_0 \mid\sim \alpha_1, \alpha_1 \mid\sim \alpha_2, \ldots, \alpha_{k-1} \mid\sim \alpha_k, \alpha_k \mid\sim \alpha_0}{\alpha_i \mid\sim \alpha_j} \ \forall i, j = 0, \ldots, k$$

3.3.1 Modèles cumulatifs ordonnés

La sémantique du système CL repose sur la notion de modèle cumulatif ordonné donné par la définition suivante :

Définition 48 *Un modèle **cumulatif ordonné** est un modèle cumulatif dans lequel la relation $\prec$ est un ordre strict partiel.*

Il est à noter que si $\prec$ est un ordre partiel (strict), alors tous les modèles finis satisfont la condition de régularité.

Le théorème de représentation des relations cumulatives avec boucle proposé par Kraus, Lehmann et Magidor postule qu'une relation de conséquence est cumulative si et seulement si elle est définie par un certain modèle cumulatif ordonné.

3.4 Raisonnement préférentiel

3.4.1 Système P

Le système décrit dans cette section occupe une position centrale par rapport aux systèmes non-monotones. Il est plus fort que le système CL. Il s'agit du système P (pour preferential). Cette appellation vient du fait que sa sémantique est une variation de celle proposée dans [90] et qui, contrairement au système P, ne fait pas de différence entre états et modèles.

Définition 49 *Le système P est construit à partir de toutes les règles du système C et de la* ***règle du OU****.*

$$\textit{Règle du OU :}\ \frac{\alpha \mathrel{|\!\sim} \gamma, \beta \mathrel{|\!\sim} \gamma}{\alpha \vee \beta \mathrel{|\!\sim} \gamma}$$

Une relation de conséquence qui satisfait toutes les règles du système P est dite relation de conséquence ***préférentielle****.*

La règle du OU dit que n'importe quelle formule qui est une conclusion plausible de deux formules différentes devrait aussi être une conclusion plausible de leur disjonction. C'est une règle valide en raisonnement monotone classique et elle n'implique pas la monotonie.

Du système P découle une règle similaire à la moitié difficile du théorème de déduction à savoir la **règle S**. Il est à noter que cette règle est appelée règle de la conditionnalisation dans [52].

$$\text{Règle S :}\ \frac{\alpha \wedge \beta \mathrel{|\!\sim} \gamma}{\alpha \wedge \beta \Rightarrow \gamma}$$

Une deuxième règle est dérivée du Système P. Il s'agit de la **règle D** qui exprime le principe de preuve par cas.

$$\text{Règle D} : \frac{\alpha \wedge \beta \mid\sim \gamma, \alpha \wedge \neg\beta \mid\sim \gamma}{\alpha \mid\sim \gamma}$$

D'autres règles sont dérivées à partir du système P telles que :

$$\text{La règle 19} : \frac{\alpha \mid\sim \gamma, \beta \mid\sim \delta}{\alpha \vee \beta \mid\sim \gamma \vee \delta}$$

$$\text{La règle 20} : \frac{\alpha \vee \gamma \mid\sim \gamma, \alpha \mid\sim \beta}{\gamma \mid\sim \alpha \Rightarrow \beta}$$

$$\text{Le règle 21} : \frac{\alpha \vee \beta \mid\sim \alpha, \beta \vee \gamma \mid\sim \beta}{\alpha \vee \gamma \mid\sim \alpha}$$

$$\text{Le règle 21} : \frac{\alpha \vee \beta \mid\sim \alpha, \beta \vee \gamma \mid\sim \beta}{\alpha \mid\sim \gamma \Rightarrow \beta}$$

Remarquons que le système P est puissant et qu'il permet de construire des preuves sophistiquées.

3.4.2 Modèles préférentiels

Les modèles préférentiels sur lesquels est basé le système préférentiel sont des modèles cumulatifs ordonnés dans lesquels à chaque état correspond un seul monde (et non pas un ensemble de mondes).

Définition 50 *Un modèle **préférentiel** est un triple $W = (S, l, \prec)$ où S est un ensemble d'états, $l : S \to U$ est une fonction qui assigne à chaque état un seul monde et $\prec$ est un ordre strict partiel sur S, satisfaisant la propriété de **régularité**.*

Le théorème de représentation des relations préférentielles dit qu'une relation de conséquence est préférentielle si et seulement si elle est définie par un modèle préférentiel.

3.5 Raisonnement rationnel

3.5.1 Système rationnel

Kraus, Lehmann et Magidor pensent qu'un bon système de raisonnement non-monotnone devrait valider toutes les règles du système P. De plus, ils disent qu'il n'est pas possible de

rajouter d'autres règles de cette forme, c'est-à-dire de la forme : à partir de la présence de certaines assertions, on déduit la présence d'autres assertions.

Néanmoins, certaines règles qui semblent intéressantes échouent pourtant à être validées par le système P. Autrement dit, un raisonnement qui est totalement conforme au système P, peut être néanmoins irrationnel.

Ces règles sont de la forme : en l'absence de certaines assertions, on déduit l'absence d'autres asserations, et elles sont toutes impliquées par la règle de la monotonie. Il s'agit de la **règle de la rationnalité de la négation**, de la **règle de la rationnalité disjonctive** et de la **règle de la monotonie rationnelle** définies comme suit :

$$\text{Règle de la rationnalité de la négation : } \frac{\alpha \wedge \gamma \not\sim \beta, \alpha \wedge \neg\gamma \not\sim \beta}{\alpha \not\sim \beta}$$

$$\text{Règle de la rationnalité disjonctive : } \frac{\alpha \not\sim \gamma, \beta \not\sim \gamma}{\alpha \vee \beta \not\sim \gamma}$$

$$\text{Règle de la monotonie rationnelle : } \frac{\alpha \wedge \beta \not\sim \gamma, \alpha \not\sim \neg\beta}{\alpha \not\sim \gamma}$$

La règle de la rationalité de la négation part du principe que si l'on accepte que β est une conséquence plausible de α, alors on doit soit accepter que c'est une conséquence plausible de $\alpha \wedge \gamma$, soit accepter que c'est une conséquence plausible de $\alpha \wedge \neg\gamma$. Cette règle est impliquée par la rationalité disjonctive qui est à son tour impliquée, en présence des règles du système C, par la monotonie rationnelle. Cette dernière règle constitue un outil important afin de minimiser la mise à jour quand on apprend une nouvelle information. Cette règle était initialement destinée à résoudre le problème de non-pertinence qui apparaissait dans le système P.

Lehmann et Magidor ont poursuivi leur investigation sur la règle de la monotonie rationnelle en définissant la notion du système R et de la relation de conséquence rationnelle [69].

Définition 51 *Le **système R** est construit à partir des règles du système P et de la monotonie rationnelle.*

La règle de la monotonie rationnelle a été présentée aussi dans [51, 52] sous une autre forme mais qui reste équivalente à la forme précédente.

$$\text{Monotonie rationnelle (forme 2) : } \frac{\alpha \mathrel{|\!\sim} \gamma, \alpha \not\sim \neg\beta}{\alpha \wedge \beta \mathrel{|\!\sim} \gamma}$$

Remarquons que cette nouvelle forme de la monotonie rationnelle est plus populaire que celle proposée par Kraus, Lehmann et Magidor.

3.5.2 Modèles rangés

La sémantique du système R s'appuie sur la notion de modèles rangés qui sont des modèles préférentiels définis comme suit :

Définition 52 *Un modèle préférentiel $(S, l, \prec)$ est dit **modèle rangé** si et seulement si la relation $\prec$ est définie de la manière suivante : il existe un ensemble ordonné Ω (représenté par la relation $\prec_\Omega$) et une fonction R de S dans Ω telle que : $\forall s, t \in S$, $s \prec t$ si et seulement si $R(s) \prec_\Omega R(t)$.*

On a alors une relation d'équivalence entre les états d'un même rang.

La théorème de représentation dans ce cas stipule qu'une relation de conséquence est rationnelle si et seulement si elle est définie par un modèle rangé. Or, quand on calcule l'ensemble des formules issu d'une relation d'inférence rationnelle pour un ensemble des formules initial donné, on constate que cet ensemble n'est pas unique. Lehmann et Magidor ont résolu ce problème en proposant un mode de sélection d'un ensemble particulier, qu'ils appellent la fermeture rationnelle. Cet ensemble est construit via une fonction qui associe à chaque formule du langage un nombre entier positif qui reflète le degré d'exceptionalité de la formule. La base de croyances est ainsi stratifiée. La sémantique de cette relation rationnelle particulière repose sur un modèle rangé particulier.

Ainsi, les différents systèmes de raisonnement décrits dans ce chapitre peuvent être récapitulés via le tableau 3.1.

Règles de déduction	Système	Modèle
Les règles de base : réflexivité, équivalence logique gauche, affaiblissement droit, coupure et monotonie prudente.	C	cumulatif
Les propriétés de base et la boucle	CL	cumulatif ordonné
Les propriétés de base et le OU	P	préférentiel
Les propriétés de base, le OU et la monotonie rationnelle	R	rangé

TAB. 3.1 – Systèmes C, CL, P, R

3.6 Conclusion

Dans ce chapitre nous avons rappelé les systèmes de propriétés logiques proposés par Kraus, Lehmann et Magidor dans [64, 69] à l'image du système C, le système C avec boucle, le système P et le système R.

En particulier, le système P qui englobe le système C et le système C avec boucle est communément considéré comme étant le noyau minimal de tout système non-monotone "raisonnable". En revanche, le système R est habituellement qualifiée de désirable. Néanmoins, nul argument définitif ne soit venu démontrer la nécessite de la règle de la monotonie rationnelle qui n'est pas dotée d'une approbation unanime. En effet, la relation d'inférence rationnelle s'avère insuffisante pour résoudre le problème de non pertinence qui est le problème motivant sa définition. Par ailleurs, d'autres relations d'inférence le contournent sans pour autant satisfaire la monotonie rationnelle.

Enfin, notons que ces travaux qui sont définis dans un cadre propositionnel, ont été étendus par Lehmann et Magidor dans [68] dans le cadre de la logique des prédicats du premier ordre.

Chapitre 4

Approches basées sur la restauration de la cohérence

4.1 Introduction

Le raisonnement en présence d'incohérence est un problème très pertinent que l'on rencontre dans diverses situations telles que le raisonnement tolérant les exceptions, la révision de croyances, la fusion d'informations provenant de sources multiples (éventuellement conflictuelles), le raisonnement incertain, etc. Dans ce cas, l'inférence classique ne peut être directement appliquée puisqu'à partir d'une base incohérente, on peut déduire n'importe quoi (principe d'ex falso quodlibet sequitur).

Différentes approches ont été alors proposées pour raisonner en présence d'incohérence. Tandis que certaines consistent à affaiblir la relation d'inférence classique, à l'instar des logiques paraconsistantes [32], d'autres affaiblissent les croyances disponibles telles que les approches basées sur la restauration de la cohérence.

Selon [82], l'inférence en présence d'incohérence via une approche basée sur la restauration de la cohérence peut être vue comme un processus à deux phases qui consiste à :

1. générer, dans un premier temps, des sous-bases cohérentes préférées,
2. ensuite, appliquer l'inférence classique à partir de certaines de ces sous-bases selon le principe d'inférence choisi.

Les approches basées sur la restauration de la cohérence sont très populaires. Leur succès peut être expliqué par plusieurs facteurs. D'abord, elles sont très simples de par leur nature. De plus, elles englobent d'autres cadres importants à l'image de la révision de croyances basée sur la syntaxe [77]. Par ailleurs, les approches basées sur la restauration de la cohérence présentent l'avantage de permettre l'utilisation de toutes les connaissances dont on dispose en logique classique, et donc on se ramène à des outils bien connus. En outre, elles permettent de profiter de toutes les améliorations concernant la logique classique.

De nombreuses approches basées sur la restauration de la cohérence ont été proposées, parmi lesquelles on peut distinguer celles destinées à des bases de croyances totalement préordonnées, et celles qui sont dédiées à des bases de croyances partiellement préordonnées. Dans ce chapitre, nous allons passer en revue un bon nombre d'approches basées sur la restauration de la cohérence par rapport à des bases de croyances totalement préordonnées, à savoir la logique possibiliste, l'inférence linéaire, l'inférence lexicographique ainsi que l'inférence relative à la préférence pour l'inclusion. Nous rappelons ensuite les extensions de certaines de ces approches dans le cadre de bases de croyances partiellement préordonnées, notamment les inférences possibilistes forte et faible, l'inférence démocratique et l'inférence relative à la préférence pour l'inclusion basée-compatibles.

4.2 Préliminaires formels

Avant de présenter les relations d'inférence à partir de bases de croyances totalement préordonnées et les relations d'inférence à partir de bases de croyances partiellement préordonnées, il convient tout d'abord de définir ce que sont de telles bases, et de rappeler aussi quelques notions élémentaires. Nous commençons par définir la notion de bases de croyances, sous-bases cohérentes et sous-bases maximales cohérentes. Ensuite, nous donnons la définition formelle des bases de croyances totalement préordonnées et partiellement préordonnées après un bref rappel sur les préordres.

4.2.1 Bases de croyances et sous-bases cohérentes

Définition 53 *Une* ***base de croyances*** Σ *est un ensemble fini de croyances*[1] *données sous la forme de formules logiques classiques* $\{\varphi_1, \ldots, \varphi_n\}$.

Les croyances peuvent être des formules propositionnelles, des formules d'une logique de description, ou plus généralement des formules de la logique du premier ordre.

Comme cela a été discuté dans [77], il existe deux manières différentes de voir une base de croyances. En effet, la première vision dit qu'une base de croyances peut être représentée d'une manière équivalente par l'ensemble de ses modèles. Quant à la seconde, elle stipule qu'une base de croyances peut être considérée syntaxiquement dans le sens où chaque formule constitue une information distincte. Par exemple, dans ce cas la base de croyances $\{\varphi_1, \varphi_2\}$ est différente de la bases de croyances $\{\varphi_1 \wedge \varphi_2\}$ puisque la première peut être obtenue via deux experts, le premier asserte φ_1 et le second asserte φ_2, alors que la deuxième doit être considérée comme un tout indivisible. En particulier, la formule φ_1 peut être gardée ou rejetée indépendamment de la formule φ_2 et inversement, alors que ce n'est pas le cas par rapport à la seconde base. Dans le reste de ce mémoire, nous adoptons l'approche syntaxique afin de traiter l'incohérence de croyances pouvant provenir de plusieurs sources indépendantes.

Formellement, la notion de sous-base est donnée comme suit :

Définition 54 *Une* ***sous-base*** A *de* Σ *est un sous-ensemble de* Σ. *A est dite une* ***sous-base cohérente*** *si et seulement si elle est cohérente au sens de la logique classique.*

L'ensemble de toutes les sous-bases cohérentes de Σ sera noté par $Cons(\Sigma)$.

Une première idée pour palier l'incohérence d'une base consiste à considérer ses sous-bases cohérentes maximales pour l'inclusion.

[1] Les croyances désignent des connaissance incertaines.

Définition 55 *Une sous-base A de Σ est une* ***sous-base maximale cohérente*** *(ou* ***sous-base cohérente maximale pour l'inclusion****) si et seulement si :*

- *A est une sous-base cohérente de Σ : $A \in Cons(\Sigma)$,*
- *il n'existe pas de sous-base cohérente de Σ contenant strictement A : $\forall B \in Cons(\Sigma), A \not\subset B$.*

L'ensemble de toutes les sous-bases maximales cohérentes de Σ sera noté par $MaxCons(\Sigma)$. L'idée derrière cette définition est claire : on souhaite modifier les croyances disponibles aussi peu que possible. Il est à noter que la notion de sous-bases maximales cohérentes est déjà ancienne (voir [86]).

Une autre idée naturelle consiste à considérer des sous-bases cohérentes maximales pour la cardinalité.

Définition 56 *Une sous-base A de Σ est une* ***sous-base cohérente maximale pour la cardinalité*** *de Σ si et seulement si :*

- *A est une sous-base cohérente de Σ : $A \in Cons(\Sigma)$,*
- *il n'existe pas de sous-base cohérente B de Σ telle que $|B| > |A|$: $\forall B \in Cons(\Sigma), |B| \leq |A|$.*

Les sous-bases cohérentes maximales pour la cardinalité sont des sous-bases maximales cohérentes.

Par ailleurs, l'introduction d'une relation de préférence entre les croyances réduit souvent le nombre de solutions générées et mène à des résultats plus plausibles. Par exemple, les croyances peuvent être préordonnées par rapport à leur degré de certitude. Dans ce cas, on préfère rejeter les formules qui sont moins certaines. Elles peuvent également être préordonnées par rapport à leur temps d'arrivée, et dans ce cas, on préfère ignorer les formules les plus anciennes afin de restaurer la cohérence. Cette relation de préférence entre les formules peut être définie par un préordre total quand toutes les informations sont comparables, comme elle peut être décrite par un préordre partiel qui permet de traiter des informations incomparables ou bien des informations dont on ignore le degré de priorité. Par conséquent, parmi les approches basées sur la restauration de la cohérence, on peut d'une manière générale, distinguer les relations d'inférence à partir de bases de croyances totalement préordonnées et les relations d'inférence à partir de bases de croyances partiellement préordonnées.

4.2.2 Bref rappel sur les préordres

Soit A un ensemble fini.

Définition 57 *Un* ***préordre partiel*** $\preceq$ *sur* A *est une relation binaire réflexive* ($\forall a \in A, a \preceq a$) *et transitive* ($\forall a, b, c \in A$, *si* $a \preceq b$ *et* $b \preceq c$ *alors* $a \preceq c$).

Soient $a, b \in A$. Dans toute la suite de ce manuscrit, $a \preceq b$ signifie intuitivement que a est au moins aussi préféré que b où le terme "est préféré" est à prendre dans son sens le plus large. Il peut être substitué par "plus plausible", "plus fiable", "plus sûr", etc.

Définition 58 *Un* ***ordre partiel strict*** $\prec$ *sur* A *est une relation binaire irréflexive* ($\forall a \in A$, *on n'a pas* $a \prec a$) *et transitive.*

Donc, $a \prec b$ signifie dans ce qui suit que a est strictement préféré à b.

Un ordre partiel strict est généralement défini à partir d'un préordre partiel par : $a \prec b$ si et seulement si $a \preceq b$ est vérifié alors que $b \preceq a$ ne l'est pas.

Notons que le terme ordre partiel strict est utilisé au lieu de préordre partiel strict car un ordre partiel strict est forcément antisymétrique.

Par ailleurs, l'égalité et l'incomparabilité sont définies comme suit :

Définition 59 *Soient* $a, b \in A$.
- *L'****égalité****, notée par* $\approx$, *est définie par* $a \approx b$ *si et seulement si* $a \preceq b$ *et* $b \preceq a$.
- *L'****incomparabilité****, notée* $\sim$, *est définie par* $a \sim b$ *si et seulement on n'a ni* $a \preceq b$ *ni* $b \preceq a$.

L'écriture $a \npreceq b$ (resp. $a \nprec b$, $a \not\approx b$) signifie que l'on n'a pas $a \preceq b$ (resp. $a \prec b$, $a \approx b$).

Définition 60 *L'ensemble des* ***éléments minimaux*** *de* A *par rapport à un ordre partiel strict* $\prec$, *noté* $Min(A, \prec)$, *est défini par :*

$$Min(A, \prec) = \{a \in A, \nexists b \in A : b \prec a\}.$$

Par conséquent, tout au long de ce manuscrit, les éléments minimaux d'un ensemble par rapport à un ordre strict désigne les éléments préférés de cet ensemble par rapport à cet ordre.

Définition 61 *Un* ***préordre total*** $\leq$ *sur* A *est une relation binaire réflexive et transitive telle que tous les éléments de* A *sont deux à deux comparables via ce préordre :* $\forall a, b \in A$, $a \leq b$ *ou* $b \leq a$.

Enfin, la notion de compatibilité entre un préordre total et un préordre partiel est définie comme suit :

Définition 62 *un préordre total $\leq$ est dit* ***compatible*** *avec un préordre partiel $\preceq$ si le préordre total étend le préordre partiel. Plus formellement :*

- $\forall a, b \in A$: *si* $a \prec b$ *alors* $a < b$,
- $\forall a, b \in A$: *si* $a \preceq b$ *alors* $a \leq b$.

4.2.3 Bases de croyances totalement préordonnées et bases de croyances partiellement préordonnées

Une base de croyances totalement préordonnées (ou stratifiées) est une base de croyances munie d'un préordre total.

Définition 63 *Une* ***base de croyances totalement préordonnées*** *est définie par la donnée d'un couple $(\Sigma, \leq)$ où $\Sigma = \{\varphi_1, \ldots, \varphi_n\}$ est une base de croyances et $\leq$ est un préordre total sur Σ.*

Le préordre total $\leq$ reflète la relation de priorité qui existe entre ces croyances.

De manière équivalente, une base de croyances totalement préordonnées $(\Sigma, \leq)$ peut être définie sous forme stratifiée (d'où l'appelation **bases de croyances stratifiées**) comme étant une suite finie $(S_1, \ldots, S_m)$ de sous-bases de Σ, où chaque S_i $(1 \leq i \leq m)$ vérifie les propriétés suivantes :

- les formules de S_i ont le même niveau de priorité qui est i : $\forall \varphi, \psi \in S_i$, on a $\varphi = \psi$,
- les formules de S_i sont strictement plus prioritaires que celles de S_j $(1 \leq j \leq m)$ pour $j > i$: $\forall \varphi \in S_i, \forall \psi \in S_j$, on a $\varphi < \psi$.

Il est claire que $\{S_1, \ldots, S_m\}$ est une partition de Σ. Chaque sous-ensemble S_i $(1 \leq i \leq m)$ est appelée strate de $(\Sigma, \leq)$, et i représente le niveau de priorité des formules de S_i. Intuitivement, plus le niveau de priorité d'une formule est bas, plus importante est sa plausibilité.

Enfin, les bases de croyances partiellement préordonnées sont définies de manière similaire par rapport aux bases de croyances totalement préordonnées en remplaçant le préordre total par un préordre partiel en vue d'avoir plus de flexibilité.

Définition 64 *Une* ***base de croyances partiellement préordonnées*** $(\Sigma, \preceq)$ *est une base de croyances $\Sigma = \{\varphi_1, \ldots, \varphi_n\}$ munie d'un préordre partiel $\preceq$.*

4.3 Inférence à partir de bases de croyances totalement préordonnées

4.3.1 Logique possibiliste

La logique possibiliste [46, 45] est une extension de la logique classique issue de la théorie des possibilités [103], qui à son tour est issue de la théorie des ensembles flous [102]. Au niveau syntaxique, une formule possibiliste n'est rien d'autre qu'une formule classique qui se voit assigner un poids reflétant son degré de certitude par rapport aux autres formules. Sémantiquement, au lieu de partitionner l'ensemble des interprétations en deux parties (modèles et contre-modèles), on a une partition plus raffinée, dite distribution de possibilités, où les contre-modèles des croyances moins plausibles sont préférés aux contre-modèles des croyances plus plausibles. Par ailleurs, la logique possibiliste possède une contrepartie qualitative définie via un préordre total aussi bien sur le plan syntaxique que sémantique, et donc elle s'applique aux bases de croyances totalement préordonnées.

Sémantique

L'élément de base de la sémantique de la logique possibiliste [103, 46] est la distribution de possibilités, qui est une application de l'ensemble des interprétations Ω vers l'intervalle [0,1] qui peut être remplacé par n'importe quel échelle totalement préordonnée, finie ou infinie.

Une distribution de possibilités π représente les connaissances disponibles quant au monde réel. Ainsi, $\pi(\omega) = 1$ signifie que ω est totalement possible, $\pi(\omega) > 0$ signifie que ω est possible alors que $\pi(\omega) = 0$ signifie que ω est totalement impossible. L'inégalité $\pi(\omega) > \pi(\omega')$ signifie que ω est plus plausible que ω'.

La distribution de possibilité π induit deux applications représentant respectivement le degré de possibilité et le degré de certitude d'une formule α :

- Le **degré de possibilité** (ou de **cohérence**) $\Pi(\alpha) = max\{\pi(\omega) : \omega \models \alpha\}$ évalue à quel degré α est cohérente avec les connaissances disponibles exprimées par π.
- Le **degré de nécessité** (ou de **certitude**) $N(\alpha) = 1 - \Pi(\neg\alpha)$ évalue à quel degré α est déductible à partir des connaissances disponibles exprimées par π.

Si on se réfère à la logique classique, $\Pi(\alpha) = 1$ exprime le fait que α est cohérente avec la base (au sens de la logique classique) et $N(\alpha) = 1$ signifie que α est déductible à partir de la base.

La dualité $N(\alpha) = 1 - \Pi(\neg\alpha)$ étend celle de la logique classique où une formule α est déductible à partir d'une base ($N(\alpha) = 1$) si et seulement si sa négation $\neg\alpha$ est incohérente avec cette même base ($\Pi(\neg\alpha) = 0$).

Un autre point important est le principe de minimum de spécificité [98].

Définition 65 *Une distribution de possibilité π est dite moins spécifique qu'une autre distribution de possibilité π' si et seulement si pour chaque interprétation ω on a $\pi'(\omega) \leq \pi(\omega)$ et il existe au moins une interprétation ω' telle que $\pi'(\omega') < \pi(\omega')$.*

Notons que d'un point de vue purement qualitatif, un préordre total sur les interprétations, noté par $\leq_\pi$, peut être associé à une distribution de possibilité π de la manière suivante :

$$\forall \omega, \omega' \in \Omega, \omega \leq_\pi \omega' \; ssi \; \pi(\omega') \leq \pi(\omega).$$

Syntaxe

La logique possibiliste dans sa version la plus simple appelée logique possibiliste standard ou logique possibiliste évaluée-nécessité manipule uniquement des formules évaluées certitude. En fait, il est plus facile d'élaborer des algorithmes dans cette version ensuite les généraliser. En outre, d'un point de vue représentation de connaissances, cette version s'avère suffisante pour de nombreuses applications. Par la suite, on se retreint à la logique possibiliste standard.

Syntaxiquement, une formule logique possibiliste est une paire formée d'une formule logique classique ψ et d'un poids $a \in]0, 1]$ exprimant son degré de certitude. Le poids a est interprété comme la borne inférieure de $N(\psi)$, c'est-à-dire l'expression possibiliste (ψ, a) exprime $N(\psi) \geq a$ [45].

Comme $N(\varphi \wedge \psi) = min(N(\varphi), N(\psi))$, il est toujours possible de mettre une formule possibiliste (φ, a) sous forme d'une conjonction de clauses (φ_i, a) avec $1 \leq i \leq n$ si $\varphi \equiv \bigwedge_{i=1}^{n} \varphi_i$.

Définition 66 *Étant donnée une base possibiliste $\mathcal{B} = \{(\psi_i, a_i) : 1 \leq i \leq n\}$, $\mathcal{B}_a$ dénote la base propositionnelle classique $= \{\psi_i : (\psi_i, a_i) \in \mathcal{B}$ et $a_i \geq a\}$. Le **degré d'incohérence** de $\mathcal{B}$, noté $Inc(\mathcal{B})$, est donné par :*

$$Inc(\mathcal{B}) = max\{a : \mathcal{B}_a \models \bot\}.$$

Notons que par convention, $max\emptyset = 0$ et donc $Inc(\mathcal{B}) = 0$ si et seulement si $\{\varphi_i : (\varphi_i, a_i) \in \mathcal{B}\}$ est cohérente au sens classique.

Exemple 13 *Soit $\mathcal{B} = \{(\neg q \vee r, 0.9), (q, 0.7), (s, 0.7), (\neg r, 0.5), (\neg s \vee r, 0.3)\}$ une base possibiliste. Alors, on a $Inc(\mathcal{B}) = 0.5$ car $\mathcal{B}_{0.7}$ est cohérente tandis que la base $\mathcal{B}_{0.5}$ ne l'est pas.*

A partir d'une base de croyances possibiliste, trois relations d'inférence peuvent être considérées, à savoir l'inférence possibiliste simple, l'inférence possibiliste pondérée et le conditionnement possibiliste. L'inférence possibiliste simple et l'inférence possibiliste pondérée sont définies comme suit :

Définition 67 *Soit $\mathcal{B}$ une base possibiliste et soit φ une formule logique classique. Alors,*
- *φ est une conséquence possibiliste (simple) de $\mathcal{B}$, noté $\mathcal{B} \vdash_\pi \varphi$, ssi $Inc(\mathcal{B} \cup \{(\neg\varphi, 1)\}) > Inc(\mathcal{B})$.*
- *φ est une conséquence possibiliste (pondérée) de $\mathcal{B}$ à un degré a_i, noté $\mathcal{B} \vdash_\pi (\varphi, a_i)$, ssi $Inc(\mathcal{B} \cup \{(\neg\varphi, 1)\}) > Inc(\mathcal{B})$ et $a_i = Inc(\mathcal{B} \cup \{(\neg\varphi, 1)\})$.*

Exemple 14 *Considérons encore la base possibiliste $\mathcal{B}$ donnée dans l'exemple 13. Rappelons que $Inc(\mathcal{B}) = 0.5$. Maintenant, vérifions si r est une conséquence possibilste de $\mathcal{B}$, et à quel degré si c'est vrai.*

En fait, on a $Inc(\mathcal{B} \cup \{(\neg r, 1)\}) = 0.7 > Inc(\mathcal{B})$. Par conséquent, la formule r est une conséquence possibiliste de $\mathcal{B}$: $\mathcal{B} \vdash_\pi r$.

Par ailleurs, il n'existe aucun poids $a > 0.7$ tel que $\mathcal{B}_a \models r$. Donc, le degré d'inférence possibiliste de r à partir de $\mathcal{B}$ vaut 0.7 : $\mathcal{B} \vdash_\pi (r, 0.7)$.

Notons que l'inférence possibiliste simple revient à appliquer l'inférence classique à partir de la base classique qui correspond aux formules ayant un degré de certitude plus grand que le degré d'incohrénce de la base en question.

Quant au conditionnement possibiliste, il est donné par la définition suivante :

Définition 68 *Soit $\mathcal{B}$ une base possibiliste et soient φ et ψ deux formules logiques classiques. Alors φ est une conséquence possibiliste du conditionnement de $\mathcal{B}$ par α (respectivement à un degré a_i) ssi $\mathcal{B} \cup \{(\alpha, 1)\} \models_\pi \varphi$(respectivement (φ, a_i)).*

Le conditionnement possibiliste est un concept important en logique possibiliste. Il représente, en quelque sorte, la contrepartie logique des algorithmes de propagation dans les réseaux possibilistes.

Étant donnée une base possibiliste $\mathcal{B}$, la distribution de possibilité associée, notée par $\pi_\mathcal{B}$, est telle que $\forall\omega \in \Omega$:

$$\pi_\mathcal{B}(\omega) = \begin{cases} 1 \text{ si pour toute } (\varphi_i, a_i) \in \mathcal{B} : \omega \models \varphi_i, \\ 1 - max\{a_i : (\varphi_i, a_i) \in \mathcal{B} \text{ et } \omega \not\models \varphi_i\} \text{ sinon.} \end{cases}$$

Clairement, $\pi_\mathcal{B}(\omega)$ représente le degré de cohérence de ω par rapport aux connaissances disponibles codées par $\mathcal{B}$. En particulier, si ω est un modèle de toutes les informations de $\mathcal{B}$, alors $\pi_\mathcal{B}(\omega) = 1$ signifie que ω complètement cohérente avec $\mathcal{B}$. Autrement, le degré de

cohérence associé à ω correspond au poids de la plus importante formule de $\mathcal{B}$ falsifiée par ω. Par exemple, si ω falsifie une formule certaine, le poids de ω sera nul, c'est-à-dire que ω est complètement incohérente.

Par ailleurs, une base de croyances possibilistes $\mathcal{B}$ peut être représentée par un préordre total sur l'ensemble de ses formules :

$$\varphi \leq \varphi' \text{ ssi } (\varphi, a), (\varphi', b) \in \mathcal{B} \text{ et } b \leq a.$$

Autrement dit, une base de croyances possibilistes peut être donnée sous forme d'une base de croyances stratifiées $(\Sigma, \leq) = (S_1, \ldots, S_m)$ avec $m \geq 1$. Dans ce cas, la relation d'inférence possibiliste simple revient à appliquer l'inférence classique à partir de la sous-base cohérente issue de l'union des strates qui se trouvent avant le degré d'incohérence, qui est dans ce cas le degré de la première strate où on rencontre une incohérence en commençant par les informations les plus prioritaires. Plus formellement :

Définition 69 *Soit $(\Sigma, \leq) = (S_1, \ldots, S_m)$ une base de croyances stratifiées, et soit φ une formule propositionnelle. Considérons la sous-base cohérente notée par $Pos(\Sigma, \leq)$ telle que* $Pos(\Sigma, \leq) = \bigcup_{i=1}^{Inc(\Sigma,\leq)-1} S_i$.

Alors, la formule φ est une conséquence possibiliste (simple) de $(\Sigma, \leq)$, noté $(\Sigma, \leq) \vdash_\pi \varphi$, si et seulement si $Pos(\Sigma, \leq) \models \varphi$.

Quant à l'inférence possibiliste pondérée, elle se définit dans ce cadre purement qualitatif par la définition suivante :

Définition 70 *Soit $(\Sigma, \leq) = (S_1, \ldots, S_m)$ une base de croyances stratifiées et soit φ une formule. Alors, φ est une conséquence possibiliste de $(\Sigma, \leq)$ à un degré i, noté $(\Sigma, \leq) \vdash_\pi (\varphi, i)$, si et seulement si :*

- *$S_1 \cup \ldots \cup S_i$ est cohérente,*
- *$S_1 \cup \ldots \cup S_i \models \varphi$,*
- *$\nexists j, j < i$ tel que $S_1 \cup \ldots \cup S_j \models \varphi$.*

Pour le conditionnement possibiliste, il suffit de mettre la formule par laquelle on conditionne dans une nouvelle strate S_0 qui sera considérée comme étant plus prioritaire que toutes les autres strates.

Exemple 15 *Considérons la base possibiliste donnée par l'exemple 13. Cette base peut être donnée sous forme stratifiée $(\Sigma, \leq) = (S_2, S_2, S_3, S_4)$ telle que :*

- *$S_1 = \{\neg q \vee r\}$,*
- *$S_2 = \{q, s\}$,*

– $S_3 = \{\neg r\}$,
– $S_4 = \{\neg s \vee r\}$.

Dans ce cas, on a S_1 *est cohérente,* $S_1 \cup S_2$ *est cohérente alors que* $S_1 \cup S_2 \cup S_3$ *ne l'est pas. De ce fait,* $Inc(\Sigma, \leq) = 3$*, et par conséquent,* $Pos(\Sigma, \leq) = S_1 \cup S_2 = \{\neg q \vee r, q, s\}$*. Par exemple, on a* $Pos(\Sigma, \leq) \models r$ *ce qui veut dire que* $(\Sigma, \leq) \vdash_\pi r$*. Aussi, puisque* $S_1 \not\models r$ *et* $S_1 \cup S_2 \models r$*, on déduit que le degré d'inférence de* r *vaut* 2*.*

Enfin, toutes les relations d'inférence que nous allons considérer dans la suite de ce mémoire seront rappelées sous forme qualitative. Par conséquent, nous nous focaliserons dans ce qui suit uniquement sur la contrepartie qualitative de la logique possibiliste en ce considérant des bases de croyances stratifiées, et ce sans perdre de généralité.

4.3.2 Inférence linéaire

Dans le cas de l'inférence possibiliste, le principe revient à appliquer l'inférence classique à partir de la sous-base formée par l'union de toutes les strates qui se situent avant le degré d'incohérence, et donc toutes le strates à partir du degré d'incohérence sont ignorées. L'inférence linéaire introduite par Nebel dans [79] est proche de l'inférence possibiliste sauf qu'elle ne s'arrête pas dès qu'elle rencontre une incohérence. En revanche, la strate responsable de l'incohérence est ignorée, et on passe à la strate suivante. Si cette dernière est cohérente avec les strates retenues précédemment, elle est retenue elle aussi sinon on l'ingnore, et dans les deux cas on passe à la strate suivante pour réitérer le même traitement. Plus formellement, la sous-base cohérente sélectionnée dans ce cas est donnée par la définition suivante :

Définition 71 *Soit* $(\Sigma, \leq) = (S_1, \ldots, S_m)$ *une base de croyances stratifiées. La seule sous-base cohérente considérée par l'inférence linéaire, notée par* $Lin(\Sigma, \leq)$*, est donnée par* $Lin(\Sigma, \leq) = \bigcup_{i=1}^{m} S'_i$ *tel que :*

$$S'_i = \begin{cases} S_i \text{ si } S_i \cup \bigcup_{j=1}^{i-1} S'_j \text{ est cohérent,} \\ \emptyset \text{ sinon.} \end{cases}$$

L'inférence linéaire revient ensuite à appliquer l'inférence classique sur cette sous-base cohérente comme le décrit la définition suivante :

Définition 72 *Soit* $(\Sigma, \leq)$ *une base de croyances stratifiées et soit* ψ *une formule. Alors* ψ *est une* ***conséquence linéaire****, noté* $(\Sigma, \leq) \vdash_{lin} \psi$*, si et seulement si* ψ *est une conséquence classique de la sous-base cohérente* $Lin(\Sigma, \leq)$*.*

$$(\Sigma, \leq) \vdash_{lin} \psi \text{ si et seulement si } Lin(\Sigma, \leq) \models \psi.$$

Exemple 16 *Soit* $(\Sigma, \leq) = (S_1, S_2, S_3)$ *une base de croyances stratifiées telle que :*

- $S_1 = \{\neg a \vee b, a\}$,
- $S_2 = \{\neg b, c\}$,
- $S_3 = \{\neg b \vee \neg d\}$.

La première strate S_1 *est cohérente. Regardons maintenant* $S_1 \cup S_2$*. Cette base est incohérente ce qui implique que la strate* S_2 *est ignorée et on passe à la strate suivante, c'est-à-dire* S_3*. La base* $S_1 \cup S_3$ *est cohérente. Par conséquent,* $Lin(\Sigma, \leq) = S_1 \cup S_3 = \{\neg a \vee b, a, \neg b \vee \neg d\}$*. En appliquant l'inférence classique sur* $Lin(\Sigma, \leq)$*, on peut déduire par exemple* $\neg d$ *:* $Lin(\Sigma, \leq) \models \neg d$*. De ce fait,* $(\Sigma, \leq) \vdash_{lin} \neg d$*.*

4.3.3 Relations de préférence entre sous-bases cohérentes d'une base de croyances stratifiées

L'idée de sélectionner des sous-bases cohérentes en utilisant des priorités n'est pas nouvelle puisqu'elle remonte au moins à la proposition donnée dans [85]. Ainsi, de nombreuses relations de préférence entre sous-bases cohérentes d'une base de croyances stratifiées ont été proposées. Parmi elles, on peut citer les plus fréquemment utilisées à savoir :

- la préférence best-out,
- la préférence de l'inclusion et
- la préférence lexicographique.

Préférence best-out

La préférence best-out présentée dans [9, 47] est définie entre des sous-bases cohérentes d'une base de croyances stratifiées comme suit :

Notation 1 *Soit* $(\Sigma, \leq) = (S_1, \ldots, S_m)$ *une base de croyances stratifiées et soit* A *une sous-base de* Σ*. On note par* $a(A)$ *la plus haute priorité d'une formule de* Σ *qui n'est pas dans* A *:*

$$a(A) = min\{i \text{ tel que } \exists \varphi \in S_i \setminus A\} \text{ avec la convention que } min\emptyset = n + 1.$$

Définition 73 *Soient* A *et* B *deux sous-bases cohérentes d'une base de croyances stratifiées* $(\Sigma, \leq) = (S_1, \ldots, S_m)$*. Alors* A *est dite préférée à* B *selon la* ***préférence best-out****, noté* $A \leq_{bo} B$*, si et seulement si* $a(B) \leq a(A)$*.*

Pour mieux comprendre cette définition, prenons l'exemple suivant :

Exemple 17 *Soit* $(\Sigma, \leq) = (S_1, S_2, S_3)$ *telle que :*
- $S_1 = \{a, \neg a \vee b\}$,
- $S_2 = \{\neg a \vee \neg c\}$,
- $S_3 = \{\neg b \vee c, \neg b \vee d\}$.

Considérons deux sous-bases cohérentes A *et* B *telles que* $A = \{a, \neg a \vee b\}$ *et* $B = \{a, \neg a \vee b, \neg a \vee \neg c, \neg b \vee d\}$. *Alors, comme* $a(A) = 2 \leq a(B) = 3$, *on déduit que* $A \leq_{bo} B$.

La préférence best-out induit un préordre total sur l'ensemble des sous-bases cohérentes. De plus, les sous-bases cohérentes préférées selon la préférence best-out ne sont pas nécessairement des sous-bases maximales cohérentes.

Par ailleurs, il s'agit d'une préférence très peu sélective. En effet, dans le cas d'une base ayant une seule strate, toutes les sous-bases cohérentes sont préférées. En général, étant donnée une base de croyances stratifiées $(\Sigma, \leq) = (S_1, \ldots, S_m)$, soit k l'indice maximal tel que $S_1 \cup \ldots \cup S_k$ soit cohérente, c'est-à-dire $k = Inc(\Sigma, \leq) - 1$. Alors $A = S_1 \cup \ldots \cup S_k$ est une sous-base cohérente préférée pour la préférence best-out ainsi que n'importe quelle sous-base cohérente de Σ contenant A.

Préférence pour l'inclusion

Dans [21], Brewka généralise l'approche proposée par Poole pour le raisonnement par défaut [83], et ce en permettant d'introduire plus de deux niveaux de priorité. La notion de sous-base maximale cohérente préférée de Brewka, qu'il appelle également sous-théorie préférée, est donnée comme suit.

Définition 74 *Soit* $(\Sigma, \leq) = (S_1, \ldots, S_n)$ *une base de croyances stratifiées. Soit* A *une sous-base de* $(\Sigma, \leq)$. A *est une* ***sous-théorie préférée*** *si et seulement si pour tout* i, $1 \leq i \leq m$: $(A \cap S_1) \cup \ldots \cup (A \cap S_i)$ *est une sous-base maximale cohérente de* $S_1 \cup \ldots \cup S_i$.

En conséquence, pour obtenir une sous-théorie préférée de $(\Sigma, \leq) = (S_1, \ldots, S_m)$, on commence par n'importe quelle sous-base maximale cohérente de S_1, on rajoute autant de formules de S_2 que possible tout en préservant la cohérence, et on continue ce processus pour S_2, S_3, etc.

Il est claire que les sous-théories préférées de Brewka sont des sous-bases maximales cohérentes. En particulier, dans le cas d'une base ne contenant qu'une seule strate, l'ensemble des sous-théories préférées coïncide avec celui des sous-bases maximales cohérentes.

Dans [28], Cayrol, Royer et Saurel ont introduit une relation de préférence entre sous-bases cohérentes d'une base de croyances stratifiées qu'ils ont baptisée préférence basée-stratification, communément appelée par la suite préférence pour l'inclusion.

Définition 75 *Soit une base de croyances stratifiées $(\Sigma, \leq) = (S_1, \ldots, S_m)$. Soient A et B deux sous-bases cohérentes de Σ. Alors, A est dite préférée à B selon la* ***préférence pour l'inclusion****, noté $A <_{incl} B$, si et seulement si A et B contiennent les mêmes formules de la première strate jusqu'à une certaine strate S_i où les éléments de B sont strictement inclus dans ceux de A. Plus formellement,*

$$\exists i,\ 1 \leq i \leq m \text{ tel que } S_i \cap B \subset S_i \cap A \text{ et } \forall j,\ 1 \leq j < i,\ S_j \cap B = S_j \cap A.$$

Voyons ce que donne cette définition sur l'exemple suivant :

Exemple 18 *Soit $(\Sigma, \leq) = (S_1, S_2, S_3)$ une base de croyances stratifiées telle que*
- $S_1 = \{r \wedge q \wedge e\}$
- $S_2 = \{\neg r \vee \neg p, \neg q \vee p, \neg e \vee p\}$
- $S_3 = \{\neg e \vee l\}$

Soient deux sous-bases cohérentes A et B telles que $A = \{r \wedge q \wedge e, \neg q \vee p, \neg e \vee p\}$ et $B = \{r \wedge q \wedge e, \neg q \vee p, \neg e \vee l\}$.

On a alors :
- $A \cap S_1 = B \cap S_1$.
- $B \cap S_2 \subset A \cap S_2$.

En conséquence, on peut conclure que $A <_{incl} B$.

Cayrol, Royer et Saurel ont démontré que la préférence pour l'inclusion $<_{incl}$ est un ordre partiel strict sur l'ensemble des sous-bases cohérentes.

En outre, ils ont prouvé qu'une sous-base cohérente A est préférée selon la préférence pour l'inclusion si et seulement si pour tout i, $1 \leq i \leq m$: $(A \cap S_1) \cup \ldots \cup (A \cap S_i)$ est une sous-base maximale cohérente de $S_1 \cup \ldots \cup S_i$. En d'autres termes, les sous-bases cohérentes préférées pour $<_{incl}$, $Min(Cons(\Sigma, \leq), <_{incl})$, coïncident avec les sous-théories préférées de Brewka.

Enfin, il est à noter que les sous-bases cohérentes préférées selon la préférence pour l'inclusion sont appelées des sous-bases fortement maximales cohérentes dans [47].

Préférence lexicographique

La préférence lexicographique entre les sous-bases cohérentes d'une base de croyances stratifiées, introduite par [9, 67], est définie de manière similaire à la préférence pour l'inclusion, et ce en remplaçant le critère d'inclusion ensembliste par un critère de cardinalité.

Définition 76 *Soit une base de croyances stratifiées* $(\Sigma, \leq) = (S_1, \ldots, S_m)$. *Soient* A *et* B *deux sous-bases cohérentes de* Σ. *Alors,* A *est dite* ***lexicographiquement préférée*** *à* B, *noté* $A <_{lex} B$, *si et seulement si*

$$\exists i,\ 1 \leq i \leq m \text{ tel que } |S_i \cap A| > |S_i \cap B| \text{ [2] et } \forall j,\ 1 \leq j < i,\ |S_j \cap B| = |S_j \cap A|.$$

Appliquons cette définition sur l'exemple suivant :

Exemple 19 *Soit* $(\Sigma, \leq) = (S_1, S_2, S_3)$ *une base de croyances stratifiées telle que*
- $S_1 = \{r \wedge q \wedge e\}$
- $S_2 = \{\neg r \vee \neg p, \neg q \vee p\}$
- $S_3 = \{\neg e \vee p\}$

Considérons ensuite deux sous-bases cohérentes A *et* B *où :* $A = \{r \wedge q \wedge e, \neg r \vee \neg p\}$ *et* $B = \{r \wedge q \wedge e, \neg q \vee p, \neg e \vee p\}$.

Alors, ont peut aisément vérifier que :
- $|A \cap S_1| = 1$, $|A \cap S_2| = 1$ *et* $|A \cap S_3| = 0$,
- $|B \cap S_1| = 1$, $|B \cap S_2| = 1$ *et* $|B \cap S_3| = 1$.

De ce fait, on peut déduire que $A <_{lex} B$ *car la strate* S_3 *contient plus d'éléments de* A *que de* B *alors qu'on a égalité pour les strates supérieures.*

L'égalité est donnée par $A =_{lex} B$ si et seulement si $\forall i, 1 \leq i \leq m : |A \cap S_i| = |B \cap S_i|$.

Par ailleurs, il a été prouvé dans [9] que la préférence lexicographique raffine la préférence pour l'inclusion dans le sens où toute sous-base préférée lexicographiquement préférée se trouve préférée selon la préférence pour l'inclusion :

$$Min(Cons(\Sigma, \leq), <_{lex}) \subset Min(Cons(\Sigma, \leq), <_{incl}).$$

Par conséquent, quoique la préférence lexicographique soit définie sur des sous-bases cohérentes, les éléments préférés dans ce cas sont des sous-bases maximales cohérentes d'où l'idée d'appliquer la préférence lexicographique directement sur les sous-bases maximales cohérentes.

De plus, il est à noter que la préférence lexicographique définit un ordre strict total sur l'ensemble des sous-bases cohérentes contrairement à la préférence pour l'inclusion qui définit un ordre strict partiel.

Enfin, soit $(\Sigma, \leq) = (S_1, \ldots, S_m)$ une base de croyances stratifiées. Alors, une sous-base A de $(\Sigma, \leq)$ est une sous-base cohérente lexicographique préférée si et seulement si $\forall i, 1 \leq i \leq m : (S_1 \cap A) \cup \ldots \cup (S_i \cap A)$ est une sous-base cohérente maximale pour la cardinalité de $S_1 \cup \ldots \cup S_i$.

[2] $|A|$ dénote le nombre de formules de A.

4.3.4 Relations d'inférence basées sur la préférence entre sous-bases cohérentes

Étant donnée une relation de préférence $\mathcal{P}$ entre les sous-bases cohérentes d'une base de croyances stratifiées $(\Sigma, \leq) = (S_1, \ldots, S_m)$, plusieurs principes d'inférence peuvent être considérés [82, 26]. Les trois principaux sont :

- le principe universel,
- le principe existentiel,
- le principe argumentatif.

Le principe universel, appelé également principe fort, a été introduit dans [86] et permet de définir un raisonnement circonspect, et ce en prenant le maximum de précaution avant d'inférer une nouvelle conséquence.

Définition 77 *Une formule ψ est une **conséquence forte** de $(\Sigma, \leq)$ relativement à $\mathcal{P}$ si et seulement si $\forall B \in Min(Cons(\Sigma, \leq), \mathcal{P})$, $B \models \psi$.*

Par opposition au principe universel, le principe existentiel, dit aussi principe faible, met en œuvre un raisonnement crédule.

Définition 78 *Une formule ψ est une **conséquence faible** de $(\Sigma, \leq)$ relativement à $\mathcal{P}$ si et seulement si $\exists B \in Min(Cons(\Sigma, \leq), \mathcal{P})$, $B \models \psi$.*

Remarquons que le principe existentiel permet d'inférer à la fois une formule et sa négation. Quant au principe argumentatif, il a été introduit dans [10] dans le cadre de l'argumentation.

Définition 79 *Une formule ψ est une **conséquence argumentative** de $(\Sigma, \leq)$ relativement à $\mathcal{P}$ si et seulement si $\exists B \in Min(Cons(\Sigma, \leq), \mathcal{P})$, $B \models \psi$ et $\forall B' \in Min(Cons(\Sigma, \leq), \mathcal{P}), B' \not\models \neg\psi$.*

Le principe universel conduit aux relations d'inférence les plus rationnelles, c'est-à-dire des relations qui vérifient un ensemble maximum de postulats de rationalité. De plus, contrairement au principe existentiel, il évite la trivialisation, c'est-à-dire inférer à la fois une formule et sa négation. Quant au principe argumentatif, il est intimement lié à l'argumentation qui sort du cadre de notre étude. Par conséquent, nous ne considérerons dans ce qui suit que le principe universel.

Maintenant, combiner les relations de préférence entre les sous-bases cohérentes et le principe universel induit plusieurs relations d'inférence permettant de raisonner en présence d'incohérence sans trivialisation. Ainsi, on obtient l'inférence best-out, l'inférence lexicographique et l'inférence selon la préférence pour l'inclusion.

Inférence relative à la préférence best-out et inférence possibiliste

L'inférence relativement à la préférence best-out est définie comme suit :

Définition 80 *Soit $(\Sigma, \leq)$ une base de croyances stratifiées et soit ψ une formule. Alors ψ est une* ***conséquence relativement à la préférence best-out****, noté $(\Sigma, \leq) \vdash_{bo} \psi$, si et seulement si ψ est une conséquence classique de chaque sous-base cohérente préférée selon la préférence best-out :*

$$(\Sigma, \leq) \vdash_{bo} \psi \text{ si et seulement si } \forall A \in Min(Cons(\Sigma, \leq), <_{bo}), B \models \psi.$$

Comme ç'a été noté précédemment, l'ensemble $Min(Cons(\Sigma, \leq), <_{bo})$ contient au moins la sous-base $S_1 \cup \ldots \cup S_k$ où k est l'indice maximal tel que $S_1 \cup \ldots \cup S_k$ soit cohérente. Par conséquent,

$$(\Sigma, \leq) \vdash_{bo} \psi \text{ si et seulement si } S_1 \cup \ldots \cup S_k \models \psi.$$

Donc, l'inférence relative à la préférence best-out est équivalente à l'inférence possibiliste simple donnée dans ce chapitre par la définition 67.

$$(\Sigma, \leq) \vdash_{bo} \psi \text{ si et seulement si } (\Sigma, \leq) \vdash_{\pi} \psi$$

Aucune formule de priorité inférieure à k n'est impliqué dans le processus d'inférence. Ceci est **l'effet de la noyade** dont souffre l'inférence possibiliste.

Inférence relativement à la préférence pour l'inclusion

Quant à l'inférence relativement à la préférence pour l'inclusion, elle consiste à appliquer l'inférence classique à partir de toutes les sous-bases cohérentes préférées pour l'inclusion. Plus formellement :

Définition 81 *Soit $(\Sigma, \leq)$ une base de croyances stratifiées et soit ψ une formule. Alors ψ est une* ***conséquence relativement à la préférence pour l'inclusion****, noté par $(\Sigma, \leq) \vdash_{incl} \psi$, si et seulement si ψ est une conséquence classique de chaque sous-base cohérente préférée selon la préférence pour l'inclusion.*

$$(\Sigma, \leq) \vdash_{incl} \psi \text{ si et seulement si } \forall A \in Min(Cons(\Sigma, \leq), <_{incl}), B \models \psi.$$

Prenons l'exemple suivant pour illustrer cette définition.

Exemple 20 *Considérons encore une fois la base de croyances stratifiées donnée par l'exemple 18. Dans ce cas, on a trois sous-bases maximales cohérentes A, B et C telles que $A = \{r \wedge q \wedge e, \neg r \vee \neg p, \neg e \vee l\}$, $B = \{r \wedge q \wedge e, \neg q \vee p, \neg e \vee p, \neg e \vee l\}$ et $C = \{\neg q \vee p, \neg e \vee p, \neg e \vee l, \neg r \vee \neg p\}$.*

Alors, d'une part, on a $A <_{incl} C$ et $B <_{incl} C$. D'autre part, on n'a ni $A <_{incl} B$ ni $B <_{incl} A$. Ceci veut dire que $Min(Cons(\Sigma, \leq), <_{incl}) = \{A, B\}$. Donc, dans ce cas l'inférence $\vdash_{incl}$ à partir de $(\Sigma, \leq)$ revient à appliquer l'inférence classique à partir de A et B à la fois. Par exemple, comme $A \models l$ et $B \models l$, ont peut dériver le fait que $(\Sigma, \leq) \vdash_{incl} l$.

Inférence lexicographique

Enfin, l'inférence lexicographique est donnée par la définition suivante :

Définition 82 *Soit $(\Sigma, \leq)$ une base de croyances stratifiées et soit ψ une formule. Alors ψ est une **conséquence lexicographique** de $(\Sigma, \leq)$, notée $(\Sigma, \leq) \vdash_{lex} \psi$, si et seulement si ψ est une conséquence classique de chaque sous-base cohérente lexicographiquement préférée.*

$$(\Sigma, \leq) \vdash_{lex} \psi \text{ si et seulement si } \forall A \in Min(Cons(\Sigma, \leq), <_{lex}), B \models \psi.$$

Exemple 21 *Considérons à nouveau la base de croyances stratifiées donnée par l'exemple 19. Cette base admet trois bases maximales cohérentes A, B et C telles que $A = \{r \wedge q \wedge e, \neg r \vee \neg p\}, B = \{r \wedge q \wedge e, \neg q \vee p, \neg e \vee p\}$ et $C = \{\neg r \vee \neg p, \neg q \vee p, \neg e \vee p\}$ qui vérifie ce qui suit :*

- $|A \cap S_1| = 1$, $|A \cap S_2| = 1$ *et* $|A \cap S_3| = 0$,
- $|B \cap S_1| = 1$, $|B \cap S_2| = 1$ *et* $|B \cap S_3| = 1$,
- $|C \cap S_1| = 0$, $|C \cap S_2| = 2$ *et* $|C \cap S_3| = 1$.

Par conséquent, on déduit que $B <_{lex} A <_{lex} C$ ce qui veut dire que $Min(Cons(\Sigma, \leq), <_{lex}) = B$. Ainsi, l'inférence lexicographique à partir de $(\Sigma, \leq)$ revient à appliquer l'inférence classique à partir de la base classique B. Par exemple, $B \models p$ équivaut à dire que $(\Sigma, \leq) \vdash_{lex} p$.

4.3.5 Comparaison des différentes relations d'inférence

Le but de cette section est de rappeler les résultats de complexité, propriétés logiques et prudence connus par rapport aux relations d'inférence à partir de bases de croyances stratifiées décrites précédemment.

Complexité

Soient POS, LIN, INCL et LEX les problèmes de décision associés respectivement à $\vdash_\pi$, $\vdash_{lin}$, $\vdash_{incl}$ et $\vdash_{lex}$.

Alors, il a été prouvé que POS est un problème $\Delta_2^p[O(log\ n)]$-complet [79], c'est-à-dire un problème qui requiert un nombre d'appels logarithmique à un oracle NP pour être résolu. En ce qui concerne l'inférence linéaire, il a été démontré que le problème de décision correspondant est un problème Δ_2^P-complet [78]. Autrement dit, il s'agit d'un problème qui nécessite un nombre d'appels polynomial à un oracle NP. Le même résultat a été obtenu dans le cas de l'inférence lexicographique dans [25]. En effet, LEX est un problème Δ_2^p-complet. Enfin, [77], le problème lié à l'inférence relativement à la préférence pour l'inclusion est, INCL, est Π_2^p-complet. Ces résultats sont résumés via le tableau 4.1.

Problème d'inférence	Complexité
POS	$\Delta_2^p[O(log\ n)]$-complet
LIN	Δ_2^p-complet
INCL	Π_2^p-complet
LEX	Δ_2^p-complet

TAB. 4.1 – Complexité de l'inférence à partir d'une base de croyances stratifiées

En conclusion, les problèmes de décision associés aux relations d'inférence à partir de bases de croyances stratifiées sont typiquement au premier niveau de la hiérarchie polynomiale, à l'exception du problème associé à l'inférence relative à la préférence pour l'inclusion qui se trouve au second niveau.

Propriétés de déduction

En vue de respecter exactement la terminologie des propriétés de déduction telles qu'elles ont été définies par Kraus, Lehmann et Magidor [64], il convient de considérer une généralisation de relations d'inférence entre deux formules pour un contexte donné. Plusieurs extensions peuvent être considérées parmi lesquelles l'extension donnée par la définition suivante :

Définition 83 *Soit $(\Sigma, \leq)$ une base de croyances stratifiées et soient φ et ψ deux formules propositionnelles. Alors, φ infère ψ par rapport à $(\Sigma, \leq)$ relativement à l'inférence R où $R \in \{bo, lin, incl, lex\}$, noté $\varphi \mathrel{|\!\sim}_R^{(\Sigma,\leq)} \psi$, si et seulement si $(\Sigma \cup \{\varphi\}, \leq^\psi) \vdash_R \psi$ où $\leq^\varphi$ est un nouveau préordre qui étend le préordre $\leq$ de telle sorte que la formule φ soit plus prioritaire que toutes les formules de Σ (ce qui revient à rajouter une nouvelle strate à Σ ne contenant que $\{\varphi\}$) :*

– $\forall \alpha, \beta \in \Sigma : \alpha \leq \beta$ *si et seulement si* $\alpha \leq^{\varphi} \beta$;
– $\forall \alpha \in \Sigma : \varphi \prec^{\varphi} \alpha$.

Notons que cette extension est bien connue dans la littérature et reflète le point de vue de la révision. De plus, elle généralise d'autres extensions permettant de définir une relation d'inférence entre formules par rapport à un contexte donné. Par ailleurs, il est claire que cette extension subsume les relations d'inférence de la forme $\vdash_R$ étant donnée que $(\Sigma, \leq) \vdash_R \varphi$ si et seulement si $\top \mid\!\sim_R^{(\Sigma,\leq)} \varphi$.

Dans [9], Benferhat, Cayrol, Dubois, Lang et Prade ont démontré les résultats suivants :
– L'inférence relative à la préférence pour l'inclusion est une relation préférentielle.
– L'inférence possibiliste est une relation rationnelle.
– L'inférence lexicographique est une relation rationnelle.

Prudence

La prudence d'une relation d'inférence est liée à l'ensemble des conséquences déductibles par cette relation. Plus la relation est prudente, plus l'ensemble de ses conséquences est restreint.

Définition 84 *Soient R_1 et R_2 deux relations d'inférence. R_1 est dite* ***au moins aussi prudente*** *(resp.* ***plus prudente****) que R_2, si et seulement si pour toute base de croyance Σ, l'ensemble des conséquences déductibles de R_1 est inclus (resp. strictement inclus) dans l'ensemble des conséquences déductibles de R_2.*

On dit aussi que la relation d'inférence R_1 est plus aventureuse ou plus productive que la relation d'inférence R_2 si et seulement si R_2 est plus prudente que R_1.

La relation de prudence entre les différentes relations d'inférence à partir de bases de croyances stratifiées est décrite par la figure 4.1 où une flèche $R_i \rightarrow R_j$ signifie que R_i est plus prudente que R_j.

– Concernant l'inférence possibiliste, une seule sous-base cohérente préférée est sélectionnée. Toutes les conséquences déductibles par cette relation sont les conséquences classiques de cette sous-base. Par ailleurs, pour les trois autres inférences, on raisonne classiquement à partir de sous-bases cohérentes contenant au moins la sous-base cohérente sélectionnée par l'inférence possibiliste. Or, la logique classique étant monotone, l'ensemble des conséquences croît avec l'ensemble des croyances à partir desquelles

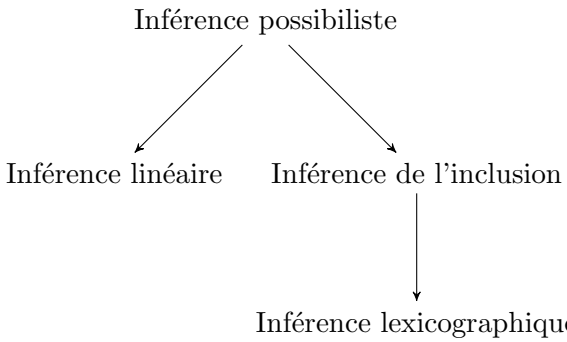

FIG. 4.1 – Prudence des relations d'inférence à partir de bases de croyances stratifiées

on raisonne. En conséquence, l'inférence possibiliste est plus prudente que les autres inférences. Une autre explication réside dans le fait que toute base préférée selon la préférence pour l'inclusion est préférée selon la préférence best-out [9].

– L'inférence linéaire n'est pas comparable avec l'inférence lexicographique et l'inférence relative à la préférence pour l'inclusion car la sous-base cohérente préférée pour la préférence linéaire n'est pas comparable avec les sous-bases cohérentes préférées des deux autres inférences.
– Enfin, les sous-bases préférées lexicographiquement sont aussi préférées selon la préférence pour l'inclusion [9]. Par conséquent, l'inférence relative à la préférence pour l'inclusion est plus prudente que l'inférence lexicographique.

4.4 Inférence à partir de bases de croyances partiellement préordonnées

4.4.1 Flexibilité des bases de croyances partiellement préordonnées

Dans de nombreuses applications, les informations proviennent de plusieurs sources hétérogènes, indépendantes et souvent entachées d'incertitude, alors préordonner totalement ces informations est difficile, et de plus génère souvent des résultats indésirables car trop aventureux.

Ainsi, la relation de priorité entre les informations disponibles n'est que partiellement définie. Les préordres partiels offrent nettement plus de flexibilité que les préordres totaux afin de représenter des informations incomplètes. En outre, ils permettent d'éviter de comparer des informations incomparables.

Le besoin de raisonner en présence d'incohérence à partir de bases de croyances par-

tiellement préordonnées est d'autant plus important si on s'intéresse à la dynamique de croyances. En effet, même si les informations disponibles sont totalement préordonnées, il se peut que, lors de l'utilisation de certains opérateurs de mise à jour [63], l'intégration d'une nouvelle information mène à préordonner partiellement la base de croyances en question.

Nous rappelons, dans un premier temps, la notion de bases de croyances totalement préordonnées compatibles avec une base de croyances partiellement préordonnées. Une base de croyances totalement préordonnées $(\Sigma, \leq)$ est dite compatible avec une autre base de croyances partiellement préordonnées $(\Sigma, \preceq)$ si et seulement si le préordre total $\leq$ étend le préordre partiel $\preceq$. Plus formellement :

Définition 85 *Soit $(\Sigma, \preceq)$ une base de croyances partiellement préordonnées. Une base de croyances totalement préordonnées $(\Sigma, \leq)$ est dite* ***compatible*** *avec $(\Sigma, \preceq)$ ssi $\leq$ est compatible avec $\preceq$, c'est à dire :*

1. $\forall \varphi, \psi \in \Sigma$: *si* $\varphi \preceq \psi$ *alors* $\varphi \leq \psi$,
2. $\forall \varphi, \psi \in \Sigma$: *si* $\varphi \prec \psi$ *alors* $\varphi < \psi$.

Par la suite, l'ensemble de toutes les bases de croyances totalement préordonnées compatibles avec $(\Sigma, \preceq)$ est noté $Comp(\Sigma, \preceq)$.

Plusieurs relations d'inférence à partir de bases de croyances partiellement préordonnées ont été définies par extension de relations d'inférence à partir de bases de croyances stratifiées, à l'instar de l'inférence démocratique [28] et de l'inférence relative à la préférence pour l'inclusion basée-compatible [21, 62] qui généralisent l'inférence relative à la préférence pour l'inclusion. Quant à l'inférence possibiliste, elle a été étendue par l'inférence possibiliste forte et l'inférence possibiliste faible [5].

4.4.2 L'inférence démocratique

La préférence démocratique proposée dans [28] est définie comme suit.

Définition 86 *Soient $A, B \in Cons(\Sigma)$. Alors, A est dite* ***démocratiquement préférée*** *à B, notée par $A \prec_{demo} B$, si et seulement si $\forall x \in B \setminus A, \exists y \in A \setminus B$ tel que $a \prec b$.*

La préférence démocratique induit un préordre partiel sur $Cons(\Sigma, \preceq)$. De plus, elle généralise la préférence pour l'inclusion.

La préférence démocratique préfère les sous-bases maximales cohérentes, c'est-à-dire $Min(Cons(\Sigma, \preceq), \prec_{demo}) \subseteq MaxCons(\Sigma, \preceq)$.

Le terme démocratique provient de travaux sur le choix social : cette appellation revient au fait qu'en préférence démocratique tout ce qui est enlevé est remplacé par quelque chose de meilleure [2].

La préférence démocratique a été utilisée dans d'autres travaux sous d'autres appellations. Il s'agit de la même relation qui définit des modèles parfaits dans les bases de données stratifiables [84]. Elle a été utilisée dans le contexte de théories de défauts ordonnées [54].

L'inférence relative à la préférence démocratique est donnée comme suit :

Définition 87 *Une formule ψ est dite* ***conséquence démocratique*** *de $(\Sigma, \preceq)$, noté par $(\Sigma, \preceq) \vdash_{demo} \psi$, si et seulement si*

$$\forall B \in Min(Cons(\Sigma, \preceq), \prec_{demo}), B \models \psi.$$

Exemple 22 *Considérons une base de croyances partiellement préordonnées $(\Sigma, \preceq)$ telle que $\Sigma = \{a, \neg a, b, \neg b\}$ et que le préordre $\preceq$ soit représenté par la figure 4.2 où $x \rightarrow y$ signifie que $x \prec y$.*

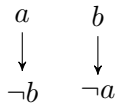

FIG. 4.2 – Exemple de $(\Sigma, \preceq)$

Clairement, cette base admet quatre sous-bases maximales cohérentes A, B, C et D telles que $A = \{a, b\}$, $B = \{a, \neg b\}$, $C = \{b, \neg a\}$ et $D = \{\neg b, \neg a\}$.

Alors, on a par exemple $A \prec_{demo} D$. En effet, $\forall x \in D \setminus A, \exists y \in A \setminus D$ tel que $y \prec x$. Par ailleurs, on n'a pas $A \preceq_{demo} B$ car $\exists \neg b \in B \setminus A$ tel que $\nexists x \in A \setminus B$ avec $x \preceq \neg b$. De la même manière, on vérifie qu'on n'a pas $B \prec_{demo} A$ ce qui veut dire que ces deux sous-bases sont incomparables démocratiquement : $A \sim_{demo} B$.

Enfin, on obtient $Min(Cons(\Sigma, \preceq), \prec_{demo}) = \{A, B, C\}$. Par exemple, puisque $A \models a \vee b$, $B \models a \vee b$ et $C \models a \vee b$, on déduit que $(\Sigma, \preceq) \vdash_{demo} a \vee b$.

4.4.3 L'inférence relative à la préférence pour l'inclusion basée-compatibles

L'inférence relative à la préférence pour l'inclusion basée-compatibles proposée dans [21, 62] utilise, comme son nom l'indique, la notion de bases de croyances totalement préordonnées compatibles avec une base de croyances partiellement préordonnées.

Définition 88 *Une sous-base $A \in Cons(\Sigma)$ est dite* ***sous-base cohérente préférée pour l'inclusion basée-compatibles*** *si et seulement s'il existe une base totalement préordonnées $(\Sigma, \leq)$ compatible avec $(\Sigma, \preceq)$ telle que $A \in Min(Cons(\Sigma, \leq), <_{incl})$.*

L'ensemble de toutes les sous-bases cohérentes préférées pour l'inclusion basée-compatibles de $(\Sigma, \preceq)$ est noté par $CmpIncl(\Sigma, \preceq)$.

Définition 89 *Une formule ψ est dite* ***conséquence relative à la préférence pour l'inclusion basée-compatibles*** *de $(\Sigma, \preceq)$, noté par $(\Sigma, \preceq) \vdash_{cincl} \psi$, si et seulement si*

$$\forall B \in CmpIncl(\Sigma, \preceq),\ B \models \psi.$$

4.4.4 Inférence possibiliste forte et inférence possibiliste faible

Deux relations de préférence entre les sous-bases cohérentes d'une base de croyances partiellement préordonnées ont été proposées dans [5] à savoir la préférence best-out forte et la préférence best-out faible.

La préférence best-out forte consiste à dire qu'une sous-base cohérente A est préférée à une autre sous-base cohérente B s'il existe un élément en dehors de B qui est préféré à tous les éléments qui sont en dehors de A. Plus formellement,

Définition 90 *Soient $A, B \in Cons(\Sigma)$. Alors, A est préférée à B par rapport à la* ***préférence best-out forte****, notée par $A \prec_{sbo} B$, si et seulement si $\exists b \notin B$ tel que $\forall a \notin A, b \prec a$.*

Une des limitations de cette préférence est qu'elle génère de nombreuses incomparabilités contrairement à la préférence best-out faible qui consiste à dire que A est préférée à B si pour chaque élément en dehors de A, il existe un élément en dehors de B qui lui est préféré.

Définition 91 *Soient $A, B \in Cons(\Sigma)$. Alors, A est préférée à B par rapport à la* ***préférence best-out faible****, notée par $A \prec_{wbo} B$ si et seulement si $\forall a \notin A, \exists b \notin B$ tel que $b \prec a$.*

La préférence best-out forte et la préférence best-out faible sont des extensions de la préférence best-out donnée par la définition 73 dans le cadre de bases de croyances stratifiées.

En se basant sur ces deux relations de préférence, on définit les inférences possibilistes forte et faible de la manière suivante :

Définition 92 *Soit $(\Sigma, \preceq)$ une base de croyances partiellement préordonnées, et soit ψ une formule classique.*

- *ψ est une **conséquence possibiliste forte de** $(\Sigma, \preceq)$, noté $(\Sigma, \preceq) \vdash_{spos} \psi$, si et seulement si $\forall B \in Min(Cons(\Sigma, \preceq), \prec_{sbo})$, $B \models \psi$.*
- *ψ est une **conséquence possibiliste faible de** $(\Sigma, \preceq)$, noté $(\Sigma, \preceq) \vdash_{wpos} \psi$, si et seulement si $\forall B \in Min((\Sigma, \preceq), \prec_{wbo})$, $B \models \psi$*

L'inférence possibiliste faible suscite un intérêt particulier : Benferhat, Lagrue et Papini [5] qualifient cette inférence de naturelle et intuitive. En effet, ils ont montré qu'une formule ψ est conséquence possibiliste faible d'une base de croyances partiellement préordonnées si et seulement si ψ est conséquence possibiliste de chaque base totalement préordonnée compatible avec $(\Sigma, \preceq)$:

$$(\Sigma, \preceq) \vdash_{wbo} \psi \text{ si et seulement si } \forall (\Sigma, \leq) \in Comp(\Sigma, \leq), (\Sigma, \leq) \vdash_{\pi} \psi.$$

Enfin, notons que l'inférence possibiliste faible a été étendue par Benferhat et Prade dans [14] afin de raisonner à partir de bases de croyances symboliquement pondérées. Dans de telles bases, les degrés de priorité associés aux formules peuvent être donnés sous forme d'expressions composées en utilisant les opérateurs maximum et minimum, et donc lorsque ces poids sont donnés sous forme atomique, les bases symboliquement pondérées peuvent être vues comme des bases partiellement préordonnées.

4.5 Conclusion

Les approches basées sur la restauration de la cohérence sont des approches très naturelles qui permettent de raisonner en présence d'incohérence sans trivialisation, c'est-à-dire sans déduire une formule et sa négation en même temps. De plus, ces approches offrent la possibilité de tirer profit de toute la machinerie de la logique classique. Ces approches peuvent être en général classifiées en deux grandes familles. D'une part, on a celles qui sont destinées à des bases de croyances totalement préordonnées ou stratifiées, et d'autre part, on a celles qui sont dédiées à des bases de croyances partiellement préordonnées, et qui sont en général obtenues par extension des approches de la première famille. Dans ce chapitre, nous avons passé en revue les approches les plus fréquemment utilisées des deux familles.

Par ailleurs, nous avons rappelé entre autres les résultats de complexité obtenus dans le cadre des bases de croyances totalement préordonnées. Ces résultats montrent qu'il s'agit d'approches coûteuses d'un point de vue calculatoire, ce qui est facilement limitatif d'un point de vue pratique. A titre d'exemple, dans [6], les auteurs proposent des méthodes de contrôle d'accès en sécurité informatique qui s'appuient sur les approches basées sur la restauration de la cohérence à partir de bases de croyances totalement préordonnées. Or,

dans ce contexte, le temps de réponse est un facteur crucial ce qui reflète le fait que le coût induit par de telles approches représente un problème considérable. Par conséquent, il est vraiment nécessaire d'avoir des approches permettant de pallier cet éccueil.

En ce qui concerne les relations d'inférence à partir de bases de croyances partiellement préordonnées, on peut constater de prime abord l'absence d'une extension de l'inférence lexicographique classique. D'autres part, les relations d'inférence dans ce cas et contrairement au cas totalement préordonné, n'ont pas été suffisamment analysées relativement à la complexité, les propriétés logiques et la prudence.

La deuxième partie de ce mémoire traite entre autres les différents points que nous venons de souligner à savoir les problèmes liés à la complexité des relations d'inférence à partir de bases de croyances totalement préordonnées, l'extension de l'inférence lexicographique au cas de bases de croyances partiellement préordonnées ainsi que l'étude des différentes relation d'inférence dédiées au dernier cas. Avant de passer aux contributions, rappellons d'abord la compilation de connaissances.

Chapitre 5

Compilation de connaissances

5.1 Introduction

En Informatique, la compilation est une technique assez fréquente par laquelle on entend une transformation d'une représentation d'entrée vers une représentation de sortie. Cette transformation comporte une réécriture en un langage approprié, par exemple passage d'un programme écrit dans un langage de haut niveau vers un code écrit en langage machine.

En Intelligence Artificielle, en particulier, la compilation de bases de connaissances désigne tout prétraitement qui réduit l'ordre de complexité des algorithmes mis en œuvres dans l'exploitation de cette base. En effet, la clef en compilation de connaissances est qu'un problème est conceptuellement divisé en deux parties : une base de connaissances et une requête. Tandis que la requête est variable, la base de connaissances est moins fréquemment modifiée et la même base est utilisée plusieurs fois afin de répondre à un bon nombre de requêtes [23].

Dans ce chapitre, nous présentons une introduction à la compilation de connaissances. Après la présentation du principe correspondant, nous passons en revue les méthodes classiques de compilation exactes, notamment celles basées sur les impliqués premiers ainsi que les méthodes classiques de compilation approximative. Ensuite, nous explorons la carte de compilation proposée par Darwiche et Marquis dans [37]. Enfin, nous rappelons les principales approches de compilation dédiées à l'inférence à partir de bases de croyances stratifiées.

5.2 Principe de la compilation de connaissances

Le principe de la compilation de connaissances se résume comme suit :

- Dans un premier temps, on effectue un prétraitement sur la base de connaissances qui est une partie que l'on qualifie généralement de fixe, ce qui produit une structure de données appropriée. Cette phase est dite raisonnement **hors ligne**.
- Ensuite, on répond aux différentes requêtes en utilisant le résultat de la phase hors ligne. Cette deuxième phase est dite raisonnement **en ligne**.

Donc, l'idée de fond est de déplacer la surcharge de travail de la phase en ligne vers la phase hors ligne de façon à alléger les traitements nécessaires pour obtenir la réponse à une requête. Étant donné que la phase hors ligne n'est appelée qu'une seule fois, son coût sera amorti par les multiples phases en ligne (qui, elles, se trouvent allégées). Un peu comme si une bibliothèque fermait quelques jours pour ranger ses livres de façon à pouvoir, à l'issue du rangement, trouver plus vite les livres demandés. Illustrons ceci par l'exemple suivant.

Exemple 23 *Étant donnée une base de connaissances Σ sous forme CNF construite à partir d'un ensemble de variables V, considérons le problème qui consiste à déterminer si $\Sigma \models l$ où l est un littéral construit également à partir de V.*

Ce problème est coNP-complet. Générons maintenant une table qui indique pour chaque littéral l construit à partir de V si $\Sigma \models l$ ou non. Le calcul d'une telle table revient à résoudre un nombre polynomial de problèmes coNP-complets. Cependant, ce calcul est effectué une fois pour toute hors ligne. De plus, la taille de la table résultante est polynomiale par rapport à la taille de V. Ainsi, la résolution en ligne du problème $\Sigma \models l$ peut s'effectuer de manière polynomiale en utilisant cette table.

La notion de problème compilable introduite par [24] est donnée comme suit :

Définition 93 *Étant donné un problème défini par le triplet (P, F, V) où P désigne la requête à laquelle on veut répondre, F sa partie fixe et V sa partie variable. Ce problème est dit* ***compilable*** *si et seulement s'il existe deux polynômes $p1$, $p2$ et un algorithme ASK tel que pour toute instance f de F, il existe une structure de données D_f vérifiant :*

1. $|D_f| \leq p_1 * |f|$,
2. *Pour toute instance v de V, l'appel $ASK(D_f, v)$ retourne "OUI" si et seulement si (f, v) est une instance positive de P,*
3. $ASK(D_f, v)$ *requiert un temps* $\leq p_2 * (|v| + |Df|)$.

Autrement dit :
- la structure de données D_f engendrée par la phase hors ligne doit occuper un espace polynomial par rapport à la taille de la partie fixe,
- l'algorithme de la phase en ligne ASK doit être correct et complet,
- ASK doit de surcroît avoir un temps d'exécution qui soit polynomial par rapport à la taille de ses entrées.

L'explication du premier point est très simple. Sans cette restriction, il serait physiquement difficile de stocker la base compilée sur une machine. Le second point stipule que toutes les requêtes qui ont une réponse positive dans le problème initial, doivent aussi avoir une réponse positive dans le problème compilé (raisonnement correct) et seules les réponses qui ont une réponse positive dans le problème initial ont une réponse positive dans le problème compilé (raisonnement complet). Enfin, le dernier point permet de s'assurer que la version compilée du problème est bien dans une classe de complexité inférieure à celle du problème initial. Si seul le second point n'est pas respecté (dans une certaine mesure) alors on se retrouve plutôt dans le cadre d'une compilation approximative.

Exemple 24 *Selon cette définition, le problème de l'exemple 23 est compilable.*

Il à noter que si le nombre des requêtes possibles est d'une taille polynomiale par rapport à la taille de la partie fixe alors le problème en question est compilable. Il s'agit juste d'une condition suffisante et non pas nécessaire. En effet, il existe des problèmes compilables qui ne vérifient pas cette condition.

Exemple 25 *Étant donnés une base de connaissances Σ sous forme CNF et un terme δ construits à partir d'un ensemble de variables V, considérons le problème qui consiste à déterminer si $\Sigma \models \delta$ qui est coNP-complet.*

Bien que le nombre de termes possibles soit exponentiel, la table de l'exemple 23 est suffisante pour avoir un raisonnement en ligne polynomial. En effet, pour tout terme $\delta = \bigwedge_{i=1}^{m} l_i$, on a $\Sigma \models \delta$ si et seulement si $\Sigma \models l_i$ pour $1 \leq i \leq m$.

En revanche, on ignore jusqu'à présent si le problème consistant à déterminer si $\Sigma \models \gamma$ où Σ est une base de croyances sous forme CNF et γ une clause, est compilable ou non. En fait, il a été démontré dans [24] que si ce problème est compilable alors $\Sigma_2^p = \Pi_2^p$ (ce qui est conjecturé faux). En outre, il est facilement démontrable que si le problème consistant à déterminer si $\Sigma \models \psi$ où ψ est une formule propositionnelle quelconque est compilable alors P = NP.

Les méthodes de compilation peuvent être classifiées en deux grandes familles. La première famille comprend les méthodes de compilation qui préservent l'équivalence, c'est-à-dire celles qui génèrent un résultat équivalent à la base initiale. Ces méthodes sont dites méthodes de compilation exactes. La seconde famille regroupe les méthodes de compilation approximatives qui produisent un résultat qui n'est pas équivalent à la base initiale et donc ne permettent d'effectuer qu'une partie des déductions pouvant être faites depuis la base initiale.

5.3 Méthodes classiques de compilation exacte

Parmi les méthodes classiques dédiées à la compilation exacte, on peut citer :

- la compilation basée impliquants premiers ou impliqués premiers,
- la compilation par complétude de la résolution unitaire,
- la compilation basée impliqués premiers modulo une théorie (une base) traitable.

Ces trois méthodes de compilation seront succinctement décrites dans ce qui suit.

5.3.1 Compilation basée impliquants premiers ou impliqués premiers

Rappelons qu'un impliquant d'une base de connaissance Σ est un terme fondamental (conjonction de littéraux qui ne contient pas de littéraux complémentaires) δ tel que $\delta \models \Sigma$.

Chaque modèle de la base de connaissances correspond à un impliquant dans lequel chaque variable apparaît comme un littéral positif si elle est affectée la valeur $Vrai$, et comme littéral négatif si elle est affectée la valeur $Faux$ dans le modèle. De plus δ est dit impliquant premier de Σ si pour tout autre impliquant δ', on a $Lit(\delta') \not\subset Lit(\delta)$.

D'autre part, un impliqué de cette base est une clause fondamentale γ (disjonction de littéraux qui ne contient pas de littéraux complémentaires) telle que $\Sigma \models \gamma$. En outre, γ est dit impliqué premier si pour tout autre impliqué γ', on a $Lit(\gamma') \not\subset Lit(\gamma)$.

La disjonction de tous les impliquants premiers $\delta_1, \ldots, \delta_k$ d'une base de connaissances Σ génère une formule DNF $IP(\Sigma) = \gamma_1 \vee \ldots \gamma_k$ qui est équivalente à Σ et tel que pour toute requête R, $\Sigma \models R$ si et seulement si pour tout impliquant premier δ_i, $\delta_i \models R$. Si R est une formule CNF alors ceci revient à vérifier que chaque clause de R a une intersection non vide avec chaque impliquant premier de Σ. Par conséquent, la déduction des requêtes sous forme CNF s'effectue en temps polynomial par rapport à la taille de la formule $IP(\Sigma)$ des impliquants premiers plus la taille de la requête.

En revanche, la conjonction de tous les impliqués premiers $\gamma_1, \ldots, \gamma_l$ d'une base de connaissances Σ engendre une formule CNF $PI(\Sigma) = \gamma_1 \wedge \ldots \wedge \gamma_l$ qui est équivalente à Σ. En plus, pour toute requête R sous forme CNF, $\Sigma \models R$ si et seulement si pour chaque clause fondamentale γ' de R il existe un impliqué premier γ tel que $\gamma \models \gamma'$. Par conséquent, la déduction des requêtes sous forme CNF s'effectue en temps polynomial par rapport à la taille de la formule $PI(\Sigma)$ des impliqués premiers plus la taille de la requête. Notons que la méthode d'impliqués premiers a été largement utilisée en compilation de connaissances.

Exemple 26 *Soit* $\Sigma = \{\neg a \vee \neg b, \neg b \vee c\}$ *une base de connaissances. Alors* $PI(\Sigma) = (\neg a \vee b) \wedge (\neg b \vee c) \wedge (\neg a \vee c)$.

Intuitivement, les impliqués premiers d'une base de connaissances sous forme CNF peuvent être générés en résolvant les clauses de la base de connaissances (chaque résolvante est un impliqué) ensuite en éliminant ceux qui ne sont pas premiers. Néanmoins, cette méthode requiert un nombre prohibitif de résolutions. Le problème de calcul des impliqués premiers d'une formule CNF est apparu il y'a longtemps et il a fait l'objet de nombreuses études [94, 93, 40].

D'après [29], le nombre d'impliquants premiers et d'impliqués premiers d'une base de connaissances ayant n variables est exponentiel en n. Dans [88], les auteurs mènent une étude expérimentale concernant le nombre d'impliqués et d'impliqués premiers en fonction du rapport :

$$\frac{nombre\ de\ clauses}{nombre\ de\ variables}$$

qui a été étudié dans de nombreuses expérimentations relatives au problème de satisfiabilité. Alors, pour une base de connaissances dont ce rapport est autour d'une valeur

critique de 4.3, le nombre d'impliqués premiers est exponentiel par rapport à sa taille. Donc, de telles bases de connaissances peuvent être considérées comme étant des benchmarks intéressants afin de tester les méthodes de compilation de connaissances basées impliqués premiers.

La plupart des méthodes dédiées à la génération des impliqués premiers peuvent être adaptées à la génération des impliquants premiers. Par ailleurs, dans [88], on propose une méthode spécifique à la génération d'impliquants premiers basée sur une variante de la procédure Davis et Putnam destinée à la résolution du problème SAT : si la formule est satisfiable alors elle retourne une assignation de vérité aux différents atomes satisfaisant la formule considérée. Comme la variante en question est exhaustive dans le sens où elle explore tout l'espace de recherche et ne s'arrête pas dès que la formule est satisfaite pour la première fois, elle produit tous les impliquants à partir desquels les impliquants premiers seront sélectionnés.

5.3.2 Compilation par complétude de la résolution unitaire

Étant donné que le nombre d'impliqués premiers est en général exponentiel, del Val propose dans [43] une nouvelle méthode de compilation basée impliqués premiers qui s'avère plus efficace. Rappelons que déterminer si une requête CNF est déductible à partir d'une base de connaissances compilée sous forme de conjonction d'impliqués premiers revient à vérifier pour chaque clause de la requête, s'il existe un impliqué premier dont elle est conséquence logique, c'est-à-dire un impliqué premier qu'elle inclut (si une clause est vue comme étant un ensemble de littéraux). D'autre part, ce test d'inclusion peut être réalisé via une réfutation par résolution unitaire qui est une forme de résolution où une des deux clauses résolvantes est une clause unitaire. La résolution unitaire est saine mais incomplète dans le sens où il existe des réfutations qu'elle n'arrive pas à déceler. Nier une clause dans la requête produit un ensemble de littéraux, et via le résolution unitaire, on obtient la clause vide si et seulement s'il existe un impliqué premier qui est un sous ensemble de l'ensemble des littéraux de la clause niée.

Toutefois, la résolution unitaire peut faire plus qu'un simple test d'inclusion. En effet, grâce à la résolution unitaire, on n'a pas besoin de garder tous les impliqués premiers, mais seulement un sous ensemble de ces derniers à partir duquel tout autre impliqué peut être calculé par résolution unitaire.

Alors, del Val [43] suggère d'utiliser n'importe quel algorithme qui génère des impliqués premiers, ensuite éliminer tous les impliqués premiers qui peuvent être reconstruits par résolution unitaire durant la phase en ligne. Rappelons que la plupart des algorithmes qui calculent les impliqués premiers d'une base de connaissances opèrent en effectuant des résolutions répétitives. En effet, la méthode de del Val garde uniquement les résolvantes dite fusion.

Définition 94 *Une résolvante de deux clauses γ_1 et γ_2 est dite* ***résolvante fusion*** *si après avoir effectué une résolution sur un littéral l, la résolvante contient deux littéraux identiques, c'est-à-dire $(Lit(\gamma_1) \setminus \{l\}) \cap (Lit(\gamma_2) \setminus \{\neg l\}) \neq \emptyset$.*

Afin de voir l'utilité de garder les résolvantes fusion, on donne l'exemple suivant :

Exemple 27 *Étant donnée une base de connaissances Σ contenant les deux clauses :*
- *$\gamma_1 = a \vee b \vee \neg l$,*
- *$\gamma_2 = b \vee c \vee l$.*

La résolvante de γ_1 et γ_2 en l est la clause $\gamma_3 = a \vee b \vee c$.

Maintenant, si l'on considère les clauses unitaires $\neg a$ et $\neg c$, à partir des clauses γ_1, γ_2 et γ_3, on peut dériver b par résolution unitaire. En revanche, en éliminant γ_3, la résolution unitaire génère les deux clauses $b \vee \neg l$ et $b \vee l$ qui ne peuvent pas être résolues via la résolution unitaire et donc b n'est pas dérivable. Par conséquent, la clause γ_3 qu'est une résolvante fusion n'est pas redondante par rapport à la résolution unitaire et doit être gardée dans le processus de compilation.

del Val donne des cas où la compilation par complétude de la résolution unitaire peut éliminer un nombre exponentiel d'impliqués premiers. En outre, il montre que l'adaptation de l'algorithme de Tison [94] dédié au calcul des impliqués premiers réduit considérablement le nombre d'impliqués premiers non redondants et donc réduit la taille de la base compilée.

5.3.3 Compilation basée impliqués premiers modulo une théorie

Dans [74], Marquis propose une méthode de compilation basée impliqués premiers plus raffinée. L'idée derrière est que les méthodes basées impliqués premiers (resp. impliquants) premiers transforment le problème de l'inférence d'une formule R à partir d'une base de connaissances Σ, qui implique une base de connaissances entière, en des tests locaux impliquant un seul impliqué (resp. impliquant) premier à la fois. Il propose alors d'améliorer ces tests locaux en introduisant la notion de compilation modulo une théorie.

La compilation basée impliqués premiers modulo une théorie se base sur le principe bien connu de "diviser pour mieux régner". En effet, partant du constat qu'une base de connaissances Σ peut souvent être scindée en deux parties distinctes : une partie traitable Φ et une 'partie difficile' Ψ telles que $\Sigma \equiv \Phi \cup \Psi$, Marquis propose de restreindre le calcul des impliqués de Σ au calcul des impliqués de Σ modulo Φ. L'idée est d'utiliser autant que possible la partie facile dans le traitement de la partie difficile.

Les impliqués modulo une théorie et les impliqués premiers modulo une théorie sont donnés par les définitions suivantes :

Définition 95 *Soit Φ une théorie. Un **impliqué modulo** Φ d'une base de connaissances Σ est une clause γ telle que $\Sigma \cup \Phi \models \gamma$, noté $\Sigma \models_\Phi \gamma$.*

Définition 96 *Étant donnée une théorie Φ, un **impliqué premier modulo** Φ d'une base de connaissances Σ est un impliqué γ modulo Φ de Σ tel que pour tout autre impliqué γ' modulo Φ de Σ, on a si $\gamma' \models_\Phi \gamma$ alors $\gamma \models_\Phi \gamma'$.*

Remarquons que si l'on considère la théorie vide, on retrouve alors les définitions usuelles d'impliqués et d'impliqués premiers.

D'autre part, étant donnée une théorie Φ, pour toute requête R sous forme CNF, $\Sigma \models R$ si et seulement pour chaque clause γ' de R, il existe γ : un impliqué premier modulo Φ de Σ tel que $\gamma \models_\Phi \gamma'$. Encore une fois, en considérant la théorie vide, on retrouve la propriété des impliqués premiers.

Maintenant, vérifier $\gamma \models_\Phi \gamma'$ est équivalent à vérifier que pour chaque littéral $l_i \in Lit(\gamma)$, si $\Phi \models \neg l_i \vee \gamma'$. L'idée est que si la déduction à partir de Φ se fait en temps polynomial alors la déduction des requêtes CNF se fait aussi en temps polynomial par rapport à la taille de l'ensemble des impliqués premiers modulo une théorie. Marquis suggère de prendre comme théorie l'ensemble de toutes les clauses de Horn de Σ, de toutes ses clauses binaires et en général n'importe quel sous-ensemble de Σ qui permet une déduction polynomiale.

5.4 Méthodes classiques de compilation approximative

Nous avons vu précédemment que certains problèmes sont compilables, tandis que pour d'autres, on ignore jusqu'à présent s'ils le sont ou non. La compilation approximative s'inscrit dans ce dernier contexte. La notion de l'approximation réside dans le fait qu'il faut changer quelque chose afin de pouvoir traiter le problème aux dépens de la l'équivalence logique entre la base initiale et la base compilée.

Plus précisément, l'idée centrale de la compilation approximative est que les réponses aux requêtes en utilisant la base de connaissances compilée peuvent soit être non saines, soit incomplètes (mais pas les deux en même temps). Plus formellement, les approximations saines et les approximations complètes sont définies comme suit :

Définition 97 *Étant donnée une base de connaissances Σ.*

1. *Une approximation A de Σ est dite **saine** si pour toute requête R, si $A \models R$ alors $\Sigma \models R$. Dans ce cas, A est une borne supérieure (ou UB pour Upper Bound) de Σ.*
2. *Une approximation B de Σ est dite **complète** si pour toute requête R, si $B \not\models R$ alors $\Sigma \not\models R$. Dans ce cas, B est une borne inférieure (ou LB pour Lower Bound) de Σ.*

Remarquons que A est une UB de Σ si et seulement si $\Sigma \models A$ et que B est une LB de Σ si et seulement si $B \models \Sigma$.

Versions anytime des méthodes exactes

Toutes les méthodes de compilation exacte décrites précédemment peuvent être arrêtées prématurément à n'importe quel moment en fournissant une approximation de la base de connaissances. Plus précisément, les méthodes basées impliquants génèrent une LB : si une requête R n'est pas déductible à partir d'un des impliquants déjà calculés d'une base Σ, alors R n'est pas impliquée par Σ. D'autre part, les méthodes basées impliqués génèrent une UB : si l'un des impliqués déjà calculés implique la requête R, alors R est aussi impliquée par Σ.

Approximations de Horn

Dans [89], Selman et Kautz proposent une méthode originale de compilation approximative. L'idée de base de cette méthode est de compiler une base de connaissances vers une formule appartenant à une classe syntaxique qui garantit une inférence en temps polynomial. Par exemple, une base Σ peut être compilée en une formule de Horn (un ensemble de clauses de Horn) Σ'.

Il existe deux moyens différents pour effectuer une telle compilation. Dans le premier cas, Σ' définit une borne inférieure de Σ ($\Sigma' \models \Sigma$), et donc est appelée LB de Horn de Σ. Dans le second cas, Σ' définit une borne supérieure de Σ ($\Sigma \models \Sigma'$), et est appelée alors UB de Horn de Σ.

Selon Selman et Kautz, certaines LB's (resp. UP's) sont meilleures que d'autres en étant plus proches de la base initiale d'où les notions de GLB pour Greates Lower Bound et LUB pour Least Upper Bound données par la définition suivante :

Définition 98 *Soit Σ une base de connaissances.*
- *Une **GLB** (Greatest Lower Bound) de Σ, notée Σ_{GLB} est définie comme étant une LB de Σ telle que pour toute autre LB Σ' de Σ, si $\Sigma_{GLB} \models \Sigma'$ alors $\Sigma' \models \Sigma_{GLB}$.*
- *Une **LUB** (Least Upper Bound) de Σ, notée Σ_{LUB} est définie comme étant une UB de Σ telle que pour toute autre UB Σ' de Σ, si $\Sigma' \models \Sigma_{LUB}$ alors $\Sigma_{LUB} \models \Sigma'$.*

Exemple 28 *Soit Σ une base de connaissances telle que $\Sigma = \{\neg a \vee b, \neg c \vee b, a \vee c\}$.*

Chacune des formules $\Sigma_{LB1} = a \wedge b \wedge c$ et $\Sigma_{LB2} = a \wedge b$ sont des LB's de Horn de Σ. Cependant, Σ_{LB2} une approximation meilleure que Σ_{LB1} étant donné que $\Sigma_{LB1} \models \Sigma_{LB2} \models \Sigma$. De plus, dans ce cas, Σ_{LB2} est une GLB de Horn de Σ.

Par ailleurs, les deux formules $\Sigma_{UB1} = (\neg a \vee b) \wedge (\neg c \vee b)$ et $\Sigma_{UB2} = b$ sont des UBs

de Σ. *En outre, on a* $\Sigma \models \Sigma_{LB2} \models \Sigma_{LB21}$ *si bien que l'approximation* Σ_{UB2} *soit meilleure que* Σ_{UB1}. *En outre,* Σ_{UB2} *est une LUB de* Σ.

5.5 La carte de compilation

Les méthodes classiques de compilation de connaissances (entre autres celles que nous avons esquissées précédemment) ont pour langage cibles de compilation des variantes de CNF ou de DNF. En outre, ces méthodes considèrent souvent un seul type de requêtes à savoir la déduction clausale, c'est-à-dire vérifier si une clause est conséquence logique d'une base de connaissances ou non.

Dans [37], Darwiche et Marquis introduisent la notion d'une carte de compilation où ils présentent un spectre plus large de langages cibles de compilation de connaissances issus de travaux en Intelligence Artificielle, en vérification formelle et en Informatique d'une manière générale. De plus, ils les analysent par rapport à trois facteurs clés à savoir leur concision, la classe des requêtes auxquelles ils permettent de répondre en temps polynomial ainsi que la classe des transformations qu'ils permettent d'appliquer en temps polynomial.

Ainsi, étant donnée une application, Darwiche et Marquis proposent de choisir le langage le plus adéquat en identifiant en premier lieu l'ensemble des requêtes et des transformations auxquelles l'application fait appel, ensuite opter pour le langage le plus concis qui supporte ces opérations en temps polynomial.

5.5.1 Langages dérivés de NNF

Tous les langages cibles de compilation considérés dans la carte de Darwiche et Marquis dérivent d'un même langage dit NNF. Le langage NNF est défini comme suit :

Définition 99 *Une formule propositionnelle est sous forme* ***NNF*** (*pour Negation Normal Form*) *si elle est construite à partir de littéraux,* $\bot$ *et* $\top$ *en utilisant uniquement les opérateurs de conjonction* $\wedge$ *et de disjonction* $\wedge$.

Les graphes dirigés acycliques ou DAG (pour Directed Acyclic Graphs) constituent une bonne représentation graphique des formules NNF où chaque feuille du DAG est étiquetée par $\bot$, $\top$ ou un littéral et chaque noeud interne est étiqueté par une conjonction $\wedge$ ou par une disjonction $\vee$.

Exemple 29 *La figure 5.1 représente une formule NNF. Les fils de chaque nœud figurent en dessous de lui si bien qu'on n'ait pas besoin de montrer l'orientation des arcs du DAG.*

Toute formule propositionnelle peut être représentée par une formule NNF donc le langage NNF est complet. Il importe de faire la différence entre un langage de représentation

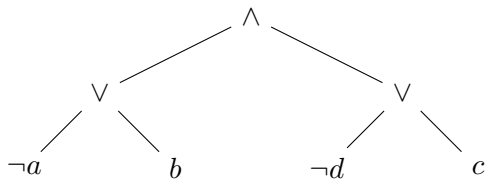

FIG. 5.1 – Exemple formule NNF

de connaissances et un langage cible de compilation de connaissances. Alors que le premier doit être assez naturel afin de permettre de coder des connaissances, le second quant à lui, doit permettre de raisonner en temps polynomial.

Définition 100 *Un langage L est dit **langage cible de compilation** s'il permet au moins la déduction clausale en temps polynomial.*

Le langage NNF n'est pas qualifié de langage cible de compilation (à moins que P = NP) [80]. Cependant, un bon nombre de ses sous ensembles le sont. Les sous ensembles du langage NNF sont définis en imposant certaines restrictions sur ce dernier.

Parmi les langages dérivés de NNF, on a les fameux langages CNF et DNF et leurs successeurs PI et IP. D'autres langages découlent de NNF. Ces derniers sont définis via les propriétés de décomposabilité, déterminisme, régularité, décision et ordre.

La propriété de décomposabilité, introduite dans [34], est définie comme suit :

Définition 101 *Une formule NNF satisfait la propriété de **décomposabilité** si et seulement si pour toute conjonction $\alpha \wedge \beta$ de cette formule, α et β ne partagent pas de variables : $Var(\alpha) \wedge Var(\beta) = \emptyset$.*

Les propriétés de déterminisme et de régularité proposées dans [35] sont données par les deux définitions suivantes :

Définition 102 *Une formule NNF satisfait la propriété de **déterminisme** si et seulement si pour toute disjonction $\alpha \vee \beta$ de cette formule, α est incohérente avec β : $\alpha \wedge \beta \models \bot$.*

Définition 103 *Une formule NNF satisfait la propriété de **régularité** si et seulement si pour toute disjonction $\alpha \vee \beta$ de cette formule, α et β partagent les mêmes variables : $Var(\alpha) = Var(\beta)$.*

En utilisant ces trois propriétés, les langages DNNF, d-NNF, s-NNF,d-DNNF et sd-DNNF sont définis comme suit :

Définition 104 – *Le langage DNNF est le sous-ensemble de NNF satisfaisant la décomposabilité.*
– *Le langage d-NNF est le sous-ensemble de NNF satisfaisant le déterminisme.*
– *Le langage s-NNF est le sous-ensemble de NNF satisfaisant la régularité.*
– *Le langage d-DNNF le sous ensemble de NNF satisfaisant la décomposabilité et le déterminisme.*
– *Le langage sd-DNNF est le sous-ensemble de NNF satisfaisant la décomposabilité, le déterminisme et la régularité.*

Rappelons maintenant la propriété de décision introduite dans [22].

Définition 105 *Un noeud de décision N dans une formule NNF est un noeud étiqueté par $\top$, $\bot$ ou un noeud de disjonction la forme $(x \wedge \alpha) \vee (\neg x \vee \beta)$ où x est une variable propositionnelle et α et β sont des noeuds de décision. $dVar(N)$ dénote la variable x.*

Définition 106 *Le langage BDD (Binary Decision Diagrams) est le sous-ensemble de NNF tel que la racine de toute formule est un noeud de décision.*

Le langage BDD correspond aux diagrammes de décision binaire (BDDs pour Binary Decision Diagrams) issus de la littérature de la vérification formelle.

Il est clair que toute formule NNF satisfaisant la propriété de décision est une formule déterministe. Donc BDD est un sous ensemble de d-NNF. D'autre part, BDD n'est pas qualifié de langage cible de compilation (à moins que P=NP), mais ses sous ensembles FBDD, OBDD et $\text{OBDD}_<$ le sont.

Définition 107 *Le langage **FBDD** (pour Free Binary Decision Diagrams) est donné par l'intersection des langages DNNF et BDD.*

Le langage FBDD correspond aux diagrammes de décision binaire libres issus aussi de la vérification formelle [55]. Un FBDD est souvent défini comme étant un BDD vérifiant la propriété "lire une seule fois" (read-once property) : sur chaque chemin entre la racine et une feuille, une variable apparaît au plus une fois.

Le langage $\text{OBDD}_<$ est obtenu à partir de FBDD en imposant la propriété d'ordre.

Définition 108 *Soit $<$ une relation d'ordre total sur l'ensemble des variables propositionnelles considérées. Le langage $OBDD_<$ est le sous ensemble de FBDD vérifiant la propriété suivante : si N et M sont des noeuds de disjonction, et si N est un prédécesseur de M alors $dVar(N) < dVar(M)$.*

Quant au langage OBDD, il correspond aux fameux diagrammes de décision binaire ordonnés (OBDDs pour Ordered Binary Decision Diagrams) [22].

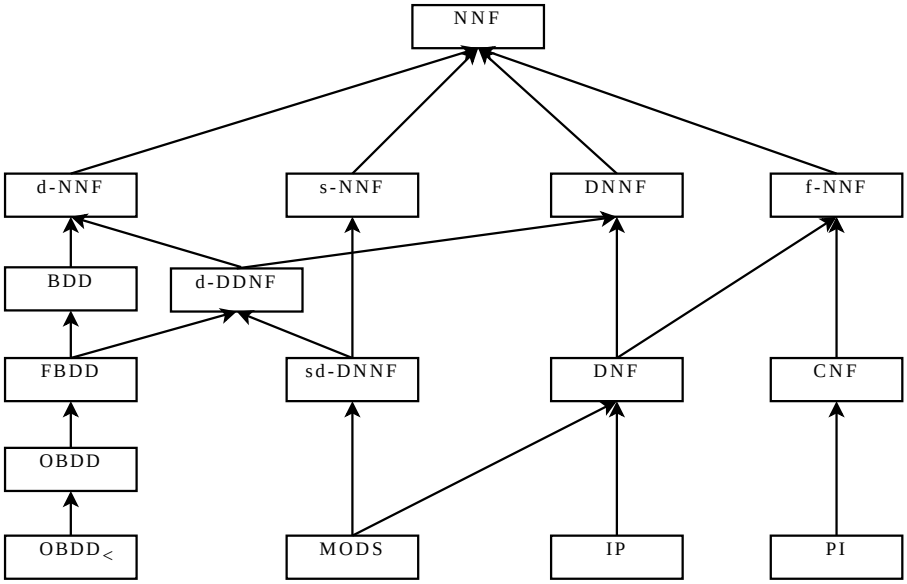

FIG. 5.2 – Relation d'inclusion entre les langages cibles de compilation

Définition 109 *Le langage OBDD est l'union de tous les langages OBDD$_<$.*

Autrement dit, le langage OBDD est le sous ensemble du langage FBDD où toute formule satisfait la propriété d'ordre : les variables apparaissent dans le même ordre quelque soit le chemin considéré entre la racine et une feuille. Étant donné un ordre particulier $<$, le langage OBDD$_<$ est le sous ensemble du langage OBDD où toutes les formules respectent le même ordre $<$. Enfin, le langage MODS se définit par :

Définition 110 *Le langage **MODS** (pour Models) est le sous-ensemble de DNF satisfaisant les deux propriétés de déterminisme et de régularité.*

Remarquons qu'à partir de la syntaxe d'une formule MODS, on peut immédiatement déduire ses modèles. La figure 5.5.1, reprise de [37] récapitule la relation d'inclusion qui existe entre les différents langages décrits dans cette section.

5.5.2 Concision des langages dérivés de NNF

La concision d'un langage représente son efficacité spatiale. Une définition formelle de ce critère est donnée par [56] comme suit :

Définition 111 *Soient L_1 et L_2 deux sous ensembles du langage NNF. Alors, L_1 est dit au moins aussi concis que L_2, noté $L_1 \leq L_2$, si et seulement si pour toute formule $\alpha \in L_2$,*

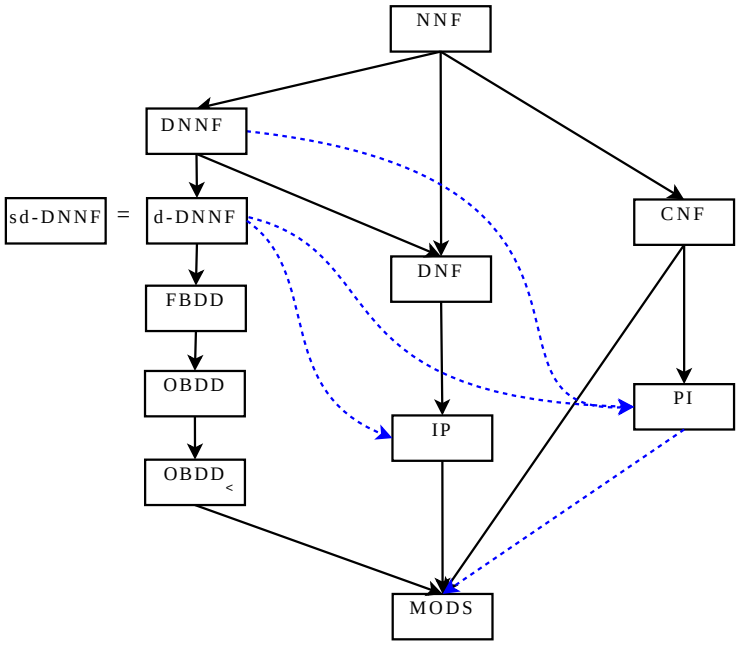

FIG. 5.3 – Relation de concision entre les langages cibles de compilation

il existe une formule $\beta \in L_1$ qui lui est équivalente telle que la taille de β est polynomiale par rapport à la taille de α.

La relation $\leq$ est un préordre et l'ordre strict associé $<$ est tel que $L_1 < L_2$ si et seulement si on a $L_1 \leq L_2$ et pas $L_2 \leq L_1$.

Darwiche et Marquis ont étudié dans leur carte de compilation les différents langages dérivés de NNF en termes de concision. Les résultats de cette étude sont récapitulés par la figure 5.5.2 où une flèche $L_1 \rightarrow L_2$ indique que L_1 est strictement plus concis que L_2 ($L_1 < L_2$) alors qu'une flèche discontinue indique une relation inconnue.

Notons qu'à l'exception du langage PI, le langage DNNF s'avère le plus concis par rapport aux autres langages cibles de compilation. En fait, PI n'est pas plus concis que DNNF mais on ignore l'inverse.

Entre les langages DNNF et MODS, on a cette relation de concision : DNNF $<$ d-DNNF $<$ FBDD $<$ OBDD $<$ OBDD$_<$ $<$ MODS. DNNF est obtenu en imposant la propriété de décomposabilité sur le langage NNF, d-DNNF en ajoutant le déterminisme, FBDD en ajoutant la décision et OBDD et OBDD$_<$ en ajoutant l'ordre (n'importe quel ordre dans le premier cas et un ordre particulier dans le second). Ajouter chacune de ces propriétés réduit la concision du langage en question. En revanche, ajouter la régularité au langage d-DNNF ne change pas sa concision : les deux langages d-DNNF et sd-DNNF sont au

même pieds d'égalité quant à leur concision.

5.5.3 Requêtes logiques et langages dérivés de NNF

Dans leur carte de compilation, Darwiche et Marquis considèrent un ensemble de requêtes logiques en plus de l'inférence où chaque requête retourne une information quant à la base de connaissances en question, et identifient pour chaque langage dérivé de NNF l'ensemble des requêtes qu'il satisfait. Rappelons tout d'abord les définitions formelles de ces différentes requêtes.

Définition 112 *Étant donné un langage cible de compilation L.*
- *Le langage L satisfait **CO** si et seulement s'il existe un algorithme polynomial permettant de déterminer pour toute formule φ de L si elle est cohérente.*
- *Le langage L satisfait **VA** si et seulement s'il existe un algorithme polynomial permettant de déterminer pour toute formule φ de L si elle est valide.*
- *Le langage L satisfait **CE** si et seulement s'il existe un algorithme polynomial permettant de déterminer pour toute formule φ de L et pour toute clause γ si $\varphi \models \gamma$.*
- *Le langage L satisfait **EQ** si et seulement s'il existe un algorithme polynomial permettant de déterminer pour toutes formules φ et ψ de L si $\varphi \equiv \psi$.*
- *Le langage L satisfait **SE** si et seulement s'il existe un algorithme polynomial permettant de déterminer pour toutes formules φ et ψ de L si $\varphi \models \psi$.*
- *Le langage L satisfait **IM** si et seulement s'il existe un algorithme polynomial qui permet de déterminer pour toute formule φ de L et pour tout terme δ si $\delta \models \varphi$.*
- *Le langage L satisfait **CT** si et seulement s'il existe un algorithme polynomial qui permet de déterminer pour toute formule φ de L le nombre de ses modèles.*
- *L satisfait **ME** si et seulement s'il existe un polynôme $p(.,.)$ et un algorithme qui donne pour toute formule φ de L tous ses modèles en temps $p(n, m)$, où n est la taille de la formule φ et m le nombre de ses modèles.*

Pour chaque langage dérivé de NNF, la table 5.1 donne les différentes requêtes qu'il satisfait.

Remarquons que chacun des langages NNF, s-NNF, d-NNF, f-NNF et BDD appartient à la même classe qui ne permet de répondre à aucune requête en temps polynomial, CNF satisfait uniquement VA et IM si bien qu'aucun de ces langages ne soit qualifié de langage cible de compilation. Sinon, tous les autres langages satisfont CO et CE.

Par ailleurs, DNNF satisfait CO, CE et ME. En ajoutant le déterminisme à la décomposabilité (d-DNNF), on satisfait de plus VA, IM et CT qui sont assez importants. Cependant, on ignore toujours si d-DNNF satisfait EQ ou non. D'autre part, ajouter la décision à la décomposabilité et au déterminisme (FBDD), n'augmente pas l'efficacité, par contre, ça réduit la concision. En outre, ajouter la propriété d'ordre à la décomposabilité,

L	CO	VA	CE	IM	EQ	SE	CT	ME
NNF	○	○	○	○	○	○	○	○
DNNF	√	○	√	○	○	○	○	√
d-NNF	○	○	○	○	○	○	○	○
s-NNF	○	○	○	○	○	○	○	○
f-NNF	○	○	○	○	○	○	○	○
d-DNNF	√	√	√	√	?	○	√	√
sd-DNNF	√	√	√	√	?	○	√	√
BDD	○	○	○	○	○	○	○	○
FBDD	√	√	√	√	?	○	√	√
OBDD	√	√	√	√	√	○	√	√
$\text{OBDD}_<$	√	√	√	√	√	√	√	√
DNF	√	○	√	○	○	○	○	√
CNF	○	√	○	√	○	○	○	○
PI	√	√	√	√	√	√	○	√
IP	√	√	√	√	√	√	○	√
MODS	√	√	√	√	√	√	√	√

TAB. 5.1 – Langages dérivés de NNF et requêtes logiques

au déterminisme et à la décision permet de satisfaire SE (pour OBDD et $\text{OBDD}_<$) et EQ (pour $\text{OBDD}_<$). Aussi, remarquons que DNNF est plus concis que DNF tout en supportant le même ensemble de requêtes en temps polynomial. D'autre part, l'ajout de la régularité et du déterminisme à DNF, ce qui donne MODS, assure cinq requêtes supplémentaires entre autres SE et EQ. Enfin, on voit que le déterminisme est nécessaire (mais pas suffisant) pour CT : seuls les langages déterministes (d-DNNF, sd-DNNF, FBDD, $\text{OBDD}_<$, OBDD et MODS) satisfont CT. De plus, CT implique VA mais l'inverse n'est pas vrai.

5.5.4 Transformations logiques et langages dérivés de NNF

Une requête est une opération qui retourne une information quant à la base de connaissances sans pour autant la modifier. Une transformation, par contre, est une opération qui retourne une base modifiée. De nombreuses applications requièrent une combinaison de transformations et de requêtes. Darwiche et Marquis prennent en compte dans leur carte de compialtion un bon nombre de transformations telles que $\wedge C$, $\vee C$, $\neg C$, $\wedge BC$, $\vee BC$, le conditionnement et l'oubli de variables.

Définition 113 *Soit L un langage dérivé de NNF.*

- *L satisfait $\wedge C$ si et seulement s'il existe un algorithme polynomial qui associe à tout ensemble fini de formules $\{\varphi_1, \ldots, \varphi_n\}$ de L une formule de L qui soit logiquement équivalente à $\varphi_1 \wedge \ldots \wedge \varphi_n$.*
- *L satisfait $\vee C$ si et seulement s'il existe un algorithme polynomial qui associe à tout ensemble fini de formules $\{\varphi_1, \ldots, \varphi_n\}$ de L une formule de L qui soit logiquement équivalente à $\varphi_1 \vee \ldots \vee \varphi_n$.*
- *L satisfait $\neg C$ si et seulement s'il existe un algorithme polynomial qui associe à toute formule φ de L une formule de L qui soit logiquement équivalente à $\neg\varphi$.*

Certains langages sont clos par rapport à un opérateur logique si seulement le nombre d'opérandes est borné par une constante d'où la définition suivante :

Définition 114 *Soit L un langage dérivé de NNF.*
- *L satisfait $\wedge BC$ si et seulement s'il existe un algorithme polynomial qui associe à toute paire φ_1 et φ_2 de L une formule de L qui soit logiquement équivalente à $\varphi_1 \wedge \varphi_2$.*
- *L satisfait $\vee BC$ si et seulement s'il existe un algorithme polynomial qui associe à toute paire φ_1 et φ_2 de L une formule de L qui soit logiquement équivalente à $\varphi_1 \vee \varphi_2$.*

Le conditionnement est une transformation ayant de nombreuses applications, et il correspond à la restriction dans la littérature des fonctions Booléennes. Formellement, il est donné par la définition suivante :

Définition 115 *Soit φ une formule propositionnelle et soit δ un terme cohérent. Le conditionnement de φ sur δ, noté par $\varphi|\delta$, est la formule obtenue en remplaçant toute occurrence d'une variable x dans φ par $\top$ (resp. $\bot$) si le littéral correspondant apparaît positif dans δ (resp. négatif).*

Exemple 30 *Soit $\psi = (\neg a \wedge \neg b) \vee (b \wedge c)$ une formule propositionnelle.*
- $\psi|(\neg a \wedge c) \equiv (\top \wedge \neg b) \vee (b \wedge \top) \equiv \top$.
- $\psi|(a \wedge b) \equiv (\bot \wedge \bot) \vee (\top \wedge c) \equiv c$.

Définition 116 *Un langage dérivé de NNF L satisfait CD si et seulement s'il existe un algorithme polynomial qui associe à toute formule φ de L et à tout terme cohérent δ une formule de L équivalente à $\varphi|\delta$.*

L'application principale du conditionnement est inhérente à un théorème qui stipule que $\varphi \wedge \delta$ est cohérente si et seulement si $\varphi|\delta$ est cohérente [34]. Par conséquent, il est au cœur de la compilation DNNF comme nous le verrons plus loin.

La dernière transformation s'articule autour de l'oubli de variables [71]. Oublier les variables d'un ensemble A dans une formule φ consiste à éliminer toute référence de A dans φ tout en maintenant toute l'information que fournit φ à propos du complément de A.

L	CD	FO	SFO	∧C	∧BC	∨C	∨BC	¬C
NNF	√	○	√	√	√	√	√	√
DNNF	√	√	√	○	○	√	√	○
d-NNF	√	○	√	√	√	√	√	√
s-NNF	√	○	√	√	√	√	√	√
f-NNF	√	○	√	•	•	•	•	√
d-DNNF	√	○	○	○	○	○	○	?
sd-DNNF	√	○	○	○	○	○	○	?
BDD	√	○	√	√	√	√	√	√
FBDD	√	•	○	•	○	•	○	√
OBDD	√	•	√	•	○	•	○	√
$OBDD_<$	√	•	√	•	√	•	√	√
DNF	√	√	√	•	√	√	√	•
CNF	√	○	√	√	√	•	√	•
PI	√	√	√	•	•	•	√	•
IP	√	•	•	•	√	•	•	•
MODS	√	√	√	•	√	•	•	•

TAB. 5.2 – Langages dérivés de NNF et transformations logiques.

Définition 117 *Soit φ une formule propositionnelle, et soit A un ensemble fini de variables propositionnelle. L'oubli de A dans φ, noté $\exists A.\varphi$ (ou $ForgetVariable(\varphi, A)$) est une formule qui ne contient aucune variable de A et telle que pour toute formule Ψ qui ne mentionne aucune variable de A ($Var(\Psi) \cap A = \emptyset$), on a $\varphi \models \Psi$ si et seulement si $\exists A.\varphi \models \Psi$.*

Définition 118 *L satisfait **FO** si et seulement s'il existe un algorithme polynomial qui associe à toute formule φ de L et à tout ensemble A de variables une formule de L équivalente à $\exists A.\varphi$. Si on considère un singleton A, on dit que L satisfait SFO.*

L'oubli est également une transformation importante étant donné qu'elle permet de projeter une base de connaissances sur un ensemble de variables. Par exemple, si on sait d'emblée que certaines variables de A ne vont pas apparaître dans les requêtes, on peut alors oublier ces variables dans la base compilée tout en préservant sa capacité de répondre correctement à ces requêtes. En outre, ça a de nombreuses applications en planification.

La table 5.2 identifie pour chaque langage dérivé de NNF, la liste de transformations qu'ils satisfait.

On constate que tous les langages satisfont CD. Quant à FO, seuls DNNF, DNF, PI et MODS le satisfont. De plus, aucun des langages FBDD, OBDD et OBDD< ne satisfait

FO. En revanche, OBDD et OBDD$_<$ satisfont SFO contrairement à FBDD.

On remarque aussi qu'aucun des langages cibles de compilation ne satisfait la conjonction. Par contre, certains d'entre eux sont clos par rapport à la conjonction bornée à savoir les langages OBDD$_<$, DNF, IP et MODS. Quant à la disjonction, les seuls langages cibles de compilation qui sont clos par rapport à elle sont DNNF et DNF. Les langages OBDD et PI satisfont la disjonction bornée. Les seuls langages cibles de compilation clos par rapport à la négation sont FBDD, OBDD et OBDD<.

Enfin, OBDD$_<$ est l'unique langage cible de compilation clos par rapport à la négation, la conjonction bornée et la disjonction bornée. Cette fermeture joue un rôle important dans la compilation des bases de connaissances propositionnelles vers le langage OBDD$_<$ et est la base de l'état de l'art des compilateurs dédiés à cette fin [22].

5.5.5 Compilateur DNNF

La première investigation du langage DNNF était faite par Darwiche dans le contexte du diagnostic basé modèles [33]. Ensuite, Darwiche a constaté que son utilité est indépendante de cette application spécifique. En effet, DNNF peut être vu comme un langage cible de compilation intéressant. L'ensemble d'opérations (requêtes et transformations) logiques qu'il supporte en temps polynomial (CO, CE, ME, CD, FO, SFO, $\vee$C, $\vee$BC) est relativement suffisant pour réaliser de nombreuses applications complexes en Intelligence Artificielle par exemple en diagnostic et en planification. En outre, hormis le langage PI, le langage DNNF est le plus concis de tous les langages cibles de compilation de connaissances (on sait que PI n'est pas plus concis que DNNF mais on ignore l'inverse).

Dans [34], Darwiche propose un compilateur DNNF dont l'idée s'articule autour de la propriété suivante :

Soient φ_1 et φ_2 deux formules DNNF, et soit $X = Var(\varphi_1) \cap Var(\varphi 2)$. Soit φ la formule donnée par $\bigvee_\delta((\varphi_1|\delta) \wedge (\varphi_2|\delta) \wedge \delta)$ où δ est un terme tel que $Var(\delta) = X$ (δ est construit sur X). Alors, Darwiche a démontré que φ est une formule DNNF équivalente à $\varphi_1 \wedge \varphi_2$.

Illustrons cette propriété via l'exemple suivant.

Exemple 31 *Soit $\varphi = (\neg a \vee b) \wedge (\neg b \vee c)$. Alors, selon la propriété précédente, la formule $((\neg a \vee b)|b \wedge (\neg b \vee c)|b \wedge b) \vee ((\neg a \vee b)|\neg b \wedge (\neg b \vee c)|\neg b) \wedge \neg b)$ est une formule DNNF équivalente à φ.*

Étant donnée une formule CNF φ, l'idée du compilateur est de considérer φ comme une conjonction de deux sous-formules φ_1 et φ_2, ensuite appliquer la propriété précédente ce qui donne $\bigvee_\delta((\varphi_1|\delta) \wedge (\varphi_2|\delta) \wedge \delta)$. Pour chaque δ, si $(\varphi_1|\delta)$ est une clause alors on ne fait rien, sinon on refait le même traitement qu'on a effectué pour φ et ainsi de suite. Il est de même pour $(\varphi_2|\delta)$. Afin de choisir δ_1 et δ_2, Darwiche propose l'utilisation d'un arbre de décomposition.

Définition 119 *Un arbre de décomposition T d'une forme CNF φ est un arbre binaire dont les feuilles correspondent aux clauses de φ. Si t est la feuille de T qui correspond à une clause α dans φ alors $\varphi(t) = \{\alpha\}$.*

Pour chaque nœud interne t, on a :
- t_l dénote son fils gauche et t_r son fils droit,
- $\varphi(t) = \varphi(t_l) \cup \varphi(t_r)$,
- $Var(t) = Var(\varphi(t))$,
- $Var^{\uparrow}(t)$ dénote l'ensemble des variables associées aux feuilles qui ne sont pas dans le sous-arbre de racine t.

Exemple 32 *La figure 5.4 décrit un arbre de décomposition de la formule CNF $\varphi = (\neg a \vee b) \wedge (\neg b \vee c) \wedge (\neg c \vee d)$. Dans ce cas, on a par exemple :*
- $\varphi(t_1) = \{\neg a \vee b, \neg b \vee c\}$,
- $Var(t_1) = \{a, b, c\}$,
- $Var^{\uparrow}(t_1) = \{c, d\}$.

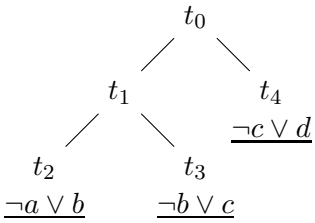

FIG. 5.4 – Arbre de décomposition

Soit un arbre de décomposition T (de racine t_0) d'une formule CNF φ. La compilation de φ vers le langage DNNF se fait par l'appel $dnnf(t_0, \top)$ où $dnnf$ est la procédure décrite par l'algorithme 5.1.

Illustrons cette procédure à travers l'exemple suivant :

Exemple 33 *Considérons la formule $\varphi = (\neg a \vee b) \wedge (b \vee c) \wedge (\neg c \vee d)$ pouvant avoir l'arbre décrit par la figure 5.5 comme arbre de décomposition.*

Afin de compiler φ vers le langage DNNF, il suffit de calculer $dnnf(t_0, \top)$. On obtient alors :

$$dnnf(t_0, \top) = (c \wedge dnnf(t_1, c) \wedge dnnf(t_4, c)) \vee (\neg c \wedge dnnf(t_1, \neg c) \wedge dnnf(t_4, \neg c)).$$

Comme t_4 correspond à une feuille et $\varphi(t_4) = \neg c \vee d$, on déduit que $dnnf(t_4, c) = (\neg c \vee d)|c \equiv d$ et $dnnf(t_4, \neg c) = (\neg c \vee d)|\neg c \equiv \top$.

Algorithme 5.1: $dnnf(t, \alpha)$

Données: un noeud t dans un arbre de décomposition
un terme α
début
  si t *est une feuille et* $\varphi(t) = \{\gamma\}$ **alors**
    $\psi \leftarrow \gamma|\alpha$
  sinon
    $\psi \leftarrow \bigvee_{\delta}(dnnf(t_l|\delta \wedge \alpha) \wedge dnnf(t_r|\delta \wedge \alpha) \wedge \delta)$ où $Var(\delta) = (Var(t_l) \cap Var(t_r)) \setminus Var(\alpha)$.
  retourner ψ
fin

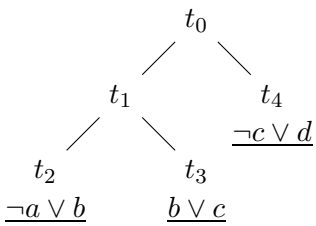

FIG. 5.5 – Arbre de décomposition

Par conséquent,

$$dnnf(t_0, \top) = (c \wedge d \wedge dnnf(t_1, c)) \vee (\neg c \wedge dnnf(t_1, \neg c)).$$

Calculons maintenant $dnnf(t_1, c)$*. En fait, on a :*

$$dnnf(t_1, c) = (b \wedge dnnf(t_2, c \wedge b) \wedge dnnf(t_3, c \wedge b)) \vee (\neg b \wedge dnnf(t_2, c \wedge \neg b) \wedge dnnf(t_3, c \wedge \neg b)).$$

Après simplification, on obtient $dnnf(t_1, c) = b \vee (\neg b \wedge \neg a)$.
De manière similaire, on obtient $dnnf(t_1, \neg c) = b$.
Enfin, on conclut que :

$$dnnf(t_0, \top) = (c \wedge d \wedge (b \vee (\neg b \wedge \neg a))) \vee (\neg c \wedge b).$$

Donc, cette formule est une compilation de la formule φ *vers DNNF :* $DNNF(\varphi)$*. Remarquons que pour toute conjonction dans* $DNNF(\varphi)$*, les fils ne partagent pas de variables.*

La complexité de la procédure $dnnf$ est fonction de la largeur de l'arbre de décomposition considéré, et donc rappelons d'abord la définition de cette largeur.

Définition 120 *Soit t un nœud dans un arbre de décomposition T.*

*Le **cluster** du nœud t est défini comme suit :*
- *si t est une feuille, alors $cluster(t) = Var(t)$,*
- *si t est un nœud interne, alors $cluster(t) = (Var(t) \cap Var^{\uparrow}(t)) \cup (Var(t_l) \cap Var(t_r))$.*

*La **largeur** d'un arbre de décomposition est la taille de son maximum cluster moins 1.*

Soit φ une formule CNF ayant n clauses. et soit T un arbre de décomposition de φ de largeur w. Alors, Darwiche montre que la complexité de la procédure $dnnf$ est en $O(nw2^w)$.

Par conséquent, la largeur de l'arbre de décomposition utilisé est un facteur critique pour l'efficacité du compilateur. La construction de 'bons' arbres de décompositions, c'est-à-dire d'une largeur qui n'est pas importante a fait l'objet de plusieurs travaux. Par exemple, dans [34], Darwiche utilise une méthode basée sur un ordre d'élimination de variables. Une autre méthode qui s'appuie sur la décomposition d'un hyper graphe a été proposée dans [36].

5.5.6 Le lien entre SAT et la compilation de connaissances

Dans [61], Huang et Darwiche exhibent un lien étroit entre la résolution du problème SAT et la compilation de connaissances en logique propositionnelle. En effet, la plupart des algorithmes complets pour SAT (hormis ceux basés sur la résolution) ne sont que des variantes de la procédure de Davis et Putnam (DP) [39]. La trace d'une version exhaustive de cette procédure mémorisée d'une façon compacte sur un DAG peut constituer une compilation de la formule en question. En imposant ou en éliminant certaines contraintes, la formule compilée appartiendra à un des langages suivants : d-DNNF, FBDD ou OBDD.

Rappelons en premier lieu la procédure DP dans une version qui se veut la plus simple possible.

En fait, dans la procédure DP, les variables sont assignées variable par variable. Le choix de la prochaine variable à affecter se fait selon une heuristique de branchement.

Compilateur FBDD

La procédure DP effectue une recherche dans l'espace des assignations et s'arrête lorsqu'elle trouve une assignation qui satisfait la formule CNF ou réalise qu'une telle assignation n'existe pas. La version exhaustive de la procédure DP ne s'arrête pas sur la première instanciation trouvée. En fait, elle énumère toutes les assignations satisfaisant

Algorithme 5.2: $DP(\varphi)$

Données: une formule propositionnelle φ
début
  si *il existe une clause vide dans* φ **alors**
    retourner *0*
  si *il n'y a pas de variable dans* φ **alors**
    retourner *1*
  retourner $DP(\varphi|\neg x)$ *ou* $DP(\varphi|x)$
fin

la formule en explorant dans tous les cas les deux parties de l'instruction (**retourner** $DP(\varphi|\neg x)$ ou $DP(\varphi|x)$) de la procédure DP.

La trace d'une procédure DP exhaustive appliquée à une formule propositionnelle permet d'identifier cette formule étant donné qu'elle spécifie ses modèles. Cependant, d'un point de vue compilation de connaissances, une trace de recherche mémorisée dans sa forme actuelle n'est pas immédiatement utile car sa taille est proportionnelle à l'effort fourni en vue de sa génération, et répondre à n'importe quelle requête en se basant sur cette représentation revient à refaire toute la recherche une seconde fois.

Ce problème peut être résolu en représentant la trace par un DAG au lieu d'un arbre en appliquant les deux règles de réduction suivantes :

- les nœuds ayant le même label, le même fils droit et le même fils gauche (nœuds isomorphiques) doivent être confondus.
- tout nœud ayant deux fils identiques est éliminé et les arêtes qui se dirigeaient vers lui seront dirigés vers un de ses fils.

D'autre part, des portions du DAG peuvent être explorées plus d'une fois. La solution réside dans l'utilisation d'un cache. Ainsi, le résultat d'un appel récursif $fbdd(\delta)$ est mémorisé dans ce cache avant d'être retourné en étant indexé par une clef identifiant φ.

Huang et Darwiche décrivent plus formellement ce processus via l'algorithme 5.3. Dans ce dernier, la fonction *getNode* retourne un nœud de décision étiqueté par le premier argument, son fils gauche correspond au second argument et son fils droit correspond au troisième argument. Cette fonction utilise une technique bien connue dans la littérature BDD qu'est la technique de la table des nœuds uniques permettant de vérifier les deux règles de réduction décrites précédemment. En effet, tous les nœuds créés par *getNode* sont mémorisés dans une table et *getNode* ne crée pas de nœud redondant dans le cas où le nœud à créer existe déjà dans la table (ce nœud existant est retourné) ou dans le cas où le second et le troisième arguments sont identiques (un des deux est retourné).

Algorithme 5.3: $fbdd(\varphi)$

Données: φ : une formule propositionnelle sous forme CNF
début
  si *il existe une clause vide dans* φ **alors**
    retourner $feuille_0$
  si *il n'y a pas de variable dans* φ **alors**
    retourner $feuille_1$
  clé $\leftarrow$ *calculerClé*(φ)
  si *clé* $\neq$ *null* **alors**
    retourner *cache(clé)*
  sélectionner une variable x de φ
  résultat$\leftarrow$ $getNode(x, fbdd(\varphi|\neg x), fbdd(\varphi|x))$
  insérerCache(clé, résultat)
  retourner *résultat*
fin

Compilateur OBDD

Dans le compilateur FBDD décrit par la procédure 5.3, le choix de la prochaine variable à instancier se fait dynamiquement selon une heuristique de branchement. Huang et Darwiche proposent de substituer cet ordre de sélection dynamique par un ordre statique afin de générer une formule OBDD. Ils proposent ainsi un compilateur OBDD ayant comme second argument un ordre de sélection de variables π qui doit être respecté lors de la construction du DAG.

Il est à noter que contrairement à ce compilateur OBDD, la méthode standard [22] largement utilisée en vérification formelle pour la construction des OBDDs souffre du problème suivant : les OBDDs intermédiaires utilisés dans la procédure de construction peuvent être d'une taille colossale ce qui empêche de continuer le calcul même si la taille de l'OBDD final est raisonnable.

Compilateur d-DNNF

Le langage FBDD étant un sous ensemble du langage d-DNNF, il est possible selon Darwiche et Huang de définir un compilateur d-DNNF en omettant certaines contraintes au niveau du compilateur FBDD. Plus précisément, au lieu de se brancher directement sur une variable à instancier, on décompose d'abord la formule en question en des sous-formules disjointes. Ces sous-formules seront compilées récursivement et liées sous forme d'un nœud de conjonction via la procédure *get-and-node*. Par conséquent, c'est seulement

Algorithme 5.4: $obdd(\varphi, \pi)$

Données: φ : une formule propositionnelle sous forme CNF
π : un ordre sur les variables de φ

début
  si *il existe une clause vide dans* φ **alors**
    retourner $feuille_0$
  si *il n'y a pas de variable dans* φ **alors**
    retourner $feuille_1$
  clé $\leftarrow$ *calculerClé*(φ)
  si *clé* $\neq$ *null* **alors**
    retourner *cache(clé)*
  $x \leftarrow$ *première variable de* π *qui apparaît dans* φ
  résultat$\leftarrow getNode(x, obdd(\varphi|\neg x), obdd(\varphi|x))$
  insérerCache(clé, résultat)
  retourner *résultat*
fin

dans le cas où la formule n'est pas décomposable qu'on se branche sur une variable comme pour la procédure DP classique. Ce compilateur d-DNNF est décrit par l'algorithme 5.5.

On remarque que ce compilateur fait recours à une décomposition dynamique : les composantes disjointes sont calculées après chaque instanciation d'une variable. Il est possible et surtout moins coûteux de procéder par une décomposition statique et ce en utilisant un arbre de décomposition.

5.6 Compilation en présence d'incohérence

Les différentes approches de compilation passées en revue dans les sections précédentes se rapportent à des bases de connaissances qui sont cohérentes. En effet, appliquer ces approches sur des bases incohérentes mène à une trivialisation. Par exemple, la compilation DNNF d'une base incohérente se résume à la contradiction $\bot$. Par conséquent, d'autres approches de compilation en présence d'incohérence, notamment pour l'inférence à partir de bases de croyances stratifiées ont été proposées.

5.6.1 Compilation de bases de croyances stratifiées

Marquis et Coste-Marquis proposent dans [31] une approche de compilation pour l'inférence possibiliste simple, l'inférence linéaire, l'inférence lexicographique et l'inférence

Algorithme 5.5: $d-DNNF(\varphi)$

Données: φ : une formule propositionnelle sous forme CNF
début
  si *il existe une clause vide dans* φ **alors**
    retourner $feuille_0$
  si *il n'y a pas de variable dans* φ **alors**
    retourner $feuille_1$
  composantes $\leftarrow$ *partitions disjointes de* φ
  si $|composante| > 1$ **alors**
    $conjonctions = \emptyset$
    pour *tout* $\varphi_c \in composante$ **faire**
      $conjonctions = conjonctions \cup d-DNNF(\varphi_c)$
    retourner $get-node-and(conjonctions)$
  clé $\leftarrow$ *calculerClé*(φ)
  si *clé* $\neq$ *null* **alors**
    retourner *cache(clé)*
  sélectionner une variable x de φ
  résultat$\leftarrow getNode(x, fbdd(\varphi|\neg x), fbdd(\varphi|x))$
  insérerCache(clé, résultat)
  retourner *résultat*
fin

basée sur la préférence pour l'inclusion à partir de bases de croyances stratifiées.

Étant donnée une base de croyances stratifiées $\Delta = (S_1, \ldots, S_m)$, les auteurs supposent tout d'abord que la strate S_1 contient des connaissances certaines et donc est cohérente (si on n'a aucune connaissance certaine, S_1 contient la tautologie). Ensuite, leur compilation consiste à donner un nom sous forme d'un nouveau littéral $hold_{i,j}$ à chaque croyance $\varphi_{i,j}$ de Δ. Après, ils mémorisent cette correspondance (croyance / nom) dans Δ_1 avant de compiler Δ_1 vers un langage cible de compilation L.

Ainsi, la compilation de Δ par rapport à L est la nouvelle base de croyances stratifiées $(X_1, \ldots, X_m)$ telle que :

- $X_i = \{holds_{i,1}, \ldots, holds_{i,|S_i|}\}$ avec $2 \leq i \leq m$,
- $X_1 = \{La\ compilation\ de\ (S_1 \cup \bigcup_{i=2}^{m} \{\bigwedge_{j=1}^{|S_i|} (holds_{i,j} \Rightarrow \varphi_{i,j})\})\ vers\ L\}$

Par ailleurs, les auteurs montrent que cette compilation est efficace, c'est-à-dire permet d'effectuer l'inférence à partir de bases de croyances stratifiées en temps polynomial, par

rapport au langage DNF. En revanche, elle n'est pas efficace par rapport au langage des impliqués premier qui tient une position importante en compilation de connaissances.

Une approche de compilation similaire, dont s'inspire d'ailleurs la compilation de Marquis et Coste-Marquis, est proposée dans [27]. Dans ce cas, le but d'effectuer l'inférence lexicographique à partir de bases de croyances stratifiées en temps polynomial. Cette approche s'appuie aussi sur un codage propositionnel de la base de croyances stratifiées en question en introduisant pour chaque formule de la base une nouvelle variable propositionnelle. Le langage cible de compilation considéré dans cette approche est le langage OBDD.

Enfin, notons que l'approche de compilation proposée par Marquis et Coste-Marquis a été étendue dans [38] par Darwiche et Marquis pour traiter l'inférence en logique de pénalités tout en permettant de tirer profit du langage DNNF.

5.6.2 Compilateur possibiliste DNF

Dans [15], Benferhat et Prade introduisent un nouveau compilateur de bases de croyances possibilistes afin de traiter en temps polynomial l'inférence possibiliste simple et l'inférence possibiliste pondérée.

Ce compilateur s'inspire de la méthode présentée dans [14] dans le cadre du traitement d'une extension de la logique possibiliste où on définit les degrés de certitude des formules de manière symbolique via l'utilisation des opérateurs maximum et minimum. Les idées sous-jacentes consistent à coder la base possibiliste à compiler $\Delta = (S_1, \ldots, S_m)$ sous la forme d'une base propositionnelle K_Δ, à utiliser les notions d'oubli de variables et de littéraux et surtout le langage DNF.

La première étape du codage propositionnel de la base possibiliste consiste à associer à chaque strate S_i $(1 \leq i \leq m)$ une nouvelle variable propositionnelle A_i. Soit S l'ensemble des variables propositionnelles introduites. Ensuite, à chaque formule φ de la strate S_i, est associée la formule $\varphi \vee A_i$. En outre, dire qu'une strate S_i est plus prioritaire qu'une strate S_j est codé par la formule propositionnelle $\neg A_i \vee A_j$.

Ainsi, le codage propositionnel associé à Σ est donné par :

$$K_\Delta = \{\varphi \vee A_i \ : \ \varphi \in S_i\} \cup \{\neg A_i \vee A_{i+1}, 1 \leq i \leq n-1\}.$$

Par ailleurs, ce compilateur utilise les opérations d'oubli de variables donnée par la définition 117 ainsi que l'opération d'oubli de littéraux. L'oubli de littéraux une généralisation de l'oubli de variables [66].

Soit φ une formule propositionnelle et soit l un littéral. Alors, l'oubli de l dans K, noté $ForgetLiteral(L, l)$, est défini comme suit :

$$ForgetLiteral(K, l) = K_{l \leftarrow \top} \vee \neg K.$$

Ensuite, les auteurs montrent comment calculer de degré d'incohérence d'une base possibiliste, et ce en utilisant son codage propositionnel et les opérations d'oubli de variables et de littéraux. En effet, ils prouvent que Δ est cohérente si et seulement si la formule $ForgetLiteral(ForgetVariable(K_\Delta, V), L_S^-)$ est une tautologie où V est l'ensemble des variables initial et L_S^- est l'ensemble des littéraux négatifs construits à partir de S. De plus, cette base est incohérente à un degré i si et seulement si on a cette équivalence $ForgetLiteral(ForgetVariable(K_\Delta, V), L_S^-) \equiv A_i \wedge \ldots \wedge A_m$.

Une fois, le degré d'incohérence calculé, les auteurs génèrent $Comp_S(\Delta)$ dénotant la compilation de cette base qui permet d'effectuer l'inférence possibiliste simple en utilisant l'inférence classique de la manière suivante :

- Si Δ est cohérente alors $Comp_S(\Delta) = ForgetVariable(\neg A_1, \ldots, \neg A_n \wedge DNF(K_\Delta), S)$.
- Si Δ est incohérente à un degré i alors $Comp_S(\Delta) = ForgetVariable(\neg A_1, \ldots, \neg A_{i-1} \wedge DNF(K_\Delta), S)$.

En ce qui concerne la base compilée relative à l'inférence possibiliste pondérée, elle correspond exactement à $DNF(K_\Delta)$, c'est-à-dire la compilation de K_Δ vers DNF. Maintenant, $\Delta \vdash_\pi \varphi$ si et seulement si :

- $ForgetVariable(\neg A_1 \wedge \ldots \wedge \neg A_i \wedge A_{i+1} \wedge \ldots \wedge A_n \wedge DNF(K_\Delta) \models \varphi$,
- $ForgetVariable(\neg A_1 \wedge \ldots \wedge \neg A_j \wedge A_{j+1} \wedge \ldots \wedge A_n \wedge DNF(K_\Delta) \not\models \varphi, \forall j < i$.

Ainsi, pour effectuer l'inférence possibiliste pondérée, Benferhat et Prade proposent l'algorithme 5.6.

Algorithme 5.6:

Données: $DNF(K_\Delta)$
une formule φ
début
  $k \leftarrow 0$
  $X \leftarrow \{A_1, \ldots, A_m\}$
  $\delta \rightarrow \bigwedge_{A_i \in X} A_i$
  tant que $ForgetVariable(\delta \wedge DNF(K_\Delta), S) \not\models \varphi$ **faire**
    $k \leftarrow k + 1$
    $X \leftarrow X \setminus \{A_k\}$
    $X \leftarrow X \cup \{\neg A_k\}$
    $\delta \rightarrow \bigwedge_{A_i \in X} A_i$
  retourner A_k
fin

Remarquons que le coût de ce compilateur est plus faible que celui des approches

précédentes. En effet, cette méthode rajoute une variable par strate et non pas une variable par formule. Néanmoins, il se trouve efficace uniquement par rapport au langage DNF.

5.7 Conclusion

La compilation de connaissances est une approche prometteuse permettant de pallier les problèmes calculatoires liés au raisonnement logique. Nous avons présenté dans ce chapitre l'idée derrière la compilation de connaissances. Aussi, nous avons passé en revue les approches classiques de compilation de connaissances, notamment celles basées sur les impliqués premiers. Un autre point d'une grande importance en compilation et que nous avons abordé dans ce chapitre est celui de la carte de compilation proposée par Darwiche et Marquis dans [37]. Étant donnée une application, cette carte permet à l'utilisateur de choisir le langage le plus concis qui répond aux besoins de son application relativement aux opérations logiques qu'il permet d'effectuer en temps polynomial. Parmi les langages de compilation proposés considérés dans cette carte, DNNF et PI tiennent une position importante. En effet, les opérations qu'ils supportent sont suffisantes pour de nombreuses applications en Intelligence Artificielle. De plus, hormis PI, DNNF est plus concis que tous les autres langages cibles de compilation. Par ailleurs, PI n'est pas plus concis que DNNF mais on ignore si l'inverse est vérifié. Notons que cette carte a été étendue dernièrement par Marquis et Fargier [48], et ce en prenant en compte d'autres langages cibles de compilation qui ne sont pas complets pour la logique propositionnelle tels que les fragments de Horn, Krom, etc. Nous avons décrit également le lien entre SAT et la compilation de connaissances. Enfin, nous avons présenté le compilateur possibiliste DNF proposée par Benferhat et Prade dans [15], ainsi que d'autres approches de compilation à partir de bases de croyances stratifiées, notamment celle de Marquis et Coste-Marquis [31].

Deuxième partie

Contributions

Chapitre 6

Compilation des inférences possibiliste et linéaire

6.1 Introduction

La logique possibiliste constitue un outil intéressant pour le raisonnement en présence d'incohérence. Cette logique trouve de plus en plus d'applications dans de nombreux domaines pratiques tels que la modélisation de politiques de sécurité informatique. Une politique de sécurité exprime formellement les règles de contrôle d'accès permettant de garantir la confidentialité, l'intégrité et la disponibilité d'un système. Néanmoins, les règles de contrôle d'accès présentent souvent des situations exceptionnelles ce qui génère des conflits. Par exemple, un conflit apparaît lorsque l'on autorise un utilisateur à effectuer une action et en même temps on lui interdit de l'exécuter dans une situation exceptionnelle. La logique possibiliste s'est avérée commode pour exprimer et traiter un tel problème. En effet, dans [7], les auteurs proposent une modélisation des politiques de sécurité dans le cadre de la logique possibiliste.

Néanmoins, d'un point de vue calculatoire, le raisonnement en logique possibiliste est un problème coûteux. Plus précisément, il s'agit d'un problème $\Delta_2^p[O(log\ n)]$-complet, c'est-à-dire un problème qui requiert un nombre d'appels logarithmique à un oracle NP. Clairement, ce dernier point est limitatif sur le plan pratique, notamment pour des problèmes où le temps de réponse est un facteur important tels que le contrôle d'accès, surtout si on considère des contextes délicats comme le domaine de la santé par exemple.

Parmi les approches prometteuses qui permettent de pallier les problèmes calculatoires liés au raisonnement logique, la compilation occupe une position importante. Afin de compiler une base de croyances stratifiées par rapport à l'inférence possibiliste simple par exemple, une approche directe consiste à générer dans un premier temps la sous-base cohérente associée. Notons que cette étape nécessite un nombre d'appels logarithmique à un oracle NP. Ensuite, on compile la base classique résultante vers un langage cible de compilation. Néanmoins, cette approche souffre de nombreux problèmes. D'abord, elle est inappropriée pour calculer les conséquences possibilistes pondérées ou le conditionnement possibiliste qui sont très importants en pratique. En outre, lorsque la relation de priorité entre les strates est modifiée, la base doit être intégralement recompilée, tout en sachant que le processus de compilation est coûteux alors que l'idée de la compilation est de l'effectuer une fois pour toutes. D'autres approches de compilation de bases de croyances possibilistes plus efficaces ont été proposées dans la littérature telles que celle proposée par Marquis et Coste-Marquis [31] et la compilation proposée par Benferhat et Prade [15], et que nous avons rappelées dans le chapitre 5.

Dans la même optique, nous proposons dans ce chapitre une nouvelle approche de compilation de bases de croyances stratifiées relativement à l'inférence possibiliste. Cette compialtion contourne les points faibles des approches précédentes. Entre autres, notre approche est plus flexible dans le sens où elle peut être paramétrée par n'importe quel langage cible de compilation. De plus, cette approche permet également le calcul du con-

ditionnement possibiliste et peut être efficacement exploitée dans le cas de l'inférence linéaire.

L'approche de compilation que nous présentons dans ce chapitre a fait l'objet d'une publication à [17].

6.2 Principe de compilation de notre approche

Soit $\Delta = (S_1, ..., S_k)$ une base de croyances stratifiées. La première étape de notre compilation consiste à coder cette base sous forme d'une base propositionnelle classique. Plus précisément, pour chaque strate S_i, nous introduisons une nouvelle variable propositionnelle A_i. Donc, à chaque formule φ_{ij} d'une strate S_i, correspondra la formule propositionnelle $\varphi_{ij} \vee A_i$.

Définition 121 *Soit $\Delta = (S_1, ..., S_m)$ une base de croyances stratifiées. Le codage propositionnel associé à Δ, noté K_Δ, est la base de connaissances propositionnelle définie par :*

$$K_\Delta = \bigcup_{i=1}^{m} \{\phi_{ij} \vee A_i : \phi_{ij} \in S_i\}.$$

Il est à noter que le nombre de strates dans une base stratifiées est en général faible par rapport au nombre de variables présentes dans cette même base. Par conséquent, le coût d'un tel codage reste faible aussi.

Par ailleurs, remarquons que le codage que nous introduisons ici est proche de celui proposé dans [15]. La différence principale réside dans le fait que dans notre cas, on ne rajoute pas de clauses binaires.

Maintenant, étant donné le codage de Δ, nous définissons la compilation de Δ par rapport à un langage cible de compilation L de la manière suivante :

Définition 122 *Soit $\Delta = (S_1, ..., S_m k)$ une base de croyances stratifiées et soit K_Δ son codage propositionnel selon la définition 121 et soit L un langage cible de compilation. Alors, la compilation de Δ par rapport à L, notée $CompPL(\Delta, L)$, est la compilation de la base classique K_Δ vers L :*

$$CompPL(\Delta, L) = L(K_\Delta).$$

Illustrons cette compilation à travers l'exemple suivant.

Exemple 34 *Soit $\Delta = (S_1, S_2, S_3)$ une base de croyances stratifiées telle que :*

- $S_1 = \{\neg a \vee b, a\}$,
- $S_2 = \{\neg b, c\}$,
- $S_3 = \{\neg b \vee \neg d\}$.

Soient A_1, A_2 et A_3 trois nouvelles variables propositionnelles associées respectivement aux strates S_1, S_2 et S_3. Alors, selon la définition 121, le codage propositionnel de Δ est donné par :

$$K_\Delta = \{\neg a \vee b \vee A_1, a \vee A_1, \neg b \vee A_2, c \vee A_2, \neg b \vee \neg d \vee A_3\}.$$

En choisissant par exemple le langage DNNF comme langage cible de compilation, la compilation de Δ par rapport à $DNNF$ n'est rien d'autre que la compilation de K_Δ vers DNNF :

$CompPL(\Delta, DNNF) = DNNF(K_\Delta) \equiv (\neg b \wedge A_1 \wedge (c \vee A_2)) \vee (b \wedge A_2 \wedge (a \vee A_1) \wedge (\neg d \vee A_3))$.

6.3 Inférence possibiliste simple

Dans cette section, nous montrons comment on peut tirer profit de la compilation d'une base de croyances stratifiées que nous avons proposée dans la section précédente en vue d'effectuer l'inférence possibiliste simple en temps polynomial.

Pour ce faire, nous générons à partir de $CompPL(\Delta, L)$ une nouvelle formule propositionnelle dont les conséquences logiques classiques coïncident avec les conséquences de la base stratifiées $\Delta = (S_1, \ldots, S_m)$ relativement à l'inférence possibiliste simple. Le processus de génération d'une telle formule est décrit par l'algorithme 6.1.

Proposition 1 *Soit Δ une base de croyances stratifiées et soit φ une formule propositionnelle. Étant donnée la formule K issue de l'application de l'algorithme 6.1 sur Δ, alors on a*

$$\Delta \vdash_\pi \varphi \ ssi \ K \models \varphi.$$

Preuve.

La proposition 1 montre en fait l'équivalence entre la formule générée par l'algorithme 6.1 et la sous-base cohérente qui est issue des strates qui se trouvent avant le degré d'incohérence de Δ, notée $Pos(\Delta)$. Cet algorithme procède strate par strate et génère graduellement la base propositionnelle associée. En fait, tant que $K \not\models A_i$ ce qui est équivalent à dire que la strate S_i est cohérente avec les strates précédentes, on garde les formules ϕ_{ij}'s de cette strate et ce en conditionnant K sur $\neg A_i$ ce qui revient à remplacer chaque $A_i \vee \phi_{ij}$ par $\bot \vee \phi_{ij}$ qui équivaut ϕ_{ij}. Ensuite, une fois une incohérence est détectée, la strate

Algorithme 6.1:
Données: $CompPL(\Delta, L)$: la compilation d'une base stratifiée Δ par rapport à un langage cible de compilation L
début
  $K \leftarrow CompPL(\Delta, L)$;
  $i \leftarrow 1$;
  tant que $K \not\models A_i$ ***et*** $i \leq m$ **faire**
    $K \leftarrow K|\neg A_i$;
    $i \leftarrow i + 1$;
  si $i \leq m$ **alors**
    $K \leftarrow K|\bigwedge_{k=i}^{m} A_k$;
  retourner K ;
fin

courante qui est responsable de l'incohérence S_i ainsi que toutes les strates moins prioritaires sont ignorées en conditionnant K cette fois-ci sur $\bigwedge_{k=i}^{m} A_k$ dans le but de substituer chaque $A_k \vee \phi_{kj}$ par $\top \vee \phi_{kj}$ c'est-à-dire par une tautologie.

■

Par ailleurs, l'algorithme 6.1 s'effectue en temps polynomial. De plus, la formule propositionnelle qu'il génère appartient au langage cible de compilation L, d'où l'efficacité de l'inférence possibiliste simple à partir de Δ comme le montre la proposition suivante :

Proposition 2 *Soit $CompPL(\Delta, L)$ la compilation d'une base de croyances Δ par rapport à un langage cible de compilation L et soit φ une formule propositionnelle sous forme CNF. Alors, l'inférence possibiliste simple de φ à partir de Δ s'effectue en temps polynomial en utilisant sa compilation $CompPL(\Delta, L)$.*

Preuve.

D'après la proposition 1, on a $\Delta \vdash_\varphi$ ssi $K \models \varphi$ où K est la formule propositionnelle issue de l'algorithme 6.1.

Par ailleurs, cet algorithme s'effectue en temps polynomial. En outre, la formule propositionnelle qu'il génère appartient au langage cible de compilation L.

En effet, on sait que pour tout langage cible de compilation L, L satisfait le conditionnement [37]. Donc le conditionnement de K sur un terme peut être calculé en temps polynomial et le résultat appartient toujours à L. En plus, puisque A_i est un littéral et est donc sous forme CNF, le test $K \models A_i$ s'effectue en temps polynomial.

Enfin, comme K appartient à un langage cible de compilation et φ est sous-forme CNF, l'inférence classique de φ à partir de K s'effectue en temps polynomial.

■

Illustrons ce processus à travers l'exemple suivant :

Exemple 35 *Considérons l'exemple 34. Initialement,* $K \leftarrow (\neg b \wedge A_1 \wedge (c \vee A_2)) \vee (b \wedge A_2 \wedge (a \vee A_1) \wedge (\neg d \vee A_3))$.

1. *A la première itération, on a* $K \not\models A_1$ *ce qui veut dire que la strate* S_1 *est cohérente. De ce fait,* $K \leftarrow K|\neg A_1 \equiv b \wedge A_2 \wedge a \wedge (\neg d \vee A_3)$.
2. *A la seconde itération,* $K \models A_2$ *c'est-à-dire* $S_1 \cup S_2$ *est incohérente. Ceci implique que* $K \leftarrow K|A_2 \wedge A_3 \equiv b \wedge a$.

Clairement, ce calcul s'effectue en temps polynomial car le langage DNNF satisfait l'inférence de clauses ainsi que le conditionnement. Aussi, on peut voir facilement que $K \equiv Pos(\Delta)$ *et est sous forme DNNF. Par conséquent, l'inférence possibiliste à partir de* Δ *qui équivaut l'inférence classique à partir de* K *s'effectue en temps polynomial.*

6.4 Inférence possibiliste pondérée et conditionnement possibiliste

La compilation de Δ par rapport à un langage cible de compilation L, $CompPL(\Delta, L)$, que nous avons proposée peut être de surcroît efficacement exploitée afin de calculer des conséquences possibilistes pondérées. Le processus correspondant est décrit par l'algorithme 6.2.

Dans cet algorithme, on commence par tester la cohérence de la première strate S_1 ce qui est équivalent à vérifier que $K \not\models A_1$. Si ce n'est pas le cas, on déduit que ψ n'est pas une conséquence possibiliste de Δ. Sinon, c'est-à-dire si on a la cohérence de S_1, on vérifie si $S_1 \models \psi$. Ceci peut être effectué d'abord en conditionnant K sur A_1, ensuite tester si le résultat infère ψ. Si c'est le cas, on déduit que ψ est une conséquence possibiliste de Δ à un degré i, sinon on passe à la strate suivante S_2, et on réitère le même traitement. Illustrons mieux ceci via l'exemple suivant.

Exemple 36 *Considérons encore une fois l'exemple 34, et vérifions si* b *est une conséquence possibiliste de* Δ *et à quel degré si elle l'est.*

Rappelons que $CompPL(\Delta, DNNF) \equiv (\neg b \wedge A_1 \wedge (c \vee A_2)) \vee (b \wedge A_2 \wedge (a \vee A_1) \wedge (\neg d \vee A_3))$.

Il se trouve qu'à la première itération, $K \not\models A_1$. *Donc,* $K \leftarrow K|\neg A_1 \equiv (b \wedge A_2 \wedge a \wedge (\neg d \vee A_3))$.

Algorithme 6.2: Calculer l'inférence possibiliste pondérée
Données: $CompPL(\Delta, C)$,
Une formule CNF ψ
début
  $K \leftarrow CompPL(\Delta, C)$;
  $Is\_consequence \leftarrow faux$;
  $i \leftarrow 1$;
  tant que $K \not\models A_i$ ***et*** $i \leq k$ ***et*** $\neg Is\_consequence$ **faire**
    $K \leftarrow K|\neg A_i$;
    si $K \models \psi$ **alors**
      $Is\_consequence \leftarrow vrai$;
    sinon
      $i \leftarrow i + 1$;
  si *Is_consequence* **alors**
    $\Delta \vdash_\pi \varphi$ à un degré i
  sinon
    $\Delta \nvdash_\pi \varphi$
fin

Dans ce cas, $K \models b$ *ce qui implique que* $\Delta \vdash_\pi b$ *à un degré* 1 *c'est-à-dire à partir de la première strate.*

Par ailleurs, pour vérifier si une formule ψ est conséquence du conditionnement possibiliste de Δ sur un terme γ, il suffit d'adapter l'algorithme 6.2 et ce en remplaçant la première partie de la condition de la boucle tant que $K \not\models \neg A_i$ par $K \not\models \neg\gamma \vee A_i$ et la condition du premier "Si" $K \models \neg\psi$ par $K \models \neg\gamma \vee \psi$ comme le montre l'algorithme 6.3.

Un terme Étant donnée la formule $\neg\gamma$ sous forme CNF, il est claire que l'algorithme 6.3 s'effectue en temps polynomial.

6.5 Inférence linéaire à partir de bases de croyances stratifiées

Nous proposons d'étendre l'approche de compilation que nous avons proposée dans le cadre de l'inférence possibiliste afin d'effectuer l'inférence linéaire à partir de bases de croyances stratifiées.

La compilation de la base de croyances stratifiée correspond exactement à celle donnée par la définition 122 c'est-à-dire $CompPL(\Delta, L)$ à partir de laquelle nous générons

Algorithme 6.3: Calculer le conditionnement possibiliste
Données: $CompPL(\Delta, C)$,
Une formule CNF ψ
Une formule γ
début
  $K \leftarrow CompPL(\Delta, C)$;
  $Is\_consequence \leftarrow faux$;
  $i \leftarrow 1$;
  tant que $K \not\models \neg\gamma \vee A_i$ ***et*** $i \leq k$ ***et*** $\neg Is\_consequence$ **faire**
    $K \leftarrow K|\neg A_i$;
    si $K \models \neg\gamma \vee \psi$ **alors**
      $Is\_consequence \leftarrow vrai$;
    sinon
      $i \leftarrow i + 1$;
  si *Is_consequence* **alors**
    retourner *v*rai
  sinon
    retourner *f*aux
fin

une nouvelle formule propositionnelle selon le processus décrit par l'algorithme 6.4. Cette formule est caractérisée par le fait que ses conséquences logiques classiques correspondent exactement aux conséquences de la base Δ par rapport à l'inférence linéaire.

Proposition 3 *Soit Δ une base de croyances stratifiées et soit φ une formule propositionnelle. Étant donnée la formule K issue de l'application de l'algorithme 6.4 sur Δ, alors on a*

$$\Delta \vdash_{lin} \varphi \ ssi \ K \models \varphi.$$

Preuve.

Selon cette proposition, on a l'équivalence entre la formule issue de l'algorithme 6.4 et $Lin(\Delta)$. En effet, pour chaque strate S_i, si $K \not\models A_i$ ce qui est équivalent à dire que la strate S_i est cohérente avec les strates qui lui sont plus prioritaires, on garde les formules ϕ_{ij}'s de cette strate par le conditionnement de K sur $\neg A_i$ ce qui revient à remplacer chaque $A_i \vee \phi_{ij}$ par $\bot \vee \phi_{ij}$ qui équivaut ϕ_{ij}. Autrement, cette strate S_i est ignorée via le conditionnement de K dans ce cas sur le terme A_i en vue de remplacer chaque $A_i \vee \phi_{ij}$ par $\top \vee \phi_{ij}$ c'est-à-dire par une tautologie.

■

En outre, l'algorithme 6.4 est polynomial, et la formule propositionnelle qu'il génère appartient au langage cible de compilation L. De ce fait, nous assurons l'efficacité de l'inférence linéaire à partir de Δ.

Proposition 4 *Soit $CompPL(\Delta, L)$ la compilation d'une base de croyances stratifiées Δ par rapport à un langage cible de compilation L et soit φ une formule propositionnelle sous forme CNF. Alors, l'inférence linéaire de φ à partir de Δ s'effectue en temps polynomial en utilisant sa compilation $CompPL(\Delta, L)$.*

Preuve.

La preuve de cette proposition est similaire à celle de la proposition 2. ■

Algorithme 6.4:
Données: $CompPL(\Delta, C)$
début
  $K \leftarrow CompPL(\Delta, C)$;
  pour $i \leftarrow 1$ **à** k **faire**
    si $K \models A_i$ **alors**
      $K \leftarrow K|A_i$;
    sinon
      $K \leftarrow K|\neg A_i$;
  retourner K
fin

Exemple 37 *Considérons l'exemple 34 où $CompPL(\Delta, DNNF) \equiv (\neg b \wedge A_1 \wedge (c \vee A_2)) \vee (b \wedge A_2 \wedge (a \vee A_1) \wedge (\neg d \vee A_3))$.*

- *A la première itération, on a $K \not\models A_1$ d'où $K \leftarrow K|\neg A_1 = b \wedge A_2 \wedge a \wedge (\neg d \vee A_3)$.*
- *A la seconde itération, $K \models A_2$ ce qui implique que $K \leftarrow K|A_2 \equiv b \wedge a \wedge (\neg d \vee A_3)$.*
- *A la troisième itération, on a $K \not\models A_3$. Par conséquent $K \leftarrow K|\neg A_3 \equiv b \wedge a \wedge \neg d$.*

On peut facilement vérifier que K est équivalente à $Lin(\Delta)$.

6.6 Résultats expérimentaux

Soit une base de croyances stratifiées Δ et soit L un langage cible de compilation. Étant donné que l'inférence possibiliste (simple et pondérée) ainsi que l'inférence linéaire s'effectuent en temps polynomial par rapport à la taille de la compilation de Δ par rapport à L, $CompPL(\Delta, L)$, nos expérimentations portent alors uniquement sur la taille de la base compilée.

Comme langage cible de compilation, nous avons opté pour le langage d-DNNF vu ses propriétés intéressantes. Nous avons donc utilisé le compilateur c2d développé par Darwiche [1].

Il est à noter que le domaine du raisonnement en présence d'incohérence, et contrairement à celui de la satisfiabilité d'une formule propositionnelle par exemple, manque de Benchmarks destinés à l'évaluation pratique et à la comparaison des méthodes proposées. Nous avons pallié cette limitation en utilisant certaines bases incohérentes issues de DIMACS challenge [2]. En particulier, nous avons considéré les Benchmarks (aim-50-1_6-no-1, ..., aim-50-1_6-no-4) contenant 50 variables et 80 clauses et (aim-50-2_0-no-1, ..., aim-50-2_0-no-4) ayant 50 variables. Nous avons partitionné ces bases en une suite de strates où chaque strate contient une vingtaine de clauses et ce en gardant les clauses dans l'ordre initial.

Pour chaque base stratifiée Δ_i, la table 6.1 donne la taille de la compilation correspondante, $CompPL(\Delta_i$, d-DNNF), en termes de nombre de nœuds et d'arcs. Nous donnons en plus le temps de compilation nécessaire.

BCS	Nombre de nœuds	Nombre d'arc	Temps(sec)
Δ_1	37917	87223	0
Δ_2	35589	79626	1
Δ_3	47725	105832	1
Δ_4	52949	122706	1
Δ_5	1211578	2757801	15
Δ_6	549331	1231462	6
Δ_7	364732	832655	4
Δ_8	507350	1092358	7

TAB. 6.1 – Résultats expérimentaux

A la lumière de ces résultats, on constate l'efficacité de l'approche de compilation que nous avons proposée qui est totalement cohérente avec le fait que le codage propositionnel associé soit faible.

6.7 Conlusion

Nous avons présenté dans ce chapitre une nouvelle approche de compilation de bases de croyances stratifiées relativement aux relations d'inférence possibilistes simple et pondérée,

[1] http ://reasoning.cs.ucla.edu/c2d/
[2] http ://dimacs.rutgers.edu/Challenges/

au conditionnement possibiliste ainsi qu'à l'inférence linéaire.

Par rapport aux approches existantes, notre compilation présente de nombreux avantages. Tout d'abord, comparée à l'approche proposée par Marquis et Coste-Marquis [31], notre approche n'introduit pas une nouvelle variable par formule mais plutôt une nouvelle variable par strate, ce qui est clairement plus efficace d'un point de vue calculatoire. De plus, contrairement à leur approche, la nôtre permet de traiter efficacement l'inférence possibiliste pondérée et le conditionnement possibiliste qui tiennent une grande importance en logique possibiliste.

Par ailleurs, à la différence de la compilation proposée dans [15] par Benferhat et Prade, notre approche ne rajoute pas de clauses binaires, et elle a été étendue pour traiter le conditionnement possibiliste et l'inférence linéaire.

Enfin, notre approche est plus flexible dans le sens où elle peut être paramétrée par n'importe quel langage cible de compilation, ce qui n'est pas le cas des deux autres approches. Ainsi, on peut par exemple exploiter des langages complémentaires en termes de concision, à l'instar de DNNF et PI.

Chapitre 7

Compilation de l'inférence lexicographique

7.1 Introduction

L'inférence lexicographique admet des propriétés intéressantes d'un point de vue théorique, pratique et psychologique. Par exemple, une étude psychologique réalisée dans le contexte du raisonnement par défaut a révélé que l'inférence lexicographique présente de fortes similarités avec le raisonnement humain non-monotone [8]. Par ailleurs, elle est plus productive que l'inférence possibiliste. Le productivité d'une relation d'inférence étant un critère subjectif, nous permet de dire que l'inférence lexicographique peut être préférée à l'inférence possibiliste dans certrains contextes.

Néanmoins, l'inférence lexicographique représente un problème coûteux d'un point de vue calculatoire. En effet, il s'agit d'un problème Δ_2^p-complet, c'est-à-dire un problème qui nécessite un nombre d'appels polynomial à un oracle NP. Afin de palier ce problème, nous proposons dans ce chapitre une nouvelle approche de compilation de l'inférence lexicographique en utilisant les contraintes de cardinalité Booléennes. Cette compilation permet d'effectuer l'inférence lexicographique en temps polynomial et offre la possibilité de mettre à jour la relation de priorité entre les strates sans aucun coût de recompilation. De plus, contrairement aux approches existantes, notre approche peut être efficacement paramétrée par n'importe quel langage cible de compilation. En particulier, elle permet de tirer profit du fameux langage des impliqués premiers qui tient une position importante en Intelligence Artificielle en général, et en compilation de connaissances en particulier. Avant de présenter notre approche en détail, nous rappelons dans un premier temps l'algorithme proposé dans [27] pour le calcul de l'inférence lexicographique, et sur lequel notre approche est basée. Nous donnons ensuite un bref rappel sur les contraintes de cardinalité Booléennes.

L'approche de compilation de bases de croyances stratifiées par rapport à l'inférence lexicographique présentée dans ce chapitre a fait l'objet d'une publication [99].

7.2 Rappels sur le calcul de l'inférence lexicographique

Dans [27], Cayrol, Lagasquie-Schiex et Schiex proposent de calculer l'inférence lexicographique en utilisant l'idée suivante : afin de vérifier si une base de croyances stratifiées $\Delta = (S_1, \ldots, S_m)$ $(m \geq 1)$ infère lexicographiquement une formule ψ, ils calculent d'abord le profil de Δ où le profil de Δ, noté P_Δ dans ce qui suit, qui est un vecteur de m éléments tel que chaque élément $P_\Delta[i]$ $(1 \leq i \leq m)$ correspond au nombre de formules de la strate S_i qui sont satisfaites par chaque sous-base cohérente lexicographiquement préférées, étant donné que toutes les sous-bases lexicographiquement préférées ont le même nombre de formules au niveau de chaque strate.

Pour ce faire, les auteurs introduisent un oracle NP-complet appelé Max-GSat-Array défini comme suit :

Définition 123 MAX-GSAT-ARRAY *est défini par :*
- ***Instance :*** *un couple* (Δ, P) *où* Δ *une base de croyances stratifiées et* P *est un vecteur ayant une taille égale au nombre de strates de* Δ.
- ***Question :*** *existe-t-il une interprétation qui satisfait au moins* $P[i]$ *formules pour chaque strate* S_i $(1 \leq i \leq m)$ *dans* Δ *?*

Ensuite, une fois le profil P_Δ calculé, les auteurs démontrent que $\Delta \vdash_{lex} \psi$ si et seulement s'il n'existe pas une interprétation qui satisfait au moins $P_\Delta[i]$ formules de chaque strate S_i $(1 \leq i \leq m)$ et ne satisfait pas la formule ψ. Ainsi, ils définissent un autre oracle NP-complet noté NGSAT-LEX de la manière suivante :

Définition 124 NGSAT-LEX *est défini par :*
- ***Instance :*** *un tuple* (Δ, P, ψ) *où* Δ *une base de croyances stratifiées et* P *est un vecteur ayant une taille égale au nombre de strates de* Δ.
- ***Question :*** *existe-t-il une interprétation qui satisfait au moins* $P_\Delta[i]$ *formules pour chaque strate* S_i $(1 \leq i \leq m)$ *dans* Δ *et ne satisfait pas* ψ *?*

L'algorithme 7.1 décrit plus formellement ce processus. Rappelons que les auteurs ont prouvé que le problème de décision associé à l'inférence lexicographique est Δ_2^p-complet. En effet, l'algorithme 7.1 est déterministe polynomial et utilise un oracle NP d'où l'appartenance à la classe Δ_2^p.

7.3 Contraintes de cardinalité Booléennes

Les contraintes de cardinalité Booléennes imposent des bornes inférieure et supérieure sur le nombre de variables assignées à *Vrai* dans un ensemble de variables propositionnelles.

Une contrainte de cardinalité Booléenne, notée $\#(\mu, \lambda, E)$, où μ et λ sont des entiers positifs et E est un ensemble de variables propositionnelles, est satisfaite si et seulement si au moins μ et au plus λ des variables de E peuvent être assignées à vrai. Le codage CNF correspondant est donné par la définition suivante :

Définition 125 *Étant donnée une contrainte de cardinalité Booléenne* $\#(\mu, \lambda, E)$*, son codage CNF est une formule CNF, notée* $\Psi_{\#(\mu,\lambda,E)}$*, construite sur un ensemble de variables qui inclut* E *telle que* $\Psi_{\#(\mu,\lambda,E)}$ *est satisfaite par une interprétation si et seulement si les valeurs assignées aux variables de* E *par cette interprétation satisfont la contrainte de cardinalité Booléenne.*

Un bon nombre de travaux ont été dédiés au codage CNF des contraintes de cardinalité Booléennes tels que [97, 4, 91]. Parmi ces travaux, nous nous sommes focalisés sur le codage

Algorithme 7.1: Inférence lexicographique

Données: Une base de croyances stratifiées $\Delta = (S_1, \ldots, S_m)$ avec $m \geq 1$
Une formule propositionnelle ψ
Result : Est-ce que $\Delta \vdash_{lex} \psi$?
début
  $P_\Delta \leftarrow < 0, 0, \ldots, 0 >$
  pour $i \leftarrow 1$ *to* m **faire**
    $k \leftarrow |S_i|$
    $Stop \leftarrow faux$
    tant que $k \geq 0$ *et* $\neg Stop$ **faire**
      $P_\Delta[i] \leftarrow k$
      si MAX-GSAT-ARRAY(Δ, P_Δ) **alors**
        $Stop \leftarrow vrai$
      sinon
        $k \leftarrow k - 1$
  si NGSAT-LEX(Δ, P_Δ, ψ) **alors**
    retourner *faux*
  sinon
    retourner *vrai*
fin

UT-MGAC (pour Unit Totalizer Maintaining Generalized Arc Consistency) proposé dans [4] pour de nombreuses raisons. Tout d'abord, ce codage est satisfaisant d'un point de vue spatial. En effet, il requiert $O(|E|log([E]))$ variables et $O(|E|^2)$ clauses additionnelles d'une taille de 3 au maximum. Aussi, ce codage est efficace dans le sens où la propagation unitaire qui est au cœur de la plupart des solveurs SAT et des compilateurs de base de connaissances, restaure la cohérence arc généralisée sur les variables de E. Ce dernier point signifie que si une contrainte est violée par une assignation partielle, la propagation unitaire génère la clause vide. De plus, ce codage est intéressant d'un point de vue compilation étant donné que les bornes μ et λ peuvent être modifiées sans aucune recompilation comme nous le verrons en détail plus loin.

La caractéristique clé de ce codage est la représentation unaire des entiers : un entier v tel que $0 \leq v \leq n$ est représenté par un ensemble $V = \{v_1, \ldots, v_n\}$ de n variables propositionnelles. Si la valeur de v est x alors $v_1 = \ldots = v_x = Vrai$ et $v_{x+1} = \ldots = v_n = Faux$.

L'avantage principal d'une telle représentation est qu'elle permet de spécifier qu'une variable entière appartient à un intervalle donné. En effet, $x \leq v \leq y$ est représentée par

l'instantiation partielle de V qui fixe à $Vrai$ tout v_i tel que $i \leq x$ et fixe à $Faux$ tout v_j tel que $j \geq y + 1$.

Un additionneur basé sur cette représentation peut être utilisé afin de déduire l'intervalle auquel appartient un entier $c = a + b$ étant donnés les intervalles de a et b. Donc, une contrainte de cardinalité $\#(\mu, \lambda, E)$ peut être codée par un additionneur structuré comme un réseau pyramidal d'additionseurs $\mathcal{T}(E)$ étendu par un comparateur $\mathcal{C}(\mu, \lambda)$ qui restreint les valeurs de sortie possible.

L'additionneur $\mathcal{T}(E)$ est une formule CNF définie sur 3 ensembles de variables :

- $E = \{e_1, \ldots, e_n\}$: l'ensemble des variables d'entrée,
- $S = \{s_1, \ldots, s_n\}$: l'ensemble des variables de sortie,
- L : l'ensemble des variables de liens.

Par conséquent, un additionneur ayant comme entrée $a_1, \ldots, a_{m_1}, b_1, \ldots, b_{m_2}$ et comme sortie $r_1, \ldots, r_m$ est codé par

$\bigwedge_{i,j,k}(\neg a_i \vee \neg b_j \vee r_k) \wedge (a_{i+1} \vee b_{j+1} \vee \neg r_{k+1})$ pour $0 \leq i \leq m_1$, $0 \leq j \leq m_2$, $0 \leq k \leq m$ et $k = i + j$ en utilisant la notation $a_0 = b_0 = r_0 = \top$ et $a_{m_1+1} = b_{m_2+1} = r_{m+1} = \bot$.

Quant au comparateur $\mathcal{C}(\mu, \lambda)$, il est donné par le terme $\mathcal{C}(\mu, \lambda) \equiv \bigwedge_{1 \leq i \leq \mu}(s_i) \wedge \bigwedge_{\lambda+1 \leq j \leq n}(\neg s_j)$.

7.4 Nouvelle approche de compilation pour l'inférence lexicographique

Clairement, selon l'algorithme 7.1, une compilation possible pour l'inférence lexicographique consiste à calculer le profil correspondant une fois pour toutes. Alors, le problème est réduit de la classe Δ_2^p-complet à la classe NP-complet. Néanmoins, le problème reste coûteux. De plus, le calcul du profil demande un nombre polynomial d'appel à un oracle NP. Ce dernier point est surtout critique lorsque les priorités entre les strates changent car le profil doit être recalculé à chaque fois.

La compilation que nous proposons permet d'effectuer l'inférence lexicographique en temps polynomial. De plus, elle est flexible dans le sens où le profil peut être calculé en temps polynomial. Ainsi, la mise à jour des priorités entre les strates ne demande aucune recompilation.

La première étape consiste à traduire les oracles utilisés dans l'algorithme 7.1 en un problème de satisfiabilité classique, et ce en utilisant des contraintes de cardinalité Booléennes.

Dans ce qui suit, nous aurons besoin d'utiliser des contraintes de cardinalité Booléennes associées à des formules propositionnelles si bien qu'on commence par donner la définition suivante :

Définition 126 *Une formule $\mathcal{BCC}$-$\mathcal{FRM}$ est définie par la donnée du couple $(\psi, \mathcal{C})$ où ψ une formule propositionnelle et $\mathcal{C}$ est un ensemble de contraintes de cardinalité Booléennes. Alors, la formule $\mathcal{BCC}$-$\mathcal{FRM}$ $(\psi, \mathcal{C})$ est satisfiable si et seulement s'il existe un modèle de ψ qui satisfait chaque contrainte de cardinalité Booléenne de $\mathcal{C}$.*

Par ailleurs, étant donnée une base de croyances stratifiées Δ, nous la codons sous forme d'une formule propositionnelle classique Φ_Δ, et ce en introduisant une nouvelle variable propositionnelle par formule comme suit :

Définition 127 *Soit $\Delta = (S_1, \ldots, S_m)$ une base de croyances stratifiées. La formule propositionnelle Φ_Δ associée à Δ est donnée par :*

$$\Phi_\Delta = \bigwedge_{i=1}^{m} (\bigwedge_{\phi_{ij} \in S_i} (\neg New_{ij} \vee \phi_{ij})).$$

Dans ce qui suit, N_i dénotera l'ensemble des variables New_{ij}'s associées aux formules de la strate S_i :

$$N_i = \{New_{ij} \ tel \ que \ \phi_{ij} \in S_i\}.$$

Ceci nous permet de traduire les oracles MAX-GSAT-ARRAY et NGSAT-LEX utilisés dans l'algorithme 7.1 en un problème de satisfiabilité d'une formule $\mathcal{BCC}$-$\mathcal{FRM}$ de la manière suivante :

1. L'instance MAX-GSAT-ARRAY (Δ, P) est satisfiable si et seulement si la formule $\mathcal{BCC}$-$\mathcal{FRM}$ $(\Phi_\Delta, \{\#(P[i], |N_i|, N_i) : i = 1..m\})$ est satisfiable.
2. L'instance NGSAT-LEX (Δ, P, ψ) est satisfiable si et seulement si la formule $\mathcal{BCC}$-$\mathcal{FRM}$ $(\Phi_\Delta \wedge \neg\psi, \{\#(P[i], |N_i|, N_i) : i = 1..m\})$ est satisfiable.

Puisque la contrainte $\#(P[i], |N_i|, N_i)$ n'impose pas de restriction sur le maximum de variables de N_i qui peuvent être assignées à Vrai, nous la noterons dans ce qui suit par $\#(P[i], N_i)$ ce qui simplement signifie que le nombre précédent est au moins égal à $P[i]$.

Ensuite, en se basant sur le codage CNF des contraintes de cardinalité Booléennes, nous proposons une traduction des oracles précédents vers un problème de satisfiabilité classique comme le montre la proposition suivante dont la preuve découle directement de ce qui précède :

Proposition 5 *Soit $\Delta = (S_1, \ldots, S_m)$ une base de croyances stratifiées et soient P un vecteur d'entiers d'une taille m et ψ une formule propositionnelle. Soit Φ_Δ la formule propositionnelle associée à Δ selon la définition 127. Alors,*

1. *l'instance* MAX-GSAT-ARRAY (Δ, P) *est satisfiable si et seulement si la formule* $\Phi_\Delta \wedge \bigwedge_{i=1}^{m} \Psi_{\#(P[i],N_i)}$ *est satisfiable* [1],
2. *L'instance* NGSAT-LEX (Δ, P, ψ) *est satisfiable si et seulement si la formule* $\neg\psi \wedge \Phi_\Delta \wedge \bigwedge_{i=1}^{m} \Psi_{\#(P[i],N_i)}$ *est satisfiable.*

Appliquons maintenant le codage UT-MGAC rappelé dans la section 7.3. Donc, afin de calculer le profil P_Δ, nous faisons un nombre polynomial de tests de satisfiabilité de la formule $\Phi_\Delta \wedge \bigwedge_{i=1}^{m}(\mathcal{T}(N_i) \wedge \mathcal{C}(P_\Delta[i]))$.

Par ailleurs, nous avons

$$\Phi_\Delta \wedge \bigwedge_{i=1}^{m}(\mathcal{T}(N_i) \wedge \mathcal{C}(P_\Delta[i]))$$

est satisfiable si et seulement si

$$\Phi_\Delta \wedge \bigwedge_{i=1}^{m} \mathcal{T}(N_i) \not\models \bigvee_{i=1}^{m} \neg\mathcal{C}(P_\Delta[i]).$$

La bonne nouvelle est que la formule $\Phi_\Delta \wedge \bigwedge_{i=1}^{m} \mathcal{T}(N_i)$ est indépendante du profil P_Δ. Ceci implique qu'elle peut être considérée comme une partie fixe durant tout le calcul de P_Δ.

En outre, la formule $\bigvee_{i=1}^{m} \neg\mathcal{C}(P_\Delta[i])$ est une clause car c'est une disjonction de négations de termes.

Par conséquent, afin de calculer P_Δ en temps polynomial, il suffit de compiler la formule $\Phi_\Delta \wedge \bigwedge_{i=1}^{m} \mathcal{T}(N_i)$ vers un langage cible de compilation L. En effet, par définition, un langage cible de compilation permet d'effectuer l'inférence de clauses en temps polynomial.

En ce qui concerne le second point, nous avons $\neg\psi \wedge \Phi_\Delta \wedge \bigwedge_{i=1}^{m}(\mathcal{T}(N_i) \wedge \mathcal{C}(P[i])$ est satisfiable si et seulement si $\Phi_\Delta \wedge \bigwedge_{i=1}^{m} \mathcal{T}(N_i) \not\models \bigvee_{i=1}^{m} \neg\mathcal{C}(P_\Delta[i]) \vee \psi$.

Ainsi, encore une fois, si on compile la formule $\Phi_\Delta \wedge \bigwedge_{i=1}^{m} \mathcal{T}(N_i)$ vers n'importe quel langage cible de compilation L, alors le test d'inférence précédent peut être effectué en temps polynomial.

Il est à noter que la formule $\bigvee_{i=1}^{m} \neg\mathcal{C}(P_\Delta[i]) \vee \psi$ peut être facilement mise sous forme CNF en temps polynomial en supposant que ψ est donnée sous forme CNF.

Ainsi, la compilation de base de croyances stratifiées associée à l'inférence lexicographique que nous proposons est définie par :

Définition 128 *Soit* $\Delta = (S_1, \ldots, S_m)$ *avec* $(m \geq 1)$ *une base de croyances stratifiées et soit* L *un langage cible de compilation. Alors, la compilation de* Δ *par rapport à* L*, notée*

[1] Rappelons que $\Psi_{\#(P[i],N_i)}$ dénote le codage CNF de la contrainte $\#(P[i], N_i)$.

$CompLex(\Delta, L)$, est la compilation de la formule propositionnelle $\Phi_\Delta \wedge \bigwedge_{i=1}^{m} \mathcal{T}(N_i)$ vers le langage L :

$$CompLex(\Delta, L) \equiv L(\Phi_\Delta \wedge \bigwedge_{i=1}^{m} \mathcal{T}(N_i)).$$

Les propriétés relatives à cette compilation sont résumées par la proposition suivante :

Proposition 6 *Soit $CompLex(\Delta, L)$ la compilation de la base de croyances stratifiées Δ par rapport au langage cible de compilation L et soit ψ formule une formule CNF. Alors,*

1. *le profil P_Δ peut être calculé en temps polynomial,*
2. *l'inférence lexicographique de ψ à partir de Δ peut être effectuée en temps polynomial.*

Exemple 38 *Considérons la base de croyances stratifiées $\Delta = (S_1, S_2, S_3)$ telle que :*

- $S_1 = \{\neg p \vee b, p\}$,
- $S_2 = \{\neg p \vee \neg f, \neg p \vee s\}$,
- $S_3 = \{\neg b \vee f, \neg b \vee w\}$.

Vérifions si $\Delta \vdash_{lex} \neg f \vee w$.

Tout d'abord, donnons le codage propositionnel Φ_Δ associé à Δ selon la définition 127 :

$\Phi_\Delta \equiv (\neg p \vee b \vee \neg New_{11}) \wedge (p \vee \neg New_{12}) \wedge (\neg p \vee \neg f \vee \neg New_{21}) \wedge (\neg p \vee s \vee \neg New_{22}) \wedge (\neg b \vee f \vee \neg New_{31}) \wedge (\neg b \vee w \vee \neg New_{32})$.

Donc, $N_1 = \{New_{11}, New_{12}\}$, $N_2 = \{New_{21}, New_{22}\}$ et $N_3 = \{New_{31}, New_{32}\}$.

L'additionneur associé à N_1 est donné par : $\mathcal{T}(N_1) \equiv (\neg New_{11} \vee s_{11}) \wedge (New_{12} \vee \neg s_{12}) \wedge (\neg New_{12} \vee s_{11}) \wedge (New_{11} \vee \neg s_{12}) \wedge (\neg New_{11} \vee \neg New_{12} \vee s_{12}) \wedge (New_{11} \vee New_{12} \vee \neg s_{11})$.

Quant à l'additionneur correspondant à N_2, il est tel que :

$\mathcal{T}(N_2) \equiv (\neg New_{21} \vee s_{21}) \wedge (New_{22} \vee \neg s_{22}) \wedge (\neg New_{22} \vee s_{21}) \wedge (New_{21} \vee \neg s_{22}) \wedge (\neg New_{21} \vee \neg New_{22} \vee s_{22}) \wedge (New_{21} \vee New_{22} \vee \neg s_{21})$.

Enfin, l'additionneur relatif à N_3 est donné par :

$\mathcal{T}(N_3) \equiv (\neg New_{31} \vee s_{31}) \wedge (New_{32} \vee \neg s_{32}) \wedge (\neg New_{32} \vee s_{31}) \wedge (New_{31} \vee \neg s_{32}) \wedge (\neg New_{31} \vee \neg New_{32} \vee s_{32}) \wedge (New_{31} \vee New_{32} \vee \neg s_{31})$.

Calculons maintenant le profil P_Δ. Initialement, on a $P_\Delta = < 0, 0, 0 >$.

1. *A la première itération, on a $P_\Delta[1] = |S_1| = 2$. Donc, $\mathcal{C}(P_\Delta[1]) \equiv s_{11} \wedge s_{12}$ et $\mathcal{C}(P_\Delta[2]) \equiv \top$.*

   En outre, on a $\Phi_\Delta \wedge \mathcal{T}(N_1) \wedge \mathcal{T}(N_2) \wedge \mathcal{T}(N_3) \not\models (\neg s_{11} \vee \neg s_{12})$. Par conséquent, on passe à la strate suivante.

2. *A la deuxième itération, on a* $P_\Delta[2] = |S_2| = 2$ *ce qui implique que* $\mathcal{C}(P_\Delta[2]) \equiv s_{21} \wedge s_{22}$.
   On a $\Phi_\Delta \wedge \mathcal{T}(N_1) \wedge \mathcal{T}(N_2) \wedge \mathcal{T}(N_3) \not\models (\neg s_{11} \vee \neg s_{12}) \vee (\neg s_{21} \vee \neg s_{22})$ *ce qui nous permet de passer à la strate suivante.*
3. *A la troisième itération, on a* $P_\Delta[3] = |S_3| = 2$ *ce qui implique que* $\mathcal{C}(P_\Delta[3]) \equiv s_{31} \wedge s_{32}$.
   Dans ce cas, $\Phi_\Delta \wedge \mathcal{T}(N_1) \wedge \mathcal{T}(N_2) \wedge \mathcal{T}(N_3) \models (\neg s_{11} \vee \neg s_{12}) \vee (\neg s_{21} \vee \neg s_{22}) \vee (\neg s_{31} \vee \neg s_{32})$.
   De ce fait, $P_\Delta[3] \leftarrow 1$.
   On trouve que $\Phi_\Delta \wedge \mathcal{T}(N_1) \wedge \mathcal{T}(N_2) \wedge \mathcal{T}(N_3) \not\models (\neg s_{11} \vee \neg s_{12}) \vee (\neg s_{21} \vee \neg s_{22}) \vee (\neg s_{31})$

Par conséquent, $P_\Delta =< 2, 2, 1 >$.

Maintenant, on a $\Phi_\Delta \wedge \mathcal{T}(N_1) \wedge \mathcal{T}(N_2) \wedge \mathcal{T}(N_3) \not\models (\neg s_{11} \vee \neg s_{12}) \vee (\neg s_{21} \vee \neg s_{22}) \vee (\neg s_{31}) \vee (\neg f \wedge w)$. *On déduit alors que* $\Delta \vdash_{lex} \neg f \wedge w$.

Par ailleurs, il est claire que le fait de compiler la formule $\Phi_\Delta \wedge \mathcal{T}(N_1) \wedge \mathcal{T}(N_2) \wedge \mathcal{T}(N_3)$ *vers n'importe quel langage cible de compilation* L*, ce qui correspond à la compilation de* $CompLex(\Delta, \mathcal{L})$*, permet d'effectuer tous les tests d'inférence précédents en temps polynomial (par définition d'un langage cible de compilation. Autrement dit, on voit bien que l'approche de compilation proposée permet d'effectuer l'inférence lexicographique en temps polynomial.*

7.5 Comparaison aux travaux connexes

Une approche de compilation de base de croyances stratifiées relativement à l'inférence lexicographique a été proposée dans [13]. Cependant, l'objectif correspondant est de réduire le problème de l'inférence lexicographique de la classe Δ_2^p-complet à un problème coNP-complet, contrairement à la nôtre qui le réduit à un problème polynomial. De plus, cette approche ne permet pas de calculer le profil en temps polynomial, et donc elle ne permet pas de modifier la relation de priorité entre les strates.

Par ailleurs, les approches de compilation les plus proches de la nôtre sont celles proposées dans [27, 31, 38]. A l'image de notre approche, elles introduisent une nouvelle variable par formule et offrent la possibilité de modifier les priorités entre les strates sans aucune recompilation. Toutefois, l'approche proposée dans [27] est basée sur le langage $OBDD$ et celle proposée dans [31] (rappelée à la fin du chapitre 5) est efficace par rapport au langage DNF. Quant à l'approche introduite dans [38], elle tire profit à la fois des langages DNF et $DNNF$. En revanche, notre approche se trouve paramétrable par n'importe quel langage cible de compilation. En particulier, contrairement aux autres approches, notre compilation peut être efficacement paramétrée par le langage des impliqués premiers PI. Rappelons que PI et $OBDD$ sont incomparables en termes de concision.

Donc, certaines formules propositionnelles ont une représentation $OBDD$ exponentiellement plus large que leur représentation en PI, et inversement. Il est de même, PI et DNF sont incomparables. De plus, PI n'est pas plus concis que $DNNF$ mais on ne sait pas si $DNNF$ est plus concis que PI ou non. En conclusion, en ce qui concerne le critère de concision, notre approche est complémentaire par rapport à celles proposée dans [27, 31, 38].

7.6 Conclusion

Dans ce chapitre, nous avons proposé une nouvelle approche de compilation de bases de croyances stratifiées relative à l'inférence lexicographique. Cette approche s'appuie sur un codage CNF des contraintes de cardinalité Booléennes.

L'approche que nous avons proposée permet d'effectuer l'inférence lexicographique en temps polynomial. De plus, elle est qualifiée de flexible dans le sens où elle n'implique aucune recompilation lorsque les priorités changent entre les strates. En outre, elle est paramétrable par n'importe quel langage cible de compilation. Par conséquent, elle est complémentaire par rapport aux approches existantes. En particulier, elle offre la possibilité de tirer profit du fameux langage PI des impliqués premiers à l'inverse des approches existantes.

Chapitre 8

Compilation de l'inférence MSP

8.1 Introduction

Le raisonnement tolérant les exceptions occupe une position importante en Intelligence Artificielle. Parmi les approches naturelles de raisonnement en présence d'exceptions, on a l'approche possibiliste, plus précisément l'inférence MSP (Minimum Specificity Principle) introduite dans [12]. Dans une telle approche, la donnée de départ est une théorie des défauts Υ c'est-à-dire un couple (D, W), où D est un ensemble de règles de défauts ou des règles admettant des exceptions, et W est un ensemble de règles strictes qui n'admettent pas d'exceptions. D'un point de vue syntaxique, l'inférence MSP revient à calculer une base possibiliste, où chaque formule exprime une règle telle que plus le degré de certitude est élevé, plus la règle correspondante est exceptionnelle. Ensuite, appliquer l'inférence possibiliste sur la base possibiliste obtenue nous permet de dériver les conclusions MSP de la théorie des défauts en question.

Néanmoins, calculer la base possibiliste est coûteux d'un point de vue calculatoire car ça demande un nombre d'appels polynomial à un oracle NP. De plus, appliquer l'inférence possibiliste à partir de cette dernière constitue un problème $\Delta_2^p[O(log\ n)]$-complet ce qui est critique pour des applications où les requêtes devraient être résolues efficacement.

Afin de dériver des conséquences MSP à partir d'une théorie des défauts $\Upsilon = (D, W)$ en temps polynomial, une première méthode consiste à calculer dans un premier temps la base stratifiée correspondante, ensuite compiler la base possibiliste résultante selon une méthode de compilation possibiliste, à l'image de celle que nous avons proposée dans le chapitre 6, en vue d'assurer une inférence possibiliste polynomiale.

Toutefois, la génération de la base stratifiées est coûteuse d'un point de vue calculatoire. En effet, elle requiert un nombre d'appel polynomial à un oracle NP. Par conséquent, cette méthode revient à résoudre un problème faisant un nombre polynomial d'appels à un oracle NP suivi d'une compilation d'une base stratifiée pour l'inférence possibiliste. Ce point devient plus critique lorsque la théorie des défauts est fréquemment mise à jour en rajoutant ou en supprimant certaines formules par exemple car le processus doit être entièrement refait, et le coût inhérent est évidemment élevé.

Nous proposons alors une nouvelle méthode de compilation de théories des défauts par rapport à l'inférence MSP. Cette compilation est particulièrement adaptée à des théories des défauts fréquemment mises à jour. De plus, elle nous permet de calculer efficacement à la fois la base stratifiée associée ainsi que l'inférence possibiliste à partir de cette dernière. Par conséquent, une telle compilation permet d'effectuer l'inférence MSP en temps polynomial.

L'approche de compialtion proposée dans ce chapitre a fait l'objet de deux publications : [18] et [19].

8.2 Rappel sur le traitement possibiliste des théories des défauts

Une assertion conditionnelle ou une règle de défauts est une règle générique de la forme "généralement, si α alors β" ayant éventuellement des exceptions. Ces règles sont notées par "$\alpha \rightarrow \beta$". Une base de défauts est un ensemble $D = \{\alpha_i \rightarrow \beta_i : i = 1, n\}$ de règles de défauts. On note par $W = \{\delta_i \Rightarrow \gamma_i : i = 1, n'\}$ l'ensemble des règles strictes. Maintenant, une théorie des défauts Υ est définie par la donnée de la paire (D, W).

Une interprétation ω satisfait une règle $\alpha \rightarrow \beta$ si et seulement si ω satisfait la formule classique $\neg\alpha \vee \beta$. D'une manière générale, ω satisfait une théorie des défauts $\Upsilon = (D, W)$ si et seulement si ω satisfait W et chaque règle de défaut de D. De plus, l'interprétation ω vérifie une règle de défauts $\alpha \rightarrow \beta$ si et seulement si ω satisfait la formule $\alpha \wedge \beta$. En outre, un ensemble de règles de défauts D tolère une règle de défauts $\alpha \rightarrow \beta$ sous W si et seulement s'il existe une interprétation ω qui satisfait D et W et vérifie la règle de défauts $\alpha \rightarrow \beta$.

Dans [12], il a été proposé de voir chaque assertion conditionnelle $\alpha \rightarrow \beta$ comme une contrainte exprimant que la situation où $\alpha \wedge \beta$ est vraie est plus plausible que celle où $\alpha \wedge \neg\beta$ est vraie ce qui est exprimé en théorie des possibilités par l'inégalité stricte $\Pi(\alpha \wedge \beta) > \Pi(\alpha \wedge \neg\beta)$. De plus, les règles strictes de la forme "tous les α's sont des β's" sont exprimées par la condition $\Pi(\alpha \wedge \neg\beta) = 0$.

Par conséquent, une théorie des défauts Υ peut être vue comme une famille de contraintes $C(\Upsilon)$ restreignant une famille $\prod(\Upsilon)$ de distributions de possibilités. Les éléments de $\prod(\Upsilon)$ sont dits compatibles avec $\prod(\Upsilon)$ et sont définis formellement par :

Définition 129 *Une distribution de possibilité π est dite **compatible** avec une théorie des défauts $\Upsilon = (D, W)$, noté $\pi \in \prod(\Gamma)$, si et seulement si on a :*
- *pour chaque règle stricte $\delta_j \Rightarrow \gamma_j$ de W, $\Pi(\delta_j \wedge \neg\gamma_j) = 0$,*
- *pour chaque règle de défaut $\alpha_i \rightarrow \beta_i$ de D, $\Pi(\alpha_i \wedge \beta_i) > \Pi(\alpha_i \wedge \neg\beta_i)$,*

où Π est la mesure de possibilité induite par π.

L'inférence MSP (MSP pour Minimum Specificity Principle), proposée dans [12], est définie comme suit :

Définition 130 *Soit π une distribution de possibilité choisie à partir de $\prod(\Upsilon)$ en utilisant le principe de minimum de spécificité. Une assertion conditionnelle $\alpha \rightarrow \beta$ est dite conséquence MSP de Υ, noté par $\Upsilon \models_{MSP} \alpha \rightarrow \beta$, si et seulement si $\Pi(\alpha \wedge \beta) > \Pi(\alpha \wedge \neg\beta)$.*

D'un point de vue syntaxique, l'inférence MSP revient à calculer dans un premier temps une base de croyances stratifiées $\Delta_\Upsilon = (S_1, \ldots, S_m)$ comme le montre l'algorithme

8.1. Il est à noter que dans ce cas les formules d'une strate S_i avec $1 \leq i \leq m$ sont plus prioritaires que celles d'une strate S_j pour tout $1 \leq j < i$. Ainsi, la strate S_m contient les formules les plus prioritaites et S_1 contient les formules les moins prioritaires.

L'algorithme 8.1 étend celui développé dans le système Z [81] au cas où l'on traite à la fois des règles strictes et des règles de défauts. Il consiste à assigner à chaque règle de défauts un degré. Les règles de défauts ayant le degré le plus faible sont les plus générales. Elles sont telles que assigner leurs antécédents à vrai ne cause pas d'incohérence. Donc, on a besoin :

- de déterminer quelles sont les règles de défauts tolérées,
- de supprimer ces règles de défauts tolérées afin de stratifier les règles restantes.

Une fois la base stratifiée correspondante générée, on applique une inférence possibiliste. En fait, il a été démontré que $\Upsilon \models_{MSP} \alpha \rightarrow \beta$ si et seulement si $\Delta_\Upsilon \cup \{(\alpha, 1)\} \models_\pi \beta$ c'est à dire β est une conclusion du conditionnement de Δ_Υ par α [12].

Algorithme 8.1: Contrepartie syntaxique de l'inférence MSP
Données: une théorie des défauts $\Upsilon = (D, W)$
Résultat: une base stratifiée Δ_Υ
début
  $m \leftarrow 1$;
  tant que $D \neq \emptyset$ **faire**
    $A \leftarrow \{\neg\alpha_i \vee \beta_i : \alpha_i \rightarrow \beta_i \in D\}$;
    $S'_m \leftarrow \{\alpha_i \rightarrow \beta_i : \alpha_i \rightarrow \beta_i \in D$ et $A \cup W \cup \{\alpha_i\}$ est cohérente $\}$
    si $S'_m = \emptyset$ **alors**
      arrêter l'algorithme
    $D \leftarrow D - S'_m$;
    $m \leftarrow m + 1$;
  $\Delta_\Upsilon \leftarrow (S_1, \ldots, S_m)$ où $S_m = W$ et pour $j = 1, m-1 : S_j = \{\neg\alpha_k \vee \beta_k : \alpha_k \rightarrow \beta_k \in S'_j\}$
  retourner Δ_Υ ;
fin

Illustrons l'Algorithme 8.1 à travers l'exemple suivant qui décrit fragment d'une politique de sécurité dans un contexte médical.

Exemple 39 *Nous considérons les règles suivantes :*

- *"Tous les chirurgiens sont des médecins"*
- *"Généralement, les chirurgiens peuvent écrire les rapports des opérations chirurgicales.*
- *"Généralement, les médecins ne peuvent pas écrire les rapports des opérations chirurgicales."*

Formellement, ceci peut être décrit par $\Upsilon = (D, W)$ *où* $D = \{s \rightarrow w, d \rightarrow \neg w\}$, *et* $W = \{s \Rightarrow d\}$.

- *A la première étape,* $A = \{\neg s \vee w, \neg d \vee \neg w\}$. $A \cup \{d\} \cup W$ *est cohérente alors que* $A \cup \{s\} \cup W$ *ne l'est pas donc* $S_1' \leftarrow \{d \rightarrow \neg w\}$.
- *A la deuxième étape,* $A = \{\neg s \vee w\}$. $A \cup \{s\} \cup W$ *est cohérente donc on obtient* $S_2' = \{s \rightarrow w\}$.

Alors, $\Delta_\Upsilon = S_1, \ldots, S_3$ *tel que :*

- $S_1 = \{\neg d \vee \neg w\}$,
- $S_2 = \{\neg s \vee w\}$,
- $S_3 = \{\neg s \vee d\}$.

Maintenant étant donné un médecin qui est chirurgien, on voudrait savoir s'il est autorisé à écrire les rapports des opérations chirurgicales ou non.

Clairement, on a $\Delta_\Upsilon \cup \{(s \wedge d, 1)\} \models_\pi w$ *ce qui implique que* $\Upsilon \models_{MSP} s \wedge d \rightarrow w$ *ce qui veut dire que ce chirurgien est autorisé à écrire les rapports des opérations chirurgicales.*

8.3 Compilation des théories des défauts possibilistes

Notre approche de compilation comprend trois étapes :

- la première étape consiste à coder la théorie des défauts sous la forme d'une base propositionnelle,
- la second étape consiste à calculer la base de croyances correspondante,
- la dernière étape concerne la dérivation des conséquences MSP.

Ces trois étapes seront successivement décrites dans les sous-sections suivantes :

8.3.1 Codage propositionnel de la théorie des défauts

Étant donnée une théorie des défauts $\Upsilon = (D, W)$, notre approche commence par coder cette théorie sous la forme d'une base propositionnelle classique. Plus précisément, ce codage revient à introduire une nouvelle variable propositionnelle A_i pour chaque règle des défauts $\alpha_i \rightarrow \beta_i$. Une variable propositionnelle A_0 est associée à W. Formellement, notre codage est donné par la définition suivante :

Définition 131 *Étant donnée une théorie des défauts* $\Upsilon = (D, W)$ *telle que* $D = \{\alpha_i \rightarrow \beta_i, 1 \leq i \leq n\}$ *et* $W = \{\delta_j \Rightarrow \gamma_j, 1 \leq j \leq n'\}$. *Soit* $\{A_k, 0 \leq k \leq n\}$ *un ensemble de* n *nouvelles variables propositionnelles. Le codage propositionnel de* Υ, *noté par* K_Υ, *est donné par :*

$$K_\Upsilon = \{\neg\alpha_i \vee \beta_i \vee A_i : \alpha_i \rightarrow \beta_i \in D\} \cup \{\neg\delta_i \vee \gamma_i \vee A_0 \ : \delta_i \Rightarrow \gamma_i \in W\}.$$

Ce codage revient alors à remplacer chaque règle des défauts $\alpha_i \rightarrow \beta_i$ par une formule propositionnelle classique $\neg\alpha_i \vee \beta_i \vee A_i$ avec $1 \leq i \leq n$. Quant aux règles strictes de W, on remplace chaque règle $\delta_j \Rightarrow \gamma_j$ $(1 \leq j \leq n')$ par la formule $\delta_j \vee \gamma_j \vee A_0$. Remarquons que la taille d'un tel codage est clairement polynomiale par rapport à la taille de la théorie des défauts initiale.

Étant donné un langage cible de compilation L, la compilation de la théorie des défauts Υ par rapport à L que nous proposons n'est rien d'autre que la compilation de son codage propositionnel K_Υ vers L. Cette compilation est donnée plus formellement par la définition suivante :

Définition 132 *Soit $\Upsilon = (D, W)$ une théorie des défauts et soit L un langage cible de compilation. Soit K_Υ le codage propositionnel associé selon la définition 131. La compilation de Υ par rapport à L, notée $CompMSP(\Upsilon, L)$, est donnée par la compilation de la base classique K_Υ vers le langage cible de compilation L :*

$$CompMSP(\Upsilon, L) = L(K_\Upsilon).$$

Il est à noter que la plupart des compilateurs propositionnels existants requièrent que l'entrée soit donnée sous forme CNF. Donc nous devons mettre dans un premier temps (si ce n'est pas le cas) $\neg\alpha_i$'s, β_i's, $\neg\delta_i$'s et γ_i's sous une forme CNF ce qui nous permet de calculer la CNF de K_Υ ;

Exemple 40 *Considérons à nouveau l'exemple 39.*

Rappelons que $\Upsilon = (D, W)$ où $D = \{s \rightarrow w, d \rightarrow \neg w\}$, et $W = \{s \Rightarrow d\}$.

Soient A_0, A_1 et A_2 trois nouvelles variables propositionnelles.

Le codage propositionnel de Υ est $K_\Upsilon = (\neg s \vee d \vee A_0) \wedge (\neg s \vee w \vee A_1) \wedge (\neg d \vee \neg w \vee A_2)$.

Une compilation de Υ par rapport à $DNNF$ est la compilation DNNF classique de K_Υ, c'est-à-dire $CompMSP(\Upsilon, DNNF) = [s \wedge [(w \wedge ((A_0 \wedge \neg d) \vee (A_2 \wedge d))) \vee (A_1 \wedge \neg w \wedge (A_0 \vee d))]] \vee [\neg s \wedge (A_2 \vee \neg d \vee \neg w)]$.

8.3.2 Calcul de la base possibiliste associée

Nous proposons de calculer la base stratifiée correspondante en utilisant l'algorithme 8.2. Cet algorithme admet en entrée la compilation de Υ par rapport à $L : CompMSP(\Upsilon, L)$. La sortie est une séquence $(I_1, \ldots, I_m)$ où I_i $(1 \leq i \leq m)$ contient les indices des règles des défauts tolérées à l'itération i, en particulier I_m contient l'indice 0 qui est associé aux règles strictes de W. Dans ce qui suit, $\overline{I}$ dénote l'ensemble $\{1, \ldots, n\} \setminus I$.

En utilisant la séquence $(I_1, \ldots, I_m)$ retournée par cet algorithme, on peut générer la base stratifiée associée à Υ comme le montre la proposition suivante :

Algorithme 8.2: Génération de la base stratifiée associée
Données: $CompMSP(\Upsilon, L)$ (une compilation de Υ par rapport à L)
Résultat: une séquence $(I_1, \ldots, I_m)$
début
  $m \leftarrow 1$;
  $I \leftarrow \{i : 0 \leq i \leq n\}$;
  $K \leftarrow CompMSP(\Upsilon, L)|(\bigwedge_{i \in I} \neg A_i)$;
  tant que $I \neq \{0\}$ **faire**
    $I_m \leftarrow \{i \in I - \{0\} : K \not\models \neg\alpha_i\}$;
    si $I_m = \emptyset$ **alors**
      Arrêter l'algorithme ;
    $I \leftarrow I - I_m$;
    $K \leftarrow CompMSP(\Upsilon, L)|(\bigwedge_{i \in I} \neg A_i \wedge \bigwedge_{i \in \overline{I}} A_i)$;
    $m \leftarrow m + 1$;
  $I_m \leftarrow \{0\}$;
  retourner $(I_1, \ldots, I_m)$;
fin

Proposition 7 *Soit $\Upsilon = (D, W)$ une théorie des défauts et soit $CompMSP(\Upsilon, L)$ la compilation correspondante par rapport à L. Soit $(I_1, \ldots, I_m)$ $(m \geq 1)$ la séquence retournée par l'algorithme 8.2 où chaque I_i dénote l'ensemble des règles des défauts tolérées à l'itération i. Alors, la base de croyances stratifiées associée à Υ est donnée par : $\Delta_\Upsilon = (S_1, \ldots, S_m)$ où $S_i = \{\neg\alpha_k \vee \beta_k : \alpha_k \rightarrow \beta_k \in D$ et $k \in I_i\}$ pour $1 \leq i < m$ et $S_m = \{\neg\delta_k \vee \gamma_k : \delta_k \Rightarrow \gamma_k \in W\}$.*

Preuve.

Selon l'algorithm 8.2, initialement, on a $I = \{0, \ldots, n\}$. Donc, $\overline{I} = \emptyset$.

Soit $K_W \equiv \bigwedge_{\phi \in W} \phi$.

Maintenant, $CompMSP(\Upsilon, L) \equiv \bigwedge_{i=1}^{n} (\neg\alpha_i \vee \beta_i \vee A_i) \wedge (K_W \vee A_0)$ car par définition $CompMSP(\Upsilon, L)$ est la compilation de K_Υ vers un langage cible de compilation L ce qui implique que $CompMSP(\Upsilon, L)$ et K_Υ sont équivalentes.

Par ailleurs, à la première itération où $m = 1$, on a

$$K \equiv (\bigwedge_{i=1}^{n} (\neg\alpha_i \vee \beta_i \vee A_i) \wedge (K_W \vee A_0))|(\bigwedge_{i \in \{0,\ldots,n\}} \neg A_i).$$

Donc , $K \equiv \bigwedge_{i=1}^{n}(\neg\alpha_i \vee \beta_i) \wedge K_W$.

En effet, la formule $(\neg\alpha_i \vee \beta_i \vee A_i)|\neg A_i \equiv \neg\alpha_i \vee \beta_i \vee \bot \equiv \neg\alpha_i \vee \beta_i$ alors que $(\neg\alpha_i \vee \beta_i \vee A_i)|A_i \equiv \neg\alpha_i \vee \beta_i \vee \top \equiv \top$ ce qui revient à supprimer la règle correspondante.

Maintenant, une règle des défauts $\alpha_i \rightarrow \beta_i \in D$ est tolérée si et seulement si $\alpha_i \wedge \beta_i \wedge \bigwedge_{j=1}^{n}(\neg\alpha_j \vee \beta_j) \wedge K_W$ est cohérente ce qui équivaut à dire que $\alpha_i \wedge \bigwedge_{j=1}^{n}(\neg\alpha_j \vee \beta_j) \wedge K_W$ est cohérente car $\alpha_i \wedge \beta_i \wedge (\neg\alpha_i \vee \beta_i) \equiv \alpha_i \wedge (\neg\alpha_i \vee \beta_i)$.

Ainsi, ce test revient à vérifier si $K \nvDash \neg\alpha_i$.

Remarquons que K appartient à L car n'importe quel langage cible de compilation satisfait le conditionnement. Par conséquent, le test $K \nvDash \neg\alpha_i$ s'effectue en temps polynomial.

De ce fait, I_1 contient les indices des règles des défauts de D qui sont tolérées à la première itération. De plus, ces indices sont retirés de I afin de les ignorer à l'itération suivante.

Par conséquent, on aura à la deuxième itération :

$$K \equiv CompMSP(\Upsilon, L)|(\bigwedge_{i\in I} \neg A_i \wedge \bigwedge_{i\in \overline{I}} A_i) \equiv \bigwedge_{i\in I}(\neg\alpha_i \vee \beta_i) \wedge (K_W).$$

Ainsi, on ne considère pas les règles tolérées à l'itération précédente car les indices correspondant ont été supprimés de I.

De manière similaire, nous définissons les règles tolérées en vérifiant si $K \models \neg\alpha_i$ pour $i \in I \setminus \{0\}$. Les indices relatifs sont sauvegardés dans I_j et retirés de I et ainsi de suite.

A la fin, $I_m \leftarrow \{0\}$ ce qui correspond aux règles sans exceptions c'est-à-dire W.

Donc, une fois nous avons identifié les règles tolérées à chaque itération via la séquence $(I_1, \ldots, I_m)$, il est claire que la base de croyances stratifiées associée à Υ est telle que : $\Delta_\Upsilon = (S_1, \ldots, S_m)$ où $S_i = \{\neg\alpha_k \vee \beta_k : \alpha_k \rightarrow \beta_k \in D \text{ et } k \in I_i\}$ pour $1 \leq i < m$ et $S_m = \{\neg\delta_k \vee \gamma_k : \delta_k \Rightarrow \gamma_k \in W\}$.

■

En termes de complexité de calcul, nous montrons la proposition suivante.

Proposition 8 *L'algorithme 8.2 s'effectue en temps polynomial.*

Preuve.

Rappelons que $CompMSP(\Upsilon, L)$ appartient à un langage cible de compilation. Rappelons aussi, que par définition, un langage cible de compilation permet au moins d'effectuer l'inférence de clauses en temps polynomial.

De plus, le conditionnement d'une formule appartenant à un langage cible de compilation sur un terme cohérent s'effectue en temps polynomial, et le résultat appartient

toujours au même langage cible de compilation.

Par conséquent, l'instruction

$K \leftarrow CompMSP(\Upsilon, L)|(\bigwedge_{i \in I} \neg A_i \wedge \bigwedge_{i \in \overline{I}} A_i)$;

s'effectue en temps polynomial et la formule K appartient au langage L.

Ainsi, le test $K \not\models \neg\alpha_i$ s'effectue en temps polynomial car K appartient au langage cible de compilation L et les α_i's sont supposées sous forme CNF. ■

Illustrons l'algorithme 8.2 avec l'exemple suivant :

Exemple 41 *Considérons l'exemple 40 où* $CompMSP(\Upsilon, DNNF) \equiv [s \wedge [(w \wedge ((A_0 \wedge \neg d) \vee (A_2 \wedge d))) \vee (A_1 \wedge \neg w \wedge (A_0 \vee d))]] \vee [\neg s \wedge (A_2 \vee \neg d \vee \neg w)]$.

- *Initialement, nous avons* $m \leftarrow 1$, $I \leftarrow \{0, 1, 2\}$ *et* $K \leftarrow CompMSP(\Upsilon, DNNF)|(\neg A_0 \wedge \neg A_1 \wedge \neg A_2) \equiv \neg s \wedge (\neg d \vee \neg w)$.
- *A la première itération,* $K \models \neg s$ *alors que* $K \not\models \neg d$. *Donc* $I_1 \leftarrow \{2\}$ *où* 2 *représente l'indice de la règle des défauts* $d \rightarrow \neg w$. *En outre,* $K \leftarrow CompMSP(\Upsilon, DNNF)|(\neg A_0 \wedge \neg A_1 \wedge A_2) \equiv (s \wedge w \wedge d) \vee \neg s$.
- *A la seconde itération,* $K \not\models \neg s$ *donc* $I_2 \leftarrow \{1\}$. $I \leftarrow \{0\}$ *ce qui arrête la boucle.*
- $I_3 \leftarrow \{0\}$.
- *Donc l'algorithme 8.2 retourne* (I_0, I_1, I_2) *à partir duquel* Δ'_Υ *est donnée par :* $\Delta'_\Upsilon = (S_1, S_2, S_3)$ *où* $S_1 = \{\neg d \vee \neg w)\}$, $S_2 = \{\neg s \vee w)\}$ *et* $S_3 = \{\neg s \vee d\}$.

On peut facilement voir que Δ'_Υ *correspond exactement à* Δ_Υ *de l'exemple 39.*

8.3.3 Dériver des conséquences MSP

Nous montrons maintenant comment on peut tirer profit de la compilation de Υ, $CompMSP(\Upsilon, L)$, afin de dériver en temps polynomial les conséquences MSP à partir de Υ comme le décrit l'algorithme 8.3.

Étant donnée une règle des défauts $\alpha \rightarrow \beta$ qu'on veut vérifier si c'est une conséquence MSP, en supposant avoir son antécédent sous forme CNF, nous montrons la proposition suivante :

Proposition 9 *Soit* $\alpha \rightarrow \beta$ *une assertion conditionnelle , l'algorithme 8.3 vérifie en temps polynomial si* $\Upsilon \models_{MSP} \alpha \rightarrow \beta$ *étant donnée la formule* $\neg\alpha$ *sous forme CNF.*

Preuve.

Tout d'abord, rappelons qu'il a été prouvé dans [12] que $\Upsilon \models_{MSP} \alpha \rightarrow \beta$ si et seulement si $\Delta_\Upsilon \cup \{(\alpha, 1)\} \models_\pi \beta$.

Ensuite, $\Delta_\Upsilon \cup \{(\alpha, 1)\}$ est stratifiée sous la forme $(S_1, \ldots, S_m)$ telle que $S_i = \{\neg\alpha_k \vee \beta_k : \alpha_k \rightarrow \beta_k \in D$ et $k \in I_i\}$ pour $1 \leq i < m$ et $S_m = W \cup \{\alpha\}$.

Algorithme 8.3: Calcul des conséquences MSP
Data: $CompMSP(\Upsilon, L)$ (une compilation de Υ par rapport à L),
la séquence $(I_1, \ldots, I_m)$ donnée par l'algorithme 8.2,
une assertion conditionnelle $\alpha \rightarrow \beta$

début
- $i \leftarrow$ m ;
- $IsConclusion \leftarrow faux$;
- $IsSat \leftarrow vrai$;
- **tant que** ($i \geq 1$ ***et*** $IsSat$ ***et*** $\neg IsConclusion$) **faire**
  - $\gamma \leftarrow (\bigwedge_{k \in X_i} \neg A_k \wedge \bigwedge_{k \in \overline{X_i}} A_k)$ où $X_i = \bigcup_{j=i}^{m} I_j$
  - $K \leftarrow CompMSP(\Upsilon, L)|\gamma$;
  - **si** $K \models \neg\alpha$ **alors**
    - $IsSat \leftarrow faux$
  - **sinon**
    - **si** $K \models \neg\alpha \vee \beta$ **alors**
      - $IsConclusion \leftarrow$ vrai ;
    - **sinon** $i \leftarrow i$ - 1 ;
- **si** $IsConclusion = vrai$ **alors**
  - $\Upsilon \models_{MSP} \alpha \rightarrow \beta$
- **sinon**
  - $\Upsilon \not\models_{MSP} \alpha \rightarrow \beta$

fin

De plus, $\Delta_\Upsilon \cup \{(\alpha, 1)\} \models_\pi \beta$ si et seulement si $\exists i \geq 1$ tel que $\bigcup_{j=i}^{m} S_j \cup \{\alpha\}$ est cohérent et $\bigcup_{j=i}^{m} S_j \cup \{\alpha\} \models \beta$.

Donc, on commence à partir de la strate la plus prioritaire jusqu'à ce que soit on arrive à conclure la formule en question soit on rencontre une incohérence.

Maintenant, $\bigcup_{j=i}^{m} S_j$ peut être obtenue en temps polynomial à partir de $CompMSP(\Upsilon, L)$ en la conditionnant sur le terme $(\bigwedge_{k \in X_i} \neg A_k \wedge \bigwedge_{k \in \overline{X_i}} A_k)$ où $X_i = \bigcup_{j=i}^{m} I_j$. Ceci correspond à

l'instruction $K \leftarrow CompMSP(\Upsilon, L)|\gamma$.

Notons que cette opération s'effectue en temps polynomial et que K appartient toujours au langage L.

Ensuite, on teste la satisfiabilité de $K \cup \{\alpha\}$ ce qui équivaut à vérifier si $K \models \neg\alpha$.

Cette dernière inférence s'effectue en temps polynomial en supposant avoir $\neg\alpha$ sous forme CNF.

Si $K \models \neg\alpha$ on conclut que $\alpha \rightarrow \beta$ n'est pas une conséquence MSP de Υ. Autrement dit, on teste si $K \models \neg\alpha \vee \beta$, et ce test s'effectue aussi en temps polynomial. ■

Voyons ce que donne cette compilation par rapport à l'exemple suivant.

Exemple 42 *Considérons l'exemple 41 et appliquons l'algorithme 8.3 afin de vérifier si un chirurgien est autorisé ou non à écrire les rapports des opérations chirurgicales.*

- *A la première itération, $i \leftarrow 3$ donc $K \leftarrow CompMSP(\Upsilon, L)|(\neg A_0 \wedge A_1 \wedge A_2) \equiv [s \wedge ((w \wedge d) \vee (\neg w \wedge d))] \vee \neg s$. Nous avons $K \not\models \neg s \vee \neg d$ et $K \not\models (\neg s \vee \neg d) \vee w$.*
- *A la seconde itération, $K \leftarrow CompMSP(\Upsilon, L)|(\neg A_0 \wedge \neg A_1 \wedge A_2) \equiv (s \wedge w \wedge d) \vee \neg s$. Nous avons $K \not\models \neg s \vee \neg d$ et $K \models (\neg s \vee \neg d) \vee w$.*

Donc on déduit que $\Upsilon \models_{MSP} s \wedge d \rightarrow w$ ce qui signifie qu'étant donné un chirurgien, il est autorisé à écrire les rapport des opérations chirurgicales.

8.4 Flexibilité de l'approche proposée

Dans cette section, nous montrons la flexibilité offerte par notre approche. Tout d'abord, cette approche est paramétrée par n'importe quel langage cible de compilation, puisqu'elle repose uniquement sur la déduction de clauses et le conditionnement. Ainsi, on peut tirer profit d'un bon nombre de langages de compilation qui sont complémentaires relativement au critère de concision.

Par ailleurs, étant donnée une théorie des défauts compilée via cette approche, il se trouve que c'est possible d'ajuster la stratification relative, pour résoudre des problèmes d'ambiguïté par exemple, et ce sans aucun coût supplémentaire.

Illustrons ce point avec le fameux exemple de Nixon Diamond.

Exemple 43 *Soit $\Upsilon = (D, W)$ une théorie des défauts où $D = \{Q \rightarrow P, R \rightarrow \neg P\}$ exprime que "généralement, les quakers sont pacifistes" et "généralement, les républicain ne sont pas pacifistes". La base $W = \{R \Rightarrow A\}$ reflète le fait que "les républicains sont américains".*

Soient A_0, A_1 et A_2 trois nouvelles variables propositionnelles. Le codage propositionnel de Υ est $K_\Upsilon = \{(\neg R \vee A \vee A_0), (\neg Q \vee P \vee A_1), (\neg R \vee \neg P \vee A_2)\}$.

Une compilation de Υ relative à $DNNF$ est

$CompMSP(\Upsilon, DNNF) \equiv [\neg R \wedge \neg P \wedge (\neg Q \vee A_1)] \vee [\neg R \wedge P] \vee [R \wedge \neg P \wedge (A \vee A_0) \wedge (\neg Q \vee A_1)] \vee [R \wedge P \wedge A_2 \wedge (A \vee A_0)]$.

On peut facilement voir qu'il n'existe pas deux variables partagées entre deux conjonctions.

Appliquons maintenant l'algorithme 8.2 pour dériver la stratification associée.

Donc, $K \leftarrow DNNF(\Upsilon)|\neg A_0 \wedge \neg A_1 \wedge A_2 \equiv [\neg R \wedge \neg P \wedge \neg Q] \vee [\neg R \wedge P] \vee [R \wedge \neg P \wedge A \wedge \neg Q]$.

$K \not\models \neg Q$ *et* $K \not\models \neg R$ *ce qui implique que* $I_1 \leftarrow \{1, 2\}$. *Enfin,* $I_2 \leftarrow \{0\}$.

La base stratifiée générée via l'inférence MSP est donnée par :

$\Delta_\Upsilon = (S_1, S_2)$ *avec* $S_1 = \{\neg Q \vee P, \neg R \vee \neg P\}$ *et* $S_2 = \{\neg R \vee A\}$.

On voit que $Q \rightarrow P$ *et* $R \rightarrow \neg P$ *sont au même niveau. De ce fait, étant donné un individu qui est à la fois républicain et quaker, en utilisant l'algorithme 8.3, on ne peut pas déduire s'il est pacifiste ou pas.*

Maintenant, si un expert préfère $Q \rightarrow P$ *à* $R \rightarrow \neg P$, *il suffit d'appliquer l'algorithme 8.3 avec la séquence* (I_1, I_2, I_3) *où* $I_1 = \{2\}$, $I_2 = \{1\}$ *et* $I_3 = \{0\}$, *et dans ce cas on déduit que* $R \wedge Q \rightarrow P$.

Inversement, si l'expert préfère $R \rightarrow \neg P$ *à* $Q \rightarrow P$, *appliquer l'algorithme 8.3 avec la séquence* (I_1, I_2, I_3) *où* $I_1 = \{1\}$, $I_2 = \{2\}$ *et* $I_3 = \{0\}$, *ce qui donne* $R \wedge Q \rightarrow \neg P$.

Enfin, notons que toutes les étapes précédentes s'effectuent en temps polynomial.

De plus, il est à noter que lorsque l'on ajuste la stratification proposée par le système MSP, il faut vérifier que la distribution de possibilité induite est compatible c'est-à-dire $\Pi(\alpha \wedge \beta) > \Pi(\alpha \wedge \neg\beta)$ pour chaque règle des défauts $\alpha \rightarrow \beta$. Ce dernier point revient à vérifier que pour chaque règle $\alpha \rightarrow \beta$, on a $\Upsilon \models_{MSP} \alpha \rightarrow b$. Heureusement, ceci peut être également effectué en temps polynomial dans notre approche.

Analysons maintenant son comportement quand une théorie des défauts déjà compilée est modifiée soit en supprimant une règle des défauts soit en rajoutant une nouvelle. Autrement dit, quel est le coût sous-jacent ?

– Considérons dans un premier temps le cas où l'on supprime une règle des défauts $\alpha_i \rightarrow \beta_i$. Ceci revient à conditionner $CompMSP(\Upsilon, L)$ par A_i et appliquer par la suite l'algorithme 8.2. Ces deux étapes peuvent être effectuées en temps polynomial et $CompMSP(\Upsilon, L)$ appartient toujours au langage L. Par conséquent, la suppression d'une formule ne demande aucune recompilation.

  Exemple 44 *Considérons encore la politique de sécurité donnée par l'exemple 40 et mettons la à jour en supprimant la règle stipulant qu'en général, les chirurgiens sont autorisés à écrire les rapports chirurgicaux. Ceci revient à supprimer la règle des défauts* $s \rightarrow w$ *de* D. *Soit* Υ' *la théorie des défauts résultante de cette mise à jour.*

  La bonne nouvelle est qu'il est possible d'effectuer une inférence polynomial à partir

de la nouvelle théorie sans aucun coût de recompilation. En effet, supprimer une règle d'une théorie des défauts déjà compilée revient à remplacer la formule correspondante $\neg s \vee w \vee A_1$ *par* $\top$*. Ceci peut être effectué en remplaçant* A_1 *par* $\top$ *ou d'une manière équivalente en conditionnant* $CompMSP(\Upsilon, DNNF)$ *sur* A_1*. Donc,*
$CompMSP(\Upsilon', DNNF) \equiv CompMSP(\Upsilon, DNNF)|A_1 \equiv [s \wedge [(w \wedge ((A_0 \wedge \neg d) \vee (A_2 \wedge d))) \vee (\top \wedge \neg w \wedge (A_0 \vee d))]] \vee [\neg s \wedge (A_2 \vee \neg d \vee \neg w)] \equiv [s \wedge [(w \wedge ((A_0 \wedge \neg d) \vee (A_2 \wedge d))) \vee (\neg w \wedge (A_0 \vee d))]] \vee [\neg s \wedge (A_2 \vee \neg d \vee \neg w)]$.
Notons que cette opération s'effectue en temps polynomial et génère une formule $DNNF$ *étant donné que le langage* $DNNF$ *satisfait le conditionnement.*
Maintenant, afin de déterminer la nouvelle stratification, on applique simplement l'algorithme 8.2 qui génère en temps polynomial la séquence (I_1, I_2) *où* $I_1 = \{2\}$ *et* $I_2 = \{0\}$.

– Quant au rajout d'une nouvelle règle $\alpha_s \rightarrow \beta_s$, rappelons tout d'abord le point suivant. On sait d'après [37] que si on prend deux formules ψ et χ appartenant à un langage cible de compilation L, on dispose d'un algorithme polynomial pour calculer une formule équivalente à leur conjonction $\psi \wedge \chi$, et qui appartient à L pour les langages $DNF, OBDD_<, IP$ et $MODS$. Donc dans le cas où $L \in \{DNF, OBDD_<, IP, MODS\}$, il suffit de compiler la formule $\neg\alpha_s \vee \beta_s \vee A_s$ vers L et d'effectuer par la suite en temps polynomial sa conjonction avec $CompMSP(\Upsilon, L)$. Ce coût est clairement beaucoup moins important que le coût correspondant à la recompilation d'une nouvelle théorie des défauts.

8.5 Conclusion

Dans ce chapitre, nous avons proposé une approche de compilation des théories des défauts. Cette compilation présente de nombreux avantages d'un point de vue calculatoire. En effet, la théorie des défauts compilée nous permet d'effectuer l'inférence MSP en temps polynomial. D'autre part, elle est paramétrée par n'importe quel langage cible de compilation d'où sa flexibilité. En outre, mettre à jour une théorie des défauts ne demande aucune recompilation dans le cas où l'on supprime une règle existante. Quant au rajout d'une nouvelle règle, le coût de recompilation est négligeable par rapport à certains langages cibles de compilation.

Chapitre 9

Inférence lexicographique à partir de bases partiellement préordonnées

9.1 Introduction

La flexibilité offerte par les bases de croyances partiellement préordonnées a motivé la définition de nouvelles approches basées sur la restauration telles que l'inférence relative à l'inclusion basée compatibles et l'inférence démocratique qui étendent l'inférence relative à la préférence pour l'inclusion, et les inférences possibilistes faible et forte qui généralisent l'inférence possibiliste.

Néanmoins, aucune extension n'a été proposée pour l'inférence lexicographique bien qu'elle soit dotée de propriétés intéressantes d'un point de vue théorique et pratique. Par exemple, elle ne souffre pas du problème de la noyade à l'inverse de l'inférence possibiliste. En outre, une étude psychologique réalisée dans le contexte du raisonnement par défaut a montré que l'inférence lexicographique présente de fortes similarités avec le raisonnement humain non-monotone. En particulier, les participants à ce test psychologique ont clairement manifesté une sensibilité à la majorité qui est au cœur de cette inférence [8].

Dans ce chapitre, nous proposons deux relations d'inférence basées sur la cardinalité à partir de bases de croyances partiellement préordonnées. Ces deux relations sont assez naturelles et étendent l'inférence lexicographique classique. La première, consiste à appliquer l'inférence lexicographique classique sur toutes les bases totalement préordonnées compatibles. Quant à la seconde, qui se trouve plus prudente, elle revient à appliquer l'inférence classique sur les sous-bases cohérentes qui sont préférées par rapport à une nouvelle relation de préférence lexicographique. L'intuition derrière cette nouvelle relation de préférence lexicographique est qu'une sous-base cohérente est préférée à une autre sous-base cohérente par rapport à une base de croyances partiellement préordonnées si et seulement si elle lui est préférée lexicographiquement de manière classique relativement à toute base de croyances stratifiées compatible. En outre, cette relation admet une définition équivalente qui ne nécessite pas le calcul de toutes les sous-bases stratifiées compatibles. Ce travail a été publié dans [100] et [101].

9.2 Inférence lexicographique basée-compatibles

La première inférence lexicographique à partir de bases de croyances partiellement préordonnées que nous proposons s'appuie uniquement sur la notion de bases de croyances totalement préordonnées compatibles.

Les sous-bases cohérentes préférées dans ce cas sont définies via la notion de bases de croyances totalement préordonnées compatibles avec la base de croyances en question. Plus précisément, une sous-base cohérente est préférée si et seulement s'il existe une base de croyances totalement préordonnées compatible où cette sous-base est lexicographiquement préférée de manière classique. Plus formellement, nous proposons la définition suivante :

Définition 133 *Soit $(\Sigma, \preceq)$ une base de croyances partiellement préordonnées et soit $B \in Cons(\Sigma)$. Alors B est dite sous-base cohérente préférée de $(\Sigma, \preceq)$ selon la* ***préférence lexicographique basée-compatibles****, ou* ***cmp-lexicographiquement préférée*** *en abrégé, si et seulement s'il existe une base de croyances totalement préordonnées $(\Sigma, \leq)$ compatible avec $(\Sigma, \preceq)$ telle que $B \in Min(Cons(\Sigma, \leq), <_{lex})$.*

En vue d'alléger l'écriture, $CmpLex(\Sigma, \preceq)$ dénotera dans ce qui suit l'ensemble de toutes les sous-bases cohérentes cmp-lexicographiquement préférées de $(\Sigma, \preceq)$:

$$CmpLex(\Sigma, \preceq) = \bigcup_{(\Sigma, \leq) \in Comp(\Sigma, \preceq)} Min(Cons(\Sigma, \leq), <_{lex}).$$

La proposition suivante stipule que toutes les sous-bases cmp-lexicographiquement préférées sont des sous-bases maximales cohérentes.

Proposition 10 *Soit $(\Sigma, \preceq)$ une base de croyances partiellement préordonnées. Alors :*

$$\forall B \in CmpLex(\Sigma, \preceq), B \in MaxCons(\Sigma).$$

Preuve.

Soit $B \in CmpLex(\Sigma, \preceq)$. Donc, il existe une base stratifiée $(\Sigma, \leq)$ compatible avec $(\Sigma, \preceq)$ telle que $B \in Min(Cons(\Sigma, \leq), <_{lex})$. Selon [9], B est une sous-base maximale cohérente de Σ.

■

Maintenant, l'inférence lexicographique basée-compatibles à partir de bases de croyances partiellement préordonnées revient à appliquer l'inférence classique à partir des sous-bases cmp-lexicographiquement préférées. Cette inférence est donnée par la définition suivante :

Définition 134 *Soit $(\Sigma, \preceq)$ une base de croyances partiellement préordonnées et soit ψ une formule propositionnelle. Alors, ψ est une conséquence relative à la préférence lexicographique basée-compatibles ou* ***conséquence cmp-lexicographique*** *de $(\Sigma, \preceq)$, noté $(\Sigma, \preceq) \vdash_{clex} \psi$, si et seulement si ψ est conséquence classique de chaque sous-base cohérente cmp-lexicographiquement préférée. Plus formellement :*

$$(\Sigma, \preceq) \vdash_{clex} \psi \ ssi \ \forall B \in CmpLex(\Sigma, \preceq), B \models \psi.$$

Cette relation d'inférence est assez naturelle. En effet, la notion de bases de croyances totalement préordonnées compatibles est très intuitive. De plus, elle a été utilisée, comme nous l'avons vu dans le chapitre 4, dans d'autres travaux, à l'instar de l'inférence relative

à la préférence de l'inclusion basée-compatibles proposée dans [21, 62], et aussi l'inférence possibiliste faible introduite dans [5].

Illustrons l'inférence cmp-lexicographique avec l'exemple suivant que nous allons considérer tout au long de ce chapitre.

Exemple 45 *Dans cet exemple, nous supposons les croyances suivantes :*
- C_1 : *En général, les quakers sont pacifistes :* $\neg Q \vee P$.
- C_2 : *En général, les républicains ne sont pas pacifistes :* $\neg R \vee \neg P$.
- C_3 :*En général, les écologues sont pacifistes :* $\neg E \vee P$.
- C_4 : *En général, les scientifiques sont logiques :* $\neg S \vee L$.
- C_5 : *Les écologues sont des scientifiques :* $\neg E \vee S$.
- C_6 : *De plus, on considère un individu qui est à la fois républicain, quaker et écologue :* $R \wedge Q \wedge S$.

Par ailleurs, nous supposons que les croyances C_2 et C_3 ont été fournies par la même source S_1 qui les considère comme étant égales d'un point de vue fiabilité. Quant aux croyances C_3 et C_4, elles sont issues d'une autre source S_2, qui considère que C_3 est plus prioritaire que C_4 car C_3 est relative à une classe qui se trouve une sous-classe de celle à laquelle se rapporte la croyance C_4. En outre, on ignore la relation de priorité entre les sources S_1 et S_2, et par conséquent, elles sont considérées comme étant incomparables en termes de fiabilité. Enfin, les croyances C_5 et C_6 sont plutôt certaines. De ce fait, elles se trouvent plus prioritaires que toutes les autres croyances.

Plus formellement, ces croyances sont représentées par la base de croyances partiellement préordonnées $(\Sigma, \preceq)$ telle que : $\Sigma = \{R \wedge Q \wedge S, \neg E \vee S, \neg Q \vee P, \neg R \vee \neg P, \neg E \vee P, \neg S \vee L\}$. Le préordre $\preceq$ entre ces croyances est décrit par la figure 9.1.

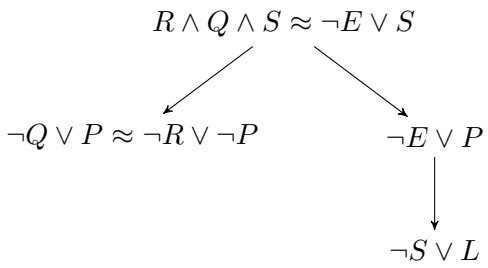

FIG. 9.1 – Le préordre partiel $\preceq$ sur Σ

Tout d'abord, la base de croyances partiellement préordonnées $(\Sigma, \preceq)$ admet cinq bases stratifiées (totalement préordonnées) compatibles : $(\Sigma, \leq^1), \ldots, (\Sigma, \leq^5)$ telles que :
- $(\Sigma, \leq^1) = (\{R \wedge Q \wedge S, \neg E \vee S\}, \{\neg Q \vee P, \neg R \vee \neg P\}, \{\neg E \vee P\}, \{\neg S \vee L\})$,
- $(\Sigma, \leq^2) = (\{R \wedge Q \wedge S, \neg E \vee S\}, \{\neg Q \vee P, \neg R \vee \neg P, \neg E \vee P\}, \{\neg S \vee L\})$,

– $(\Sigma, \leq^3) = (\{R \wedge Q \wedge S, \neg E \vee S\}, \{\neg E \vee P\}, \{\neg Q \vee P, \neg R \vee \neg P\}, \{\neg S \vee L\})$,
– $(\Sigma, \leq^4) = (\{R \wedge Q \wedge S, \neg E \vee S\}, \{\neg E \vee P\}, \{\neg S \vee L, \neg Q \vee P, \neg R \vee \neg P\})$,
– $(\Sigma, \leq^5) = (\{R \wedge Q \wedge S, \neg E \vee S\}, \{\neg E \vee P\}, \{\neg S \vee L\}, \{\neg Q \vee P, \neg R \vee \neg P\})$.

Dans ce cas, les sous-bases lexicographiquement préférées de chaque base sont comme suit :

$Min((\Sigma, \leq^1), <^1_{lex}) = \ldots = Min((\Sigma, \leq^5), <^5_{lex}) = \{R \wedge Q \wedge S, \neg E \vee S, \neg Q \vee P, \neg E \vee P, \neg S \vee L\}$.

Par conséquent, $CmpLex(\Sigma, \preceq) = \bigcup_{i=1}^{5} Min(Cons(\Sigma, \leq^i), <^i_{lex}) = \{R \wedge Q \wedge S, \neg E \vee S, \neg Q \vee P, \neg E \vee P, \neg S \vee L\}$.

Alors, on a par exemple $(\Sigma, \preceq) \vdash_{clex} P$*, ce qui reflète la plausibilité du fait que l'individu en question soit plutôt pacifiste, et ce qui est intuitivement attendu.*

D'un point de vue sémantique, nous introduisons tout d'abord une notion de préférence parmi les interprétations, et ce en regardant pour chaque interprétation si l'ensemble des formules qu'elle satisfait est cmp-lexicographiquement préféré.

Définition 135 *Soit ω une interprétation. Alors, ω est dite une interprétation préférée de Ω selon la* ***préférence lexicographique basée-compatibles****, ou* ***cmp-lexicographiquement préférée*** *en abrégé, ssi* $V(\omega, \Sigma) \in CmpLex(\Sigma, \preceq)$ *où* $V(\omega, \Sigma)$ *désigne l'ensemble de toutes les formules de Σ satisfaites par ω.*

Soit $CmpLex(\Omega)$ l'ensemble des interprétations cmp-lexicographiquement préférées de Ω.

Nous proposons alors une caractérisation sémantique de l'inférence cmp-lexicographique telle que décrite par la proposition suivante :

Proposition 11 *Soient $(\Sigma, \preceq)$ une base de croyances partiellement préordonnées et ψ une formule propositionnelle. Alors,*

$$(\Sigma, \preceq) \vdash_{clex} \psi \text{ ssi } \forall \omega \in CmpLex(\Omega), \omega \models \psi.$$

Preuve.

Ceci revient à montrer que : $CmpLex(\Omega) = \bigcup_{B \in CmpLex(\Sigma, \preceq)} Mod(B)$.

– Montrons que $CmpLex(\Omega) \subseteq \bigcup_{B \in CmpLex(\Sigma, \preceq)} Mod(B)$.

Soit $\omega \in CmpLex(\Omega)$. Par définition, $V(\omega, \Sigma) \in CmpLex(\Sigma, \preceq)$. De plus, il est claire que $\omega \in Mod(V(\omega, \Sigma))$. En conséquence, $\omega \in \bigcup_{B \in CmpLex(\Sigma, \preceq)} Mod(B)$.

– Montrons que $\bigcup_{B \in CmpLex(\Sigma, \preceq)} Mod(B) \subseteq CmpLex(\Omega)$.
Soit $\omega \in Mod(B)$ tel que $B \in CmpLex(\Sigma, \preceq)$. Selon la proposition 10, $B \in MaxCons(\Sigma, \preceq)$ ce qui implique que $V(\omega, \Sigma) = B$. Par conséquent, $\omega \in CmpLex(\Omega)$.

■

9.3 Préférence lexicographique partielle entre sous-bases cohérentes

Nous proposons une nouvelle relation de préférence lexicographique entre les sous-bases cohérentes d'une base de croyances partiellement préordonnées.

Étant donnée $(\Sigma, \preceq)$ une base de croyances partiellement préordonnées, nous considérons tout d'abord une partition de Σ comme suit :

Définition 136 *La partition de Σ est donnée par $\Sigma = E_1 \cup \ldots \cup E_n$ $(n \geq 1)$ tel que :*

1. *$\forall i$, $1 \leq i \leq n$, nous avons $\forall \varphi, \varphi' \in E_i : \varphi \approx \varphi'$;*
2. *$\forall i$, $1 \leq i \leq n$, $\forall j$, $1 \leq j \leq n$ avec $i \neq j$, nous avons $\forall \varphi \in E_i$, $\forall \varphi' \in E_j : \varphi \not\approx \varphi'$.*

En d'autres termes, chaque sous ensemble E_i représente une classe d'équivalence de Σ par rapport à la relation $\approx$ qui est une relation d'équivalence.

Nous définissons, ensuite, une relation de préférence entre les classes d'équivalence E_i's, notée $\prec_s$, comme suit :

Définition 137 *Soient E_i et E_j deux classes d'équivalence de Σ par rapport à $\approx$. Alors, $E_i \prec_s E_j$ si et seulement si*

$$\exists \varphi \in E_i,\ \exists \varphi' \in E_j \text{ tel que } \varphi \prec \varphi'.$$

Clairement, comme tous les éléments d'une classe E_i sont égaux, la relation $\prec_s$ peut être définie d'une manière équivalente par :

$$E_i \prec_s E_j \text{ ssi } \forall \varphi \in E_i,\ \forall \varphi' \in E_j :\ \varphi \prec \varphi'.$$

De plus, $E_i \sim_s E_j$ si et seulement si ni $E_i \prec_s E_j$ ni $E_j \prec_s E_i$ ne sont vérifiés.

Nous proposons ensuite une nouvelle relation de préférence lexicographique entre les sous-bases cohérentes de $(\Sigma, \preceq)$, baptisée préférence lexicographique partielle ou préférence p-lexicographique en abrégé, de la manière suivante :

Définition 138 *Soient $(\Sigma, \preceq)$ une base de croyances partiellement préordonnées et A et B deux sous-bases cohérentes de Σ. Alors, A est dite* ***p-lexicographiquement préférée*** *à B, noté $A \preceq_{plex} B$, si et seulement si*

$$\forall i,\ 1 \leq i \leq n :\ si\ |E_i \cap B| > |E_i \cap A|\ alors\ \exists j,\ 1 \leq j \leq n\ tel\ que\ |E_j \cap A| > |E_j \cap B|\ et\ E_j \prec_s E_i.$$

Autrement dit, A est préférée à B si pour toute classe d'équivalence qui contient plus d'élements de B que de A, il existe une classe d'équivalence qui contient plus d'éléments de A que de B, et qui lui est plus prioritaire.

Voyons ce que donne cette définition sur notre exemple.

Exemple 46 *Commençons par identifier les classes d'équivalence de Σ par rapport à l'égalité $\approx$. En fait, on a $\Sigma = E_1 \cup E_2 \cup E_3 \cup E_4$ où :*

- $E_1 = \{R \wedge Q \wedge S, \neg E \vee S\}$,
- $E_2 = \{\neg Q \vee P, \neg R \vee \neg P\}$,
- $E_3 = \{\neg E \vee P\}$,
- $E_4 = \{\neg S \vee L\}$.

De plus, on a $E_1 \prec_s E_2$, $E_1 \prec_s E_3$ et $E_3 \prec_s E_4$ comme le montre la figure 9.2.

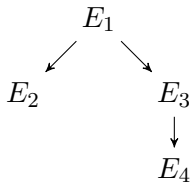

FIG. 9.2 – Le préordre partiel $\prec_s$ sur E_i's

Parmi les sous-bases cohérentes de Σ, considérons par exemple A, B et C telles que :

- $A = \{R \wedge Q \wedge S, \neg E \vee S, \neg E \vee P, \neg Q \vee P, \neg S \vee L\}$,
- $B = \{R \wedge Q \wedge S, \neg E \vee S, \neg R \vee \neg P, \neg S \vee L\}$,
- $C = \{R \wedge Q \wedge S, \neg E \vee S, \neg E \vee P, \neg S \vee L\}$,

Alors, en comparant A et B, on obtient $A \prec_{plex} B$. En effet, pour toute classe E_i, $1 \leq i \leq 4$, on a $|E_i \cup A| \geq |E_i \cup B|$ ce qui implique que $A \preceq_{plex} B$. De plus, on a $B \not\preceq_{plex} A$ car il existe une classe, à savoir E_3, contenant plus d'éléments de A que de B alors qu'il n'existe aucune classe qui lui est plus prioritaire qui vérifie l'inverse.

Maintenant, si on compare B et C, on constate qu'elles sont incomparables : $B \sim_{plex} C$. Ceci découle du fait que ni $B \preceq_{plex} C$ n'est vérifié ni $C \preceq_{plex} B$ ne l'est. Par exemple, $C \not\preceq_{plex} B$ car on a la classe E_2 qui contient un élément de B et aucun élément de C tandis que la seule classe qui est plus prioritaire que E_2, c'est-à-dire la classe E_1, ne contient pas plus d'éléments de C que de B. D'une manière similaire, on vérifie que $B \not\preceq_{plex} C$.

La proposition suivante décrit quelques propriétés de la relation de préférence $\preceq_{plex}$. En fait, cette dernière définit un préordre partiel sur l'ensemble des sous-bases cohérentes. De plus elle vérifie la monotonie ainsi que la monotonie stricte dans le sens où une base cohérente est préférée à ses sous-bases.

Proposition 12 *Soit $(\Sigma, \preceq)$ une base de croyances partiellement préordonnées. Alors,*

1. *$\preceq_{plex}$ est un préordre partiel sur l'ensemble des sous-bases cohérentes de Σ.*
2. *$\preceq_{plex}$ vérifie la propriété de monotonie, c'est-à-dire :*

$$\forall A, B \subseteq \Sigma, \text{ si } B \subseteq A \text{ alors } A \preceq_{plex} B.$$

3. *$\preceq_{plex}$ vérifie la propriété de monotonie stricte, c'est-à-dire :*

$$\forall A, B \subseteq \Sigma, \text{ si } B \subset A \text{ alors } A \prec_{plex} B.$$

Preuve.

1. $\preceq_{plex}$ est un préordre partiel sur $Cons(\Sigma)$:
   - $\preceq_{plex}$ **est une relation réflexive :**
     Soit $A \in Cons(\Sigma)$. Il est clair que, $\forall i, 1 \leq i \leq n : |E_i \cap A| = |E_i \cap A|$.
     Donc, $A \preceq_{plex} A$ c'est-à-dire $\preceq_{plex}$ est une relation réflexive.
   - $\preceq_{plex}$ **n'est pas une relation antisymétrique :**
     Considérons le contre exemple suivant : $\Sigma = \{\varphi, \delta, \psi, \chi\}$ avec $\varphi \approx \chi$ et $\delta \approx \psi$. Soient $A, B \in Cons(\Sigma)$ tels que $A = \{\varphi, \delta\}$ et $B = \{\psi, \chi\}$. On peut facilement voir que $A \preceq_{plex} B$ et $B \preceq_{plex} A$ mais A n'est pas identique à B. Par conséquent, $\preceq_{plex}$ n'est pas une relation antisymétrique.
   - $\preceq_{plex}$ **est une relation transitive :**
     Soient $A, B, C \in Cons(\Sigma)$. Supposons que $A \preceq_{plex} B$ et $B \preceq_{plex} C$ et montrons que $A \preceq_{plex} C$. Raisonnons par l'absurde en supposant que $A \preceq_{plex} C$ n'est pas vérifié. Ceci implique qu'il existe i, $1 \leq i \leq n$ tel que $|E_i \cap C| > |E_i \cap A|$ et $\nexists j, 1 \leq j \leq n$ tel que $E_j \prec_s E_i$ et $|E_j \cap A| > |E_i \cap C| \ldots$ **(Hypothèse 1)** En fait, nous distinguons deux cas :
     - **Cas 1 :** $|E_i \cap B| > |E_i \cap A|$:
       Soit $F = \{E_j, 1 \leq j \leq n : |E_j \cap A| > |E_j \cap B|$ et $E_j \prec_s E_i\}$. Comme $A \preceq_{plex} B$

alors $F \neq \emptyset$. En particulier, soit $E_j \in Min(F, \prec_s)$. Maintenant, selon l'hypothèse 1, $|E_j \cap C| \geq |E_j \cap A|$ donc $|E_j \cap C| > |E_j \cap B|$. Ensuite, étant donné que $B \preceq_{plex} C$, soit $E_k, 1 \leq k \leq n$ tel que $E_k \prec_s E_j$ et $|E_k \cap B| > |E_k \cap C|$. Toujours, selon l'hypothèse 1, $|E_k \cap C| \geq |E_k \cap A|$ donc $|E_k \cap B| > |E_k \cap A|$. Puisque $A \preceq_{plex} B$, il doit exister $l, 1 \leq l \leq n$ tel que $E_l \prec_s E_k$ et $|E_l \cap A| > |E_l \cap B|$. Clairement, nous avons $E_l \prec_s E_j$ et $E_l \prec_s E_i$ (par transitivité de la relation $\prec_s$). Finalement, nous déduisons que $E_l \in F$ et $E_l \prec_s E_j$ mais ceci contredit le fait que $E_j \in Min(F, \prec_s)$.

– **Cas 2 :** $|E_i \cap B| \leq |E_i \cap A|$:
Évidemment, ceci implique que $|E_i \cap B| < |E_i \cap C|$. Soit $G = \{E_j, 1 \leq j \leq n : |E_j \cap B| > |E_j \cap C|$ et $E_j \prec_s E_i\}$. Comme $B \preceq_{plex} C$, $G \neq \emptyset$. En particulier, soit $E_j \in Min(G, \prec_s)$. Selon l'hypothèse 1, $|E_j \cap C| \geq |E_j \cap A|$ donc $|E_j \cap B| > |E_j \cap A|$. Ensuite, étant donné que $A \preceq_{plex} B$, soit $E_k, 1 \leq k \leq n$ tel que $E_k \prec_s E_j$ et $|E_k \cap A| > |E_k \cap B|$. Encore une fois, selon l'hypothèse 1, $|E_k \cap C| \geq |E_k \cap A|$ donc $|E_k \cap C| > |E_k \cap B|$. Comme $B \preceq_{plex} C$, il doit exister $l, 1 \leq l \leq n$ tel que $E_l \prec_s E_k$ et $|E_l \cap B| > |E_l \cap C|$. Donc, on a $E_l \prec_s E_j$ et $E_l \prec_s E_i$ (par transitivité de $\prec_s$). Par conséquent, on déduit que $E_l \in G$ et $E_l \prec_s E_j$ ce qui est incohérent avec le fait que $E_j \in Min(G, \prec_s)$.

Enfin, $\forall i, 1 \leq i \leq n$: si $|E_i \cap C| > |E_i \cap A|$ alors $\exists j, 1 \leq j \leq n$ tel que $E_j \prec_s E_i$ et $|E_j \cap A| > |E_i \cap C|$. Ceci signifie que $A \preceq_{plex} C$.

2. $\preceq_{plex}$ satisfait la propriété de **monotonie** : Soient $A, B \in Cons(\Sigma)$. $B \subseteq A$ implique $\forall i$, $1 \leq i \leq n$: $|E_i \cap A| \geq |E_i \cap B|$. Donc, selon la définition de $\preceq_{plex}$, on obtient $A \preceq_{plex} B$.
3. $\preceq_{plex}$ satisfait la propriété de **monotonie stricte** : Soient $A, B \in Cons(\Sigma)$. $B \subset A$ implique $\forall i$, $1 \leq i \leq n : |E_i \cap A| \geq |E_i \cap B|$ et que $\forall j$, $1 \leq j \leq n : |E_j \cap A| > |E_j \cap B|$. Donc, selon la définition de $\prec_{plex}$, on obtient $A \prec_{plex} B$.

■

L'ordre partiel strict associé à $\preceq_{plex}$, noté $\prec_{plex}$, est défini par :

$$A \prec_{plex} B \textit{ si et seulement si } A \preceq_{plex} B \textit{ et } B \not\preceq_{plex} A.$$

L'égalité correspondante, notée $\approx_{plex}$, est définie par :

$$A \approx_{plex} B \textit{ si et seulement si } A \preceq_{plex} B \textit{ et } B \preceq_{plex} A.$$

La proposition suivante fournit une caractérisation de l'égalité $\approx_{plex}$:

Proposition 13 *Soit $(\Sigma, \preceq)$ une base de croyances partiellement préordonnées et soient A et B deux sous-bases cohérentes de Σ. Alors,*

$$A \approx_{plex} B \textit{ ssi } \forall i, 1 \leq i \leq n : |E_i \cap B| = |E_i \cap A|.$$

Preuve.

- Par définition de la relation $\preceq_{plex}$, nous avons $\forall i, 1 \leq i \leq n : |E_i \cap A| = |E_i \cap B|$ implique que $A \preceq_{plex} B$ et $B \preceq_{plex} A$ c'est-à-dire $A \approx_{plex} B$.
- Montrons maintenant que $A \approx_{plex} B$ implique que $\forall i, 1 \leq i \leq n : |E_i \cap A| = |E_i \cap B|$. Raisonnons par l'absurde en supposant que $A \approx_{plex} B$ et $\exists i, 1 \leq i \leq n : |E_i \cap A| \neq |E_i \cap B|$. On distingue deux cas :
  - **Cas 1 :** $A \approx_{plex} B$ et $\exists i, 1 \leq i \leq n$ tel que $|E_i \cap B| > |E_i \cap A|$:
    Soit $F = \{E_i, 1 \leq i \leq n$ tel que $|E_i \cap B| > |E_i \cap A|\}$. $F \neq \emptyset$ donc soit $E_i \in Min(F, \prec_s)$. Puisque $A \preceq_{plex} B$, $\exists j, 1 \leq j \leq n$ tel que $E_j \prec_s E_i$ et $|E_j \cap A| > |E_j \cap B|$. En outre, comme $B \preceq_{plex} A$, $\exists k$ tel que $E_k \prec_s E_j$ et $|E_k \cap B| > |E_k \cap A|$. Il est clair que $E_k \in F$ et $E_k \prec_s E_i$ (par transitivité de $\prec_s$) ce qui contredit le fait que $E_i \in Min(F, \prec_s)$.
  - **Cas 2 :** $A \approx_{plex} B$ et $|E_i \cap A| > |E_i \cap B|$: preuve similaire.

■

La préférence lexicographique entre les sous-bases cohérentes d'une base de croyances partiellement préordonnées recouvre ou étend la préférence lexicographique classique entre les sous-bases cohérentes d'une base de croyances totalement préordonnées. Autrement dit, dans le cas de bases de croyances stratifiées, les deux relations de préférence coïncident comme le montre la proposition suivante :

Proposition 14 *Soit une base de croyances totalement préordonnées* $(\Sigma, \leq) = (S_1, \ldots, S_m)$, *soient* A *et* B *deux sous-bases cohérentes de* Σ. *Alors,*

1. $A <_{lex} B$ *ssi* $A \prec_{plex} B$.
2. $A =_{lex} B$ *ssi* $A \approx_{plex} B$.

Preuve.

Notons que dans le cadre d'une base de croyances totalement pré-ordonnées $(\Sigma, \leq) = (S_1, \ldots, S_m)$, les classes d'équivalence E_i's ne sont rien d'autres que les strates S_i's et $S_i \prec_s S_j$ si et seulement si $i < j$.

1. $A <_{lex} B$ si et seulement si $A \prec_{plex} B$:
   - Montrons que $A <_{lex} B$ implique que $A \prec_{plex} B$:
     Soit $A <_{lex} B$. Par définition, $\exists i$, $1 \leq i \leq m$ tel que $|S_i \cap A| > |S_i \cap B|$ et $\forall j$, $j < i$, $|S_j \cap B| = |S_j \cap A|$.
     Donc, $\forall k, 1 \leq k \leq m$, si $|S_k \cap B| > |S_k \cap A|$ alors $k > i$ c'est-à-dire $\exists i$ tel que $|S_i \cap A| > |S_i \cap B|$ avec $i < k$.
     Ceci implique que $A \preceq_{plex} B$.
     De plus, $\exists i, 1 \leq i \leq m$ tel que $|S_i \cap A| > |S_i \cap B|$ et $\nexists j, 1 \leq j \leq m$ tel que $|S_j \cap B| > |S_j \cap A|$ et $j < i$ ce qui signifie que $B \not\preceq_{plex} A$. Par conséquent, $A \prec_{plex} B$.

– Montrons maintenant que $A \prec_{plex} B$ implique que $A <_{lex} B$:
Soit $A \prec_{plex} B$. Alors, $\exists i, 1 \leq i \leq m$ tel que $|S_i \cap A| > |S_i \cap B|$ and $\forall j, 1 \leq j \leq m$, si $E_j \prec_s E_i$ alors $|S_j \cap B| \leq |S_j \cap A|$ car $B \not\prec_{plex} A$. Ceci signifie que $A <_{lex} B$.

2. $A =_{lex} B$ si et seulement si $A \approx_{plex} B$:
Ceci découle directement de la définition de l'inférence lexicographique et de la proposition 13.

■

Enfin, l'intuition derrière la préférence p-lexicographique proposée dans cette section réside dans le fait qu'une sous-base A est p-lexicographiquement préférée à une sous-base B si et seulement si A est lexicographiquement préférée à B de manière classique par rapport à toute base de croyances stratifiées compatible avec la base partiellement préordonnées considérée. Il est de même pour l'égalité p-lexicographique. En effet, nous prouvons les deux propositions suivantes où la première concerne l'égalité.

Proposition 15 *Soit $(\Sigma, \preceq)$ une base de croyances partiellement préordonnées et soient A et B deux sous-bases cohérentes de Σ. Alors, $A \approx_{plex} B$ si et seulement si $A =_{lex} B$ par rapport à n'importe quelle base de croyances totalement préordonnées $(\Sigma, \leq)$ compatible avec $(\Sigma, \preceq)$.*

Preuve.

1. $A \approx_{plex} B$ implique $A =_{lex} B$ pour toute $(\Sigma, \leq) \in Comp(\Sigma, \preceq)$:
Soit $(\Sigma, \leq) = S_1 \cup \ldots \cup S_m$ une base de croyances totalement pré-ordonnées compatible avec $(\Sigma, \preceq)$.
$\forall i, 1 \leq i \leq m : S_i = \bigcup_{j \in X} E_j$ où $X \subseteq \{1, \ldots, n\}$. Donc $|S_i \cap A| = |(\bigcup_{j \in X} E_j) \cap A| = |\bigcup_{j \in X}(E_j \cap A)| = \sum_{j \in X} |E_j \cap A|$ puisque E_j's sont disjoints. De manière similaire, on obtient $|S_i \cap B| = \sum_{j \in X} |E_j \cap B|$. Selon la proposition 13, $A \approx_{plex} B$ si et seulement si $\forall i, 1 \leq i \leq n : |E_i \cap A| = |E_i \cap B|$. Donc, on conclut que $\forall j, 1 \leq j \leq m : |S_i \cap A| = |S_i \cap B|$ c'est à dire $A =_{lex} B$.

2. $A =_{lex} B$ pour toute $(\Sigma, \leq) \in Comp(\Sigma, \preceq)$ implique $A \approx_{plex} B$:
Raisonnons par l'absurde en supposant que $A =_{lex} B$ pour toute $(\Sigma, \leq) \in Comp(\Sigma, \preceq)$ et que $A \not\approx_{plex} B$.
Selon la proposition 13, $A \not\approx_{plex} B$ signifie que $\exists i, 1 \leq i \leq n$ tel que $|E_i \cap A| \neq |E_i \cap B|$.
Considérons une base de croyances totalement pré-ordonnées $(\Sigma, \leq) = S_1 \cup \ldots \cup S_m$ compatible avec $(\Sigma, \preceq)$ et supposons que $(\Sigma, \leq)$ met la classe E_i dans une strate à part S_k $(1 \leq k \leq m)$. Dans ce cas, il est claire que $A \neq_{lex} B$ ce qui contredit le fait que $A =_{lex} B$ pour toute $(\Sigma, \leq) \in Comp(\Sigma, \preceq)$. En conséquence, on déduit que $A \approx_{plex} B$.

■

Quant à la proposition suivante, elle dit qu'une sous-base cohérente A est p-lexicographiquement préférée à une autre sous-base cohérente B si et seulement si A est lexicographiquement préférée à B par rapport à toute base totalement préordonnée compatible.

Proposition 16 *Soit $(\Sigma, \preceq)$ une base de croyances partiellement préordonnées et soient A et B deux sous-bases cohérentes de Σ. Alors, $A \prec_{plex} B$ si et seulement si $A <_{lex} B$ par rapport à n'importe quelle base de croyances totalement préordonnées $(\Sigma, \leq)$ compatible avec $(\Sigma, \preceq)$.*

Preuve.

1. $A \prec_{plex} B$ implique que $A <_{lex} B$:
   Nous rappelons que $A \prec_{plex} B$ si et seulement $A \preceq_{plex} B$ et $B \not\preceq_{plex} A$.
   On peut distinguer deux cas :
   - **Cas 1 :** $\nexists i, 1 \leq i \leq n$ tel que $|E_i \cap B| > |E_i \cap A|$ c'est à dire $\forall i$, $1 \leq i \leq n$, $|E_i \cap B| \leq |E_i \cap A|$.
     D'abord, il est à noter que $\forall j$, $1 \leq j \leq m$, on a $S_j = \bigcup_{i \in X} E_i$ où $X \subseteq \{1, \ldots, n\}$. Donc, $|S_j \cap B| = |(\bigcup_{i \in X} E_i) \cap B| = |\bigcup_{i \in X}(E_i \cap B)| = \sum_{i \in X} |E_i \cap B|$ puisque les E_i's sont par définition disjoints. Par analogie, on obtient $|S_j \cap A| = \sum_{i \in X} |E_i \cap A|$. Donc, selon l'hypothèse de ce cas, on conclut que : $\sum_{i \in X} |E_i \cap B| \leq \sum_{i \in X} |E_i \cap A|$ c'est à dire $|S_j \cap B| \leq |S_j \cap A|$.
     D'autre part, $B \not\preceq_{plex} A$ c'est à dire $\exists k, 1 \leq k \leq n$ tel que $|E_k \cap A| > |E_k \cap B|$ et $\nexists l, 1 \leq l \leq n$ tel que $E_l \prec_s E_k$ et $|E_l \cap B| > |E_l \cap A|$.
     Maintenant, soit S_i $(1 \leq i \leq m)$ la strate contenant la classe E_k qui vérifie la propriété précédente c'est à dire $S_i = E_k \cup \bigcup_{j \in Y} E_j$ où $Y \subseteq \{1, \ldots, n\}$. Donc, $|S_i \cap B| = |(\bigcup_{j \in Y} E_j \cup E_k) \cap B| = |\bigcup_{j \in Y}(E_j \cap B) \cup (E_k \cap B)| = \sum_{j \in Y} |E_j \cap B| + |E_k \cap B|$ car les E_j's sont par définition disjoints.
     D'une manière similaire, on déduit que $|S_i \cap A| = \sum_{j \in Y} |E_j \cap A| + |E_k \cap A|$. Clairement, $|S_i \cap B| < |S_i \cap A|$.
     Donc, d'une part on a $\forall j, 1 \leq j \leq m : |S_j \cap B| \leq |S_j \cap A|$ et d'autre part on a $\exists i, 1 \leq i \leq m : |S_i \cap B| < |S_i \cap A|$. Par conséquent, $A <_{lex} B$.
   - **Cas 2 :** $\exists k, 1 \leq k \leq n$ tel que $|E_k \cap B| > |E_k \cap A|$.
     Soit S_i la première strate (avec le petit i) contenant une classe E_k vérifiant la propriété précédente. Comme $A \prec_{plex} B$, il doit exister une classe E_p telle que $|E_p \cap A| > |E_p \cap B|$ et $E_p \prec_s E_k$.
     Donc, E_p doit être placée dans une strate S_l plus prioritaire que la strate S_i contenant E_k c'est à dire $p < i$. Ceci implique que $i \neq 1$. Ensuite, $\forall j$, $1 \leq j < i$: S_j est construite à partir de E_r's telles que $|E_r \cap B| \leq |E_r \cap A|$ par construction de S_i. Donc $\forall j$, $1 \leq j < i : |S_j \cap B| \leq |S_j \cap A|$. En particulier, $|S_l \cap B| < |S_l \cap A|$. Par conséquent, on déduit que $A <_{lex} B$.

2. $A <_{lex} B$: pour toute $(\Sigma, \preceq) \in Comp(\Sigma, \preceq)$ implique $A \prec_{plex} B$:
Raisonnons par l'absurde et supposons que $A <_{lex} B$ pour toute $(\Sigma, \preceq) \in Comp(\Sigma, \preceq)$ et que $A \not\prec_{plex} B$.
$A \not\prec_{plex} B$ si et seulement si $A \not\preceq_{plex} B$ ou $A \approx_{plex} B$. Donc, on peut distinguer deux cas :
   (a) **Cas 1 :** $A \approx_{plex} B$:
   Selon la proposition 15, $A \approx_{plex} B$ implique que $A =_{lex} B$ pour toute base compatible ce qui est clairement incohérent avec l'hypothèse disant que $A <_{lex} B$ pour toute $(\Sigma, \preceq) \in Comp(\Sigma, \preceq)$.
   (b) **Cas 2 :** $A \not\preceq_{plex} B$:
   $A \not\preceq_{plex} B$ signifie que $\exists i, 1 \leq i \leq n$ tel que $|E_i \cap B| > |E_i \cap A|$ et $\forall j, 1 \leq j \leq n$, $E_j \prec_s E_i$ implique que $|E_j \cap B| \geq |E_j \cap A|$.
   Soit $J = \{j, 1 \leq j \leq n \text{ tel que } E_j \prec_s E_i\}$.
   Soit une sous-base stratifiée $(\Sigma, \leq) = (S_1, \ldots, S_k, \ldots, S_m)$ compatible avec $(\Sigma, \preceq)$ telle que :
      i. $S_k = E_i$
      ii. $\forall l, 1 \leq l < k$, $S_l = \bigcup_{p \in X} E_p$ où $X \subseteq J$.

   Par conséquent, $|S_k \cap B| = |E_i \cap B| > |E_i \cap A| = |S_k \cap A|$.
   En outre, $\forall l, 1 \leq l < k$, $|S_l \cap B| = |(\bigcup_{p \in X} E_p) \cap B| = |\bigcup_{p \in X} (E_p \cap B)| = \sum_{p \in X} |E_p \cap B|$.
   De la même manière, on obtient $|S_l \cap A| = \sum_{p \in X} |E_p \cap A|$.
   Par ailleurs, $\forall j \in J$, $|E_j \cap B| \geq |E_j \cap B|$ car $E_j \prec_s E_i$ et $A \not\preceq_{plex} B$.
   Donc, $\sum_{p \in X} |E_p \cap B| \geq \sum_{p \in X} |E_p \cap A|$ d'où $|S_l \cap B| \geq |S_l \cap A|$ quelque soit l avec $1 \leq l < k$.
   Par conséquent, $B <_{lexA}$ par rapport à la base stratifiée compatible $(\Sigma, \preceq)$. Néanmoins, ceci contredit l'hypothèse qui dit que $A <_{lex} B$ pour toute $(\Sigma, \preceq) \in Comp(\Sigma, \preceq)$.

■

9.4 Inférence lexicographique basée sur les sous-bases cohérentes préférées

Nous définissons dans cette section une inférence lexicographique à partir de bases de croyances partiellement préordonnées qui est basée sur les sous-bases cohérentes préférées selon le préordre $\preceq_{plex}$ introduit précédemment.

Définition 139 *Soit* $(\Sigma, \preceq)$ *une base de croyances partiellement préordonnées et soit* ψ *une formule. Alors* ψ *est une* ***conséquence p-lexicographique*** *de* $(\Sigma, \preceq)$*, noté* $(\Sigma, \preceq) \vdash_{plex} \psi$*, ssi* ψ *est conséquence classique de chaque sous-base cohérente de* $(\Sigma, \preceq)$ *préférée selon* $\prec_{plex}$*, c'est-à-dire :*

$$(\Sigma, \preceq) \vdash_{plex} \psi \text{ ssi } \forall B \in Min(Cons(\Sigma, \preceq), \prec_{plex}),\ B \models \psi.$$

A l'image de la préférence lexicographique entre les sous-bases cohérentes d'une base de croyances stratifiées, chaque sous-base cohérente préférée selon $\prec_{plex}$ est une sous-base cohérente maximale pour l'inclusion ensembliste.

Proposition 17 *Soit* $(\Sigma, \preceq)$ *une base de croyances partiellement préordonnées. Alors,*

$$\forall B \in Min(Cons(\Sigma, \prec), \prec_{plex}), B \in MaxCons(\Sigma).$$

Preuve.

Étant donnée une sous-base $B \in Min(Cons(\Sigma, \preceq), \prec_{plex})$, supposons que $B \notin MaxCons(\Sigma)$. Donc, il existe une sous-base $B' \in Cons(\Sigma)$ telle que $B \subset B'$. Selon la proposition 12, la relation $\prec_{plex}$ vérifie la monotonie stricte si bien que $B' \prec_{plex} B$ ce qui contredit le fait que $B \in Min(Cons(\Sigma, \preceq), \preceq_{plex})$. Par conséquent, on déduit que $B \in MaxCons(\Sigma)$.

■

Exemple 47 *L'ensemble des sous-bases maximales cohérentes de* Σ *contient trois éléments :* $MaxCons(\Sigma) = \{A, B, C\}$ *avec*

- $A = \{R \wedge Q \wedge S, \neg E \vee S, \neg E \vee P, \neg Q \vee P, \neg S \vee L\}$,
- $B = \{R \wedge Q \wedge S, \neg E \vee S, \neg R \vee \neg P, \neg S \vee L\}$,
- $C = \{\neg E \vee S, \neg R \vee \neg P, \neg E \vee P, \neg Q \vee P, \neg S \vee L\}$.

Dans ce cas, on peut facilement voir que $A \prec_{plex} C$ *et que* $B \prec_{plex} C$*. En outre, on a déduit dans l'exemple 46 que* $A \prec_{plex} B$*.*

De ce fait, on conclut que $Min(Cons(\Sigma, \preceq), \prec_{plex}) = \{A\}$*. Par conséquent, l'inférence p-lexicographique à partir de* $(\Sigma, \preceq)$ *revient à une inférence classique à partir de la base classique* A*. Par exemple, comme* $A \models P$*, on peut déduire que* $(\Sigma, \preceq) \vdash_{plex} P$*.*

Toutes les conclusions obtenues par cette inférence sont également obtenues par l'inférence lexicographique basée sur les compatibles donnée par la définition 134 comme le montre la proposition suivante :

Proposition 18 *Soit* $(\Sigma, \preceq)$ *une base de croyances partiellement préordonnées et soit* ψ *une formule propositionnelle. Alors,*

$$(\Sigma, \preceq) \vdash_{plex} \psi \textit{ implique } (\Sigma, \preceq) \vdash_{clex} \psi.$$

Preuve.

Par définition, $(\Sigma, \preceq) \vdash_{plex} \psi$ ssi $\forall B \in Min(Cons(\Sigma, \preceq), \prec_{plex}) : B \models \psi$ et $(\Sigma, \preceq) \vdash_{clex} \psi$ ssi $\forall B \in \bigcup_{(\Sigma,\leq) \in Comp(\Sigma,\preceq)} Min(Cons(\Sigma, \leq), <_{lex}), B \models \psi$.

Donc ceci revient à montrer que

$$\bigcup_{(\Sigma,\leq) \in Comp(\Sigma,\preceq)} Min(Cons(\Sigma, \leq), <_{lex}) \subseteq Min(Cons(\Sigma, \preceq), \prec_{plex}).$$

Soit $(\Sigma, \leq) \in Comp(\Sigma, \preceq)$. Soit $A \in Min(Cons(\Sigma, \leq), <_{lex})$ et supposons que $A \notin Min(Cons(\Sigma, \preceq), \prec_{plex})$. Ceci signifie qu'il existe une sous base cohérente B de Σ telle que $B \prec_{plex} A$. Selon la proposition 16, on déduit que $B <_{lex} A$. Cependant, ceci est contradictoire avec le fait que $A \in Min(Cons(\Sigma, \leq), <_{lex})$.

Par conséquent, $A \in Min(Cons(\Sigma, \preceq), \prec_{plex})$ et $\bigcup_{(\Sigma,\leq) \in Comp(\Sigma,\preceq)} Min(Cons(\Sigma, \leq), <_{lex}) \subset Min(Cons(\Sigma, \preceq), \prec_{plex})$. ■

La réciproque de la proposition 18 n'est pas vraie comme le montre le contre-exemple suivant :

Exemple 48 *Soit* $(\Sigma, \preceq)$ *la base suivante :* $\Sigma = \{\alpha_1, \alpha_2, \beta_1, \beta_2, \beta_3, \gamma_1, \gamma_2\}$ *avec :*

- $\alpha_1 = a \wedge \neg b \wedge c$,
- $\beta_1 = \neg a \wedge \neg b \wedge c$,
- $\gamma_1 = b \wedge d$,
- $\alpha_2 = a \wedge \neg b \wedge d$,
- $\beta_2 = \neg a \wedge \neg b \wedge d$,
- $\gamma_2 = b \wedge e$,
- $\beta_3 = \neg a \wedge \neg b \wedge e$.

De plus, on a :

- $\alpha_1 \approx \gamma_1 \approx \gamma_2$,
- $\alpha_2 \approx \beta_1 \approx \beta_2 \approx \beta_3$.

Clairement, $MaxCons(\Sigma, \preceq) = \{A, B, C\}$ *tel que* $A = \{\alpha_1, \alpha_2\}$, $B = \{\beta_1, \beta_2, \beta_3\}$, $C = \{\gamma_1, \gamma_2\}$.

D'une part, $Min(Cons(\Sigma, \preceq), \prec_{plex}) = \{A, B, C\}$. *En fait, on a* $A \sim_{plex} B$, $A \sim_{plex} C$, $B \sim_{plex} C$.

D'autre part, $CmpLex(\Sigma, \preceq) = \{B, C\}$.

Donc, on déduit que $Min(Cons(\Sigma, \preceq), \prec_{plex}) \nsubseteq CmpLex(\Sigma, \preceq)$.

Nous donnons enfin une caractérisation sémantique de l'inférence p-lexicographique. Une relation de préférence entre deux interprétations est donnée dans ce cas comme suit :

Définition 140 *Soit $(\Sigma, \preceq)$ une base de croyances partiellement préordonnées. Soient ω et ω' deux interprétations. Alors, ω est dite* ***p-lexicographiquement préférée*** *à ω', noté $\omega \trianglelefteq_{plex} \omega'$, ssi $V(\omega, \Sigma) \preceq_{plex} V(\omega', \Sigma)$ où $V(\omega, \Sigma)$ est l'ensemble de formules de Σ satisfaites par ω.*

Soit $Min(\Omega, \trianglelefteq_{plex})$ l'ensemble des interprétations préférées de Ω par rapport à $\trianglelefteq_{plex}$. L'inférence p-lexicographique est caractérisée comme suit :

Proposition 19 *Soient $(\Sigma, \preceq)$ une base de croyances partiellement préordonnées et ψ une formule propositionnelle. Alors,*

$$(\Sigma, \preceq) \vdash_{plex} \psi \text{ ssi } \forall \omega \in Min(\Omega, \trianglelefteq_{plex}), \omega \models \psi.$$

Preuve.

Ceci revient à montrer que : $\bigcup_{B \in Min(Cons(\Sigma, \preceq), \prec_{plex})} Mod(B) = Min(\Omega, \trianglelefteq_{plex})$.

– Montrons que $\bigcup_{B \in Min(Cons(\Sigma, \preceq), \prec_{plex})} Mod(B) \subseteq Min(\Omega, \trianglelefteq_{plex})$:

Soit $\omega \in Mod(B)$ où $B \in Min(Cons(\Sigma, \preceq), \prec_{plex})$. Supposons que $\omega \notin Min(\Omega, \trianglelefteq_{plex})$. $\omega \notin Min(\Omega, \trianglelefteq_{plex})$ implique $\exists \omega' \in \Omega$ tel que $\omega' \trianglelefteq_{plex} \omega$ c'est à dire $V(\omega', \Sigma) \prec_{plex} V(\omega, \Sigma)$. Cependant, on sait que B est une sous-base maximale cohérente. Donc, ω ne satisfait aucune formule de Σ en dehors de B. Autrement dit, $V(\omega, \Sigma) = B$. Par conséquent, $B' \prec_{plex} B$ où $B' = V(\omega', \Sigma)$. Or, ceci contredit le fait que $B \in Min(Cons(\Sigma, \preceq), \prec_{plex})$. Donc, $\omega \in Min(\Omega, \trianglelefteq_{plex})$.

– Montrons maintenant que $Min(\Omega, \trianglelefteq_{plex}) \subseteq \bigcup_{B \in Min(Cons(\Sigma, \preceq), \prec_{plex})} Mod(B)$:

Soit $\omega \in Min(\Omega, \trianglelefteq_{plex})$. Ceci signifie qu'il n'existe pas $\omega' \in \Omega$ tel que $\omega' \trianglelefteq_{plex} \omega$. En particulier, pour chaque base $B \in Min(Cons(\Sigma, \preceq), \prec_{plex})$, pour chaque interprétation $\omega' \neq \omega$ qui est modèle de B, $\omega' \not\trianglelefteq_{plex} \omega$ c'est-à-dire $V(\omega', \Sigma) \not\prec_{plex} V(\omega, \Sigma)$. Cependant, $V(\omega', \Sigma) = B$ car B est une sous-base maximale cohérente. Donc, pour chaque $B \in Min(Cons(\Sigma, \preceq), \prec_{plex})$, $B \not\prec_{plex} V(\omega, \Sigma)$. Ceci signifie que $V(\omega, \Sigma) \in Min(Cons(\Sigma, \preceq), \prec_{plex})$.

Par conséquent, $\omega \in \bigcup_{B \in Min(Cons(\Sigma, \preceq), \prec_{plex})} Mod(B)$.

■

9.5 Conclusion

Dans ce chapitre, nous avons combiné les avantages de l'inférence lexicographique et ceux des bases de croyances partiellement préordonnées pour raisonner en présence d'incohérence. Nous avons présenté une inférence lexicographique à partir de bases de croyances partiellement préordonnées qui étend l'inférence lexicographique classique. Plus précisément, nous avons proposé deux relations d'inférence assez naturelles. La première consiste à appliquer l'inférence lexicographique classique sur toutes les bases totalement préordonnées compatibles. Quant à la seconde, elle s'articule autour de la définition d'un nouveau préordre partiel basé sur la cardinalité entre sous-bases cohérentes. Il s'agit ensuite d'appliquer l'inférence classique sur les sous-bases cohérentes préférées selon ce préordre. Les conclusions obtenues par la seconde inférence sont également obtenues par la première.

L'étude des relations d'inférence proposées dans ce chapitre en termes de complexité, de productivité et de propriétés de déduction fait entre autres l'objet du chapitre suivant.

Chapitre 10

Étude comparative

10.1 Introduction

Vue la flexibilité offerte par les bases de croyances partiellement préordonnées, différentes relations d'inférence existent actuellement pour raisonner à partir de telles bases. La question pertinente qui se pose alors est la suivante : étant donnée une application, quelle relation d'inférence choisir ? Une réponse naturelle à une telle question prend en considération trois critères clés, à savoir la complexité qui reflète le coût induit par cette relation d'inférence, ses propriétés logiques qui décrivent à quel point le comportement de cette relation est acceptable ou raisonnable, et enfin on a le critère de la prudence qui désigne le nombre de conclusions déductibles via cette relation d'inférence.

Néanmoins, et à la différence des relations d'inférence à partir de bases de croyances totalement préordonnées, les relations d'inférence à partir de bases de croyances partiellement préordonnées n'ont pas été suffisamment analysées par rapport à ces trois dimensions prépondérantes, et ce en dépit du caractère intéressant dont jouissent les bases de croyances partiellement préordonnées.

Nous proposons alors dans ce chapitre d'analyser et d'étudier un bon nombre de relations d'inférence à partir de bases de croyances partiellement préordonnées relativement aux facteurs de complexité, propriétés logiques et prudence. En particulier, les relations d'inférence sur lesquelles nous nous focalisons dans cette étude sont les deux relations d'inférence lexicographiques que nous avons proposées dans le chapitre 9, ainsi que les relations rappelées dans le chapitre 4, à savoir l'inférence possibiliste forte, l'inférence possibiliste faible, l'inférence démocratique et l'inférence relative à la préférence pour l'inclusion basée-compatibles. Les contributions de ce chapitre ont fait l'objet d'une publication à [16].

10.2 Résultats de complexité

La plupart des preuves liées à la complexité que nous fournissons dans ce chapitre s'appuient essentiellement sur les deux étapes suivantes :

- premièrement, exhiber un algorithme permettant de résoudre le problème étudié. On obtient ainsi une borne supérieure pour la complexité, et on parle de preuve d'appartenance à une classe de complexité.
- ensuite, affiner le résultat obtenu précédemment en trouvant une borne inférieure soit par preuve de complétude soit par preuve de difficulté par rapport à une autre classe de complexité différente de la borne supérieure. Dans tous les cas, cela revient à trouver une transformation polynomiale permettant de passer d'un problème dont la complexité est bien connue au dit-problème.

Avant de commencer notre étude, considérons d'abord l'inférence suivante et qui nous sera utile par la suite pour certaines preuves de complétude.

Définition 141 *Soit Σ une base de croyances, et soit ψ une formule propositionnelle. La formule ψ est dite* ***MC-conséquence*** *de Σ si et seulement si elle est déductible à partir de toutes les sous-bases maximales cohérentes de Σ :*

$$\Sigma \vdash_{mc} \psi \ ssi \ \forall B \in MaxCons(\Sigma),\ B \models \psi.$$

Nebel a démontré dans [77] que le problème de décision correspondant à l'inférence $\vdash_{mc}$, noté dans ce qui suit par MaxCons, est un problème Π_2^P-complet.

Soient Pos-S, Pos-W, Demo, CmpIncl, PLex et CmpLex les problèmes de décision respectivement associés aux inférences $\vdash_{spos}$, $\vdash_{wpos}$, $\vdash_{demo}$, $\vdash_{cincl}$, $\vdash_{plex}$ et $\vdash_{clex}$.

10.2.1 Complexité de l'inférence lexicographique partielle

La complexité du problème de décision associé à l'inférence lexicographique partielle est donnée par la proposition suivante :

Proposition 20 PLex *est Π_2^p-complet.*

Preuve.

La preuve de cette proposition consiste en deux étapes : preuve d'appartenance à la classe Π_2^p et preuve de difficulté par rapport à cette même classe d'où la complétude.

1. **Appartenance à Π_2^p**

   Afin de montrer que le problème PLex est dans Π_2^p, nous montrons que le problème complémentaire co-PLex ($(\Sigma, \preceq) \nvdash_{plex} \psi$) appartient à Σ_2^p. Cette dernière appartenance découle de l'algorithme 10.1.

Algorithme 10.1: co-PLex$((\Sigma, \preceq), \psi)$
début
1. deviner une sous-base A de Σ
2. vérifier que A est cohérente
3. vérifier que $A \in Min(Cons(\Sigma, \preceq), \prec_{plex})$
4. vérifier que $A \not\models \psi$

fin

Dans cet algorithme, les points 2 et 4 peuvent être résolus en utilisant un oracle NP. Quant au point 3, il peut être résolu en vérifiant s'il n'existe pas une sous-base cohérente B telle que $B \prec_{plex} A$. Or, le problème qui consiste à vérifier si une telle base existe, noté par NotLexPref, peut être résolu via l'algorithme 10.2.

Algorithme 10.2: NotLexPref$((\Sigma, \preceq), A)$
begin
  deviner une interprétation ω
  $B \leftarrow \emptyset$
  for *chaque* $\phi \in \Sigma$ **do**
    if ω *satisfait* ϕ **then**
      $B \leftarrow B \cup \{\phi\}$
  vérifier que $B \prec_{lex} A$
end

L'algorithme 10.2 est non déterministe polynomial. Donc, NotLexPref $\in$ NP. De plus, on peut réduire le problème GSat (le problème de satisfiabilité d'une formule propositionnelle) qui est NP-complet à NotLexPref en utilisant une transformation polynomiale. Par conséquent, NotLexPref est NP-complet.

Ainsi, l'algorithme 10.1 est non déterministe (point 1) polynomial et utilise un oracle NP Donc, co-PLex $\in NP^{NP} = \Sigma_2^p$ si bien que PLex $\in$ co-$\Sigma_2^p = \Pi_2^p$.

2. **Complétude pour Π_2^p**

   Étant donnée une base de croyances Σ où on ignore le préordre entre les formules, considérons une nouvelle base de croyances partiellement préordonnées $(\Sigma, \preceq)$ telle que : $\forall \alpha, \beta \in \Sigma : \alpha \sim \beta$.

   Maintenant, montrons que $MaxCons(\Sigma) = Min(Cons(\Sigma, \preceq), \prec_{plex})$.

   – Nous avons montré dans le chapitre 9 que chaque sous-base cohérente p-lexicographiquement préférée est maximale cohérente, c'est-à-dire que $Min(Cons(\Sigma, \preceq), \prec_{plex}) \subseteq MaxCons(\Sigma)$.
   – Pour montrer l'autre inclusion, il suffit de prouver que :

$$\forall A, B \in MaxCons(\Sigma) \ : A \ \sim_{lex}^{p} B.$$

   La base partiellement préordonnée $(\Sigma, \preceq)$ peut être partionnée de cette manière $(\Sigma, \preceq) = \bigcup_{i=1}^{m=|\Sigma|} E_i$ tel que
   – $\forall i, 1 \leq i \leq m : E_i$ contient une seule formule $\phi \in \Sigma$.
   – $\forall i, 1 \leq i \leq m, \forall j, 1 \leq j \leq m$ tel que $i \neq j$, on a $E_i \sim_s E_j$.

   Soient $A, B \in MaxCons(\Sigma)$. $A \not\subset B$ et $B \not\subset A$ car A et B sont maximales cohérentes. Donc, $\exists \alpha \in A \setminus B$ et $\exists \beta \in B \setminus A$. Soient $E_a = \{\alpha\}$ et $E_b = \{\beta\}$. Alors, $\exists b, 1 \leq b \leq m$ avec $|E_b \cap B| = 1 > |E_b \cap A| = 0$ tel que $\nexists j, 1 \leq j \leq m$ avec $E_j \prec_s E_b$ et $|E_j \cap A| > |E_j \cap B|$. En effet, $\forall j, 1 \leq j \leq m$ et $j \neq b$, $E_j \sim_s E_b$.

   Ceci signifie que $A \not\preceq_{plex} B$.

   D'une manière similaire, nous montrons que $B \not\preceq_{plex} A$. Donc, $\forall A \in MaxCons(\Sigma)$, $A \in Min(Cons(\Sigma, \preceq), \prec_{plex})$.

Par conséquent, $MaxCons(\Sigma) = Min(Cons(\Sigma, \preceq), \prec_{plex})$ c'est-à-dire $\Sigma \vdash_{mc} \psi$ ssi $(\Sigma, \preceq) \vdash_{plex} \psi$. Donc, MaxCons est réductible en PLex : MaxCons $\propto$ PLex. De plus, puisque MaxCons est Π_2^p-complet et PLex $\in \Pi_2^p$, on déduit que PLex est Π_2^p-complet.

■

10.2.2 Complexité de l'inférence lexicographique basée-compatibles

En ce qui concerne la complexité du problème de décision inhérent à l'inférence lexicographique basée-compatibles, elle est fournie par la proposition suivante :

Proposition 21 CmpLex *est* Π_2^p*-complet.*

Preuve.

1. **L'appartenance à Π_2^p**
   En effet, nous montrons que le problème complémentaire co-CmpLex $\in \Sigma_2^p$ via l'algorithme 10.3.

Algorithme 10.3: co-CmpLex$((\Sigma, \preceq), \psi)$
begin
1. deviner une base de croyances totalement préordonnées $(\Sigma, \leq)$
2. vérifier que $(\Sigma, \leq)$ est compatible avec $(\Sigma, \preceq)$
3. vérifier que $(\Sigma, \leq) \nvdash_{lex} \psi$

end

Cet algorithme est non déterministe étant donné le point 1. Le point 2 peut être exécuté en temps polynomial. Quant au point 3, il peut être exécuté en utilisant un nombre polynomial d'appels à un oracle NP puisque Lex est Δ_2^p-complet. Donc, co-CmpLex $\in NP^{NP} = \Sigma_2^p$ ce qui signifie que CmpLex appartient à la classe Π_2^p.

2. **Complétude pour Π_2^p**
   En utilisant la même transformation polynomiale que nous avons proposée pour PLex, nous démontrons que MaxCons se réduit en CmpLex : MaxCons $\propto$ CmpLex. En effet, étant donnée une base de croyances Σ, considérons une nouvelle base de croyances partiellement préordonnées $(\Sigma, \preceq)$ telle que : $\forall \alpha, \beta \in \Sigma : \alpha \sim \beta$.
   Maintenant, montrons que $MaxCons(\Sigma) = CmpLex(\Sigma, \preceq)$.
   – On sait que $\forall A \in Min(Cons(\Sigma, \preceq), <_{lex})$ où $(\Sigma, \leq) \in Comp(\Sigma, \preceq)$, on a $A \in MaxCons(\Sigma)$.
   Donc, $CmpLex(\Sigma, \preceq) = \bigcup_{(\Sigma, \leq) \in Comp(\Sigma, \preceq)} Min(Cons(\Sigma, \preceq), <_{lex}) \subseteq MaxCons(\Sigma)$.

– Soit $A \in MaxCons(\Sigma)$.
Considérons une base de croyances totalement préordonnées $(\Sigma, \leq) \in Comp(\Sigma, \preceq)$ telle que :
(a) $\forall \alpha \in A, \forall \beta \in A : \alpha \approx \beta$ et
(b) $\forall \alpha \in A, \forall \beta \notin A : \alpha \prec \beta$.
Notons qu'une telle compatible existe puisqu'on considère tous les préordres totaux qui étendent un préordre partiel sans aucune contrainte.
Donc, $(\Sigma, \leq) = (S_1, \ldots, S_k)$ $(k \geq 2)$ et $S_1 = A$.[1]
Clairement, on a $Min(Cons(\Sigma, \leq), <_{lex}) = A$ puisque $A \in MaxCons(\Sigma)$.
Donc, $MaxCons(\Sigma) \subseteq \bigcup_{(\Sigma,\leq) \in Comp(\Sigma,\preceq)} Min(Cons(\Sigma, \preceq), <_{lex}) = CmpLex(\Sigma, \preceq)$.

On déduit alors que $MaxCons(\Sigma) = CmpLex(\Sigma, \preceq)$.
Par conséquent, $\Sigma \vdash_{mc} \psi$ ssi $(\Sigma, \preceq) \vdash_{clex} \psi$ ce qui implique que MaxCons $\propto$ CmpLex et donc CmpLex est Π_2^p-complet.

■

10.2.3 Complexité de l'inférence démocratique

La complexité liée à l'inférence démocratique est identique à celle de l'inférence lexicographique partielle et donc à celle de l'inférence lexicographique basée-compatibles comme le montre la proposition suivante :

Proposition 22 Demo *est* Π_2^p*-complet.*

Preuve.

1. **Appartenance à Π_2^p**
Afin de prouver l'appartenance du problème Demo à la classe Π_2^p, il suffit de démontrer par exemple que le problème complémentaire co-Demo appartient à la classe Σ_2^p. Ceci est possible en utilisant un algorithme similaire à celui que nous avons proposé pour prouver l'appartenance de co-PLex à Σ_2^p.
2. **Complétude pour Π_2^p**
Concernant la complétude, étant donné que le problème Demo généralise le problème Incl pour des bases de croyances partiellement préordonnées, on a Incl $\propto$ Demo. Or, le problème Incl est Π_2^p-complet ce qui nous permet de conclure que le problème Demo est Π_2^p-difficile. Comme Demo est en plus dans Π_2^p, on peut déduire que Demo est Π_2^p-complet.

■

[1] Nous supposons que Σ est incohérente donc $k \geq 2$.

10.2.4 Complexité de l'inférence relative à la préférence de l'inclusion basée-compatibles

Contrairement aux problèmes PLex, CmpLex et Demo, afin de prouver l'appartenance du problème CmpIncl à la classe de complexité Π_2^p, nous n'avons pas réussi à trouver un algorithme permettant de le résoudre et qui permet de déduire que le problème correspondant est dans Π_2^p.

En revanche, nous le démontrons en se basant sur le fait que la classe Π_2^p est fermée par réduction polynomiale, et en prouvant que ce problème est réductible au problème CmpLex. Or, ce dernier point s'appuie sur les lemmes 1, 2 et 3 présentés successivement dans ce qui suit.

Lemme 1 *Soit $(\Sigma, \preceq)$ une base de croyances totalement préordonnées et soit $(\Sigma, \prec_I)$ une nouvelle base de croyances partiellement strictement ordonnées obtenue à partir de $(\Sigma, \preceq)$ en remplaçant les égalités dans $(\Sigma, \preceq)$ par des incomparabilités : $\forall \varphi, \psi \in \Sigma$, si $\phi = \psi$ alors $\phi \sim_I \psi$. Alors, on a :*

$$Min(Cons(\Sigma, \preceq), <_{incl}) = CmpIncl(\Sigma, \prec_I).$$

Preuve.

Soit $(\Sigma, \preceq) = (S_1, \ldots, S_m)$ avec $m \geq 1$.

- Montrons que $Min(Cons(\Sigma, \preceq), <_{incl}) \subseteq CmpIncl(\Sigma, \prec_I)$.
  On sait que $CmpIncl(\Sigma, \prec_I) = \bigcup_{(\Sigma, \leq_I) \in Comp(\Sigma, \prec_I)} Min(Cons(\Sigma, \leq_I), <_{incl})$.
  Puisque $(\Sigma, \preceq) \in Comp(\Sigma, \prec_I)$ (car il suffit de remplacer les incomparabilités par des égalités), on déduit que $Min(Cons(\Sigma, \preceq), <_{incl}) \subseteq CmpIncl(\Sigma, \prec_I)$.
- Montrons maintenant que $CmpIncl(\Sigma, \prec_I) \subseteq Min(Cons(\Sigma, \preceq), <_{incl})$.
  Supposons qu'il existe une base totalement préordonnée $(\Sigma, \leq_I) \in Comp(\Sigma, \prec_I)$ pour laquelle il existe une sous-base cohérente $B \in Min(Cons(\Sigma, \leq_I), <_{incl})$ mais $B \notin Min(Cons(\Sigma, \preceq), <_{incl})$.
  Soit $(\Sigma, \leq_I) = (R_1, \ldots, R_l)$ avec $l \geq 1$.
  A chaque strate S_k $(1 \leq k \leq m)$, correspond un ensemble de strates R_p's.
  Posons $f(S_k) = \{R_p, 1 \leq p \leq l$ tel que $R_p \subseteq S_k\}$. Donc, $S_k = \bigcup_{R_p \in f(S_k)} R_p$.
  Maintenant, $B \notin Min(Cons(\Sigma, \preceq), <_{incl})$ signifie qu'il existe une sous-base cohérente A de Σ telle que $A <_{incl} B$ par rapport à la base $(\Sigma, \leq)$. Autrement dit, $\exists i, 1 \leq i \leq m$ tel que $S_i \cap B \subset S_i \cap A$ et $\forall j, 1 \leq j < i$, on a $S_i \cap B = S_i \cap A$.
  Ce qui signifie que $\exists i, 1 \leq i \leq m$ tel que $(\bigcup_{R_p \in f(S_i)} R_p) \cap B \subset (\bigcup_{R_p \in f(S_i)} R_p) \cap A$, et $\forall j, 1 \leq j < i$, on a $(\bigcup_{R_p \in f(S_i)} R_p) \cap B = (\bigcup_{R_p \in f(S_i)} R_p) \cap A$

D'une part, $(\bigcup_{R_p \in f(S_i)} R_p) \cap B \subset (\bigcup_{R_p \in f(S_i)} R_p) \cap A$ implique que $\forall R_p \in f(S_i)$, on a $R_p \cap B \subseteq R_p \cap A$ et il existe $R_{p'} \in f(S_k)$ tel que $R_{p'} \cap B \subset R_{p'} \cap A$.
D'autre part, $(\bigcup_{R_p \in f(S_i)} R_p) \cap B = (\bigcup_{R_p \in f(S_i)} R_p) \cap A$ signifie que $\forall R_p \in f(S_i)$, on a $R_p \cap B = R_p \cap A$.
Remarquons que $\forall k, 1 \leq k \leq m$, $\forall k', 1 \leq k' \leq m$, si $k < k'$ alors $\forall R_p \in f(S_k), \forall R_{p'} \in f(S_{k'})$, on a $p < p'$.
En conséquence, $\exists p', 1 \leq p' \leq l$ tel que $R_{p'} \cap B \subset R_{p'} \cap A$ et $\forall p < p'$, on a $R_p \cap B \subseteq R_p \cap A$. Or, ceci veut dire que $A <_{incl} B$ par rapport à la base $(\Sigma, \leq_I)$ ce qui contredit le fait que $B \in Min(Cons(\Sigma, \leq_I)$ d'où $B \in Min(Cons(\Sigma, \leq), <_{incl})$.
Par conséquent, $\bigcup_{(\Sigma, \leq_I) \in Comp(\Sigma, \prec_I)} Min(Cons(\Sigma, \leq_I), <_{incl}) \subseteq Min(Cons(\Sigma, \preceq), <_{incl}$) c'est-à-dire $CmpIncl(\Sigma, \prec_I) \subseteq Min(Cons(\Sigma, \preceq), <_{incl})$.

■

Le lemme suivant étend le lemme précédent au cas des bases partiellement préordonnées.

Lemme 2 *Soit $(\Sigma, \preceq)$ une base de croyances partiellement préordonnées et soit $(\Sigma, \prec_I)$ une nouvelle base de croyances partiellement strictement ordonnées obtenue à partir de $(\Sigma, \preceq)$ en remplaçant les égalités par des incomparabilités : $\forall \varphi, \psi \in \Sigma$, si $\phi = \psi$ alors $\phi \sim_I \psi$. Alors, on a :*

$$CmpIncl(\Sigma, \preceq) = CmpIncl(\Sigma, \prec_I).$$

Preuve.

D'une part, étant donnée une base de croyances totalement préordonnées $(\Sigma, \leq^*)$, on a $(\Sigma, \leq^*) \in Comp(\Sigma, \prec_I)$ si et seulement si $\forall \varphi, \psi \in \Sigma$, si $\varphi \prec \psi$ alors $\varphi <^* \psi$. En effet, en passant de $(\Sigma, \preceq)$ à $(\Sigma, \prec_I)$, on élimine juste les égalités tout en préservant les préférences strictes. Ensuite, pour obtenir les compatibles de $(\Sigma, \prec_I)$, par définition, on doit préserver les égalités ainsi que les préférences strictes, et comme on n'a pas d'égalités, on doit juste garder les préférences strictes.

D'autre part, considérons l'ensemble $Comp(\Sigma, \preceq)$ des bases totalement préordonnées compatibles avec $(\Sigma, \preceq)$. Chaque base $(\Sigma, \leq) \in Comp(\Sigma, \preceq)$ préserve les égalités ainsi que les préférences strictes. Maintenant, soit $(\Sigma, <_I)$ une base partiellement strictement ordonnée obtenue à partir de $(\Sigma, \leq)$ et ce en remplaçant les égalités par des incomparabilités si bien que $(\Sigma, <_I)$ préserve les préférences strictes. Ensuite, en passant aux compatibles de $(\Sigma, <_I)$, on préserve uniquement les préférences strictes car il n'y plus d'égalités. De ce fait,

une base de croyances totalement préordonnées $(\Sigma, \leq^*) \in \bigcup_{(\Sigma,\leq)\in Comp(\Sigma,\preceq)} Comp(\Sigma, <_I)$ si et seulement si $\forall \varphi, \psi \in \Sigma$, si $\varphi \prec \psi$ alors $\varphi <^* \psi$.

En conclusion, $\bigcup_{(\Sigma,\leq)\in Comp(\Sigma,\preceq)} Comp(\Sigma, <_I) = Comp(\Sigma, \prec_I)$.

Ceci signifie que $\bigcup_{(\Sigma,\leq)\in\bigcup_{(\Sigma,\leq)\in Comp(\Sigma,\preceq)} Comp(\Sigma,<_I)} Min(Cons(\Sigma, \leq), <_{incl}) =$

$\bigcup_{(\Sigma,\leq)\in Comp(\Sigma,\prec_I)} Min(Cons(\Sigma, \leq), <_{incl})$. ... (1)

Par ailleurs, d'après le lemme 1, pour chaque base $(\Sigma, \leq) \in Comp(\Sigma, \preceq)$, on a

$\bigcup_{(\Sigma,\leq^i_I)\in Comp(\Sigma,<_I)} Min(Cons(\Sigma, \leq^i_I), <_{incl}) = Min(Cons(\Sigma, \leq), <_{incl})$.

Ce qui implique que $\bigcup_{(\Sigma,\leq)\in Comp(\Sigma,\preceq)} \bigcup_{(\Sigma,\leq^i_I)\in Comp(\Sigma,<_I)} Min(Cons(\Sigma, \leq^i_I), <_{incl}) =$

$\bigcup_{(\Sigma,\leq)\in Comp(\Sigma,\preceq)} Min(Cons(\Sigma, \leq), <_{incl})$.

Mais, $\bigcup_{(\Sigma,\leq)\in Comp(\Sigma,\preceq)} \bigcup_{(\Sigma,\leq^i_I)\in Comp(\Sigma,<_I)} Min(Cons(\Sigma, \leq^i_I), <_{incl}) =$

$\bigcup_{(\Sigma,\leq^i_I)\in\bigcup_{(\Sigma,\leq)\in Comp(\Sigma,\preceq)} Comp(\Sigma,<_I)} Min(Cons(\Sigma, \leq^i_I), <_{incl})$.

A partir de (1), on déduit alors que :

$\bigcup_{(\Sigma,\leq)\in Comp(\Sigma,\preceq)} Min(Cons(\Sigma, \leq), <_{incl}) = \bigcup_{(\Sigma,\leq)\in Comp(\Sigma,\prec_I)} Min(Cons(\Sigma, \leq), <_{incl})$.

Autrement dit, on a $CmpIncl(\Sigma, \preceq) = CmpIncl(\Sigma, \prec_I)$.

■

Par ailleurs, nous montrons le lemme suivant qui stipule que pour une base de croyances partiellement strictement ordonnées, les conséquences par rapport aux relation d'inférence $\vdash_{clex}$ et $\vdash_{cincl}$ coïncident.

Lemme 3 *Soit $(\Sigma, \prec)$ une base de croyances partiellement strictement ordonnées. Alors, on a :*

$$CmpLex(\Sigma, \prec) = CmpIncl(\Sigma, \prec).$$

Preuve.

Par définition, $CmpLex(\Sigma, \prec) = \bigcup_{(\Sigma,\leq)\in Comp(\Sigma,\prec)} Min(Cons(\Sigma, \leq), <_{lex})$ et $CmpIncl(\Sigma, \prec) = \bigcup_{(\Sigma,\leq)\in Comp(\Sigma,\prec)} Min(Cons(\Sigma, \leq), <_{incl})$.

Comme $Min(Cons(\Sigma, \leq), <_{lex}) \subseteq Min(Cons(\Sigma, \leq), <_{incl})$ pour toute base de croyances totalement préordonnées selon [9], on déduit alors que $CmpLex(\Sigma, \prec) \subseteq CmpIncl(\Sigma, \prec)$.

Étant donnée une base de croyances partiellement strictement ordonnées $(\Sigma, \prec)$, partitionnons l'ensemble des compatibles de $(\Sigma, \prec)$, $Comp(\Sigma, \prec)$ en deux sous ensembles $Comp(\Sigma, \prec) = Comp^i(\Sigma, \prec) \cup Comp^e(\Sigma, \prec)$ tels que $Comp^i(\Sigma, \prec)$ contient les compatibles ayant une formule par strate alors que $Comp^e(\Sigma, \prec)$ comprend les compatibles admettant au moins une strate ayant plus d'une formule.

Montrons tout d'abord que :

$$\bigcup_{(\Sigma,\leq)\in Comp(\Sigma,\prec)} Min(Cons(\Sigma, \leq), <_{incl}) = \bigcup_{(\Sigma,\leq)\in Comp^i(\Sigma,\prec)} Min(Cons(\Sigma, \leq), <_{incl}).$$

Pour cela, il suffit de montrer que pour toute compatible $(\Sigma, \leq) \in Comp^e(\Sigma, \prec)$, on a $Min(Cons(\Sigma, \leq), <_{incl}) \subseteq \bigcup_{(\Sigma,\leq)\in Comp^i(\Sigma,\prec)} Min(Cons(\Sigma, \leq), <_{incl})$.

Etant donnée une compatible $(\Sigma, \leq) = (S_1, \ldots, S_m) \in Comp^e(\Sigma, \prec)$ telle que $B \in Min(Cons(\Sigma, \leq), <_{incl})$.

Considérons maintenant une nouvelle compatible $(\Sigma, <^*)$, et ce en éclatant chaque strate S_i $(1 \leq i \leq m)$ de $(\Sigma, \leq)$ en plusieurs nouvelles strates R_p's contenant chacune une seule formule tout en mettant les strates R_p's contenant des éléments de B plus prioritaires que celles qui ne contiennent pas des éléments de B. Il est à noter qu'une telle compatible avec $(\Sigma, \prec)$ existe toujours étant donné que l'égalité entre les éléments de chaque strate S_i $(1 \leq i \leq m)$ de $(\Sigma, \leq)$ n'est rien d'autre qu'une incomparabilité dans la base initiale $(\Sigma, \prec)$. En outre, $f(S_i)$ dénote l'ensemble des strates R_p's issues de la strate S_i. Autrement dit, $S_i = \cup_{R_p\in f(S_i)} R_p$.

Ainsi, $(\Sigma, \leq^*) = (R_1, \ldots, R_l)$ avec $l \geq 1$ admet une formule par strate c'est-à-dire $(\Sigma, \leq^*) \in Comp^i(\Sigma, \prec)$.

Le but ensuite est de montrer que $B \in Min(Cons(\Sigma, <^*), <_{incl})$.

Raisonnons par l'absurde et supposons que $B \notin Min(Cons(\Sigma, <^*), <_{incl})$ ce qui veut dire que $\exists i, 1 \leq i \leq l$ tel que $R_i \cap B \subset R_i \cap A$ et $\forall j, 1 \leq j < i$, $R_j \cap A = R_j \cap B$.

Soit, S_k $(1 \leq k \leq m)$ une strate telle que $R_i \in f(S_k)$. Donc, $\forall R_{p'} \in f(S_k)$ tel que $p' < p$, on a $R_{p'} \cap A = R_{p'} \cap B$, et $\forall R_{p''} \in f(S_k)$ tel que $p'' > p$, on a $R_{p''} \cap B = \emptyset \subseteq R_{p''} \cap B$.

Par conséquent, $S_k \cap B \subset S_k \cap A$. De plus, $\forall l, 1 \leq l < k$, on $S_l \cap B = S_l \cap A$. A partir de ça, on déduit que $A <_{incl} B$ par rapport à $(\Sigma, \leq)$ ce qui est impossible car $B \in Min(Cons(\Sigma, \leq$

$), <_{incl})$. Donc, $B \in Min(Cons(\Sigma, <^*), <_{incl})$ ce qui implique que pour toute compatible $(\Sigma, \leq) \in Comp^e(\Sigma, \prec)$, on a $Min(Cons(\Sigma, \leq), <_{incl}) \subseteq \bigcup_{(\Sigma,\leq)\in Comp^i(\Sigma,\prec)} Min(Cons(\Sigma, \leq), <_{incl})$ si bien que

$$\bigcup_{(\Sigma,\leq)\in Comp(\Sigma,\prec)} Min(Cons(\Sigma, \leq), <_{incl}) = \bigcup_{(\Sigma,\leq)\in Comp^i(\Sigma,\prec)} Min(Cons(\Sigma, \leq), <_{incl}).$$

Par ailleurs, on a

$CmpLex(\Sigma, \prec) = \bigcup_{(\Sigma,\leq)\in Comp(\Sigma,\prec)} Min(Cons(\Sigma, \leq), <_{lex})$

$= \bigcup_{(\Sigma,\leq)\in Comp^i(\Sigma,\prec)} Min(Cons(\Sigma, \leq), <_{lex}) \cup \bigcup_{(\Sigma,\leq)\in Comp^e(\Sigma,\prec)} Min(Cons(\Sigma, \leq), <_{lex})$

$= \bigcup_{(\Sigma,\leq)\in Comp^i(\Sigma,\prec)} Min(Cons(\Sigma, \leq), <_{incl}) \cup \bigcup_{(\Sigma,\leq)\in Comp^e(\Sigma,\prec)} Min(Cons(\Sigma, \leq), <_{lex})$ (voir [28].)

$= \bigcup_{(\Sigma,\leq)\in Comp(\Sigma,\prec)} Min(Cons(\Sigma, \leq), <_{incl}) \cup \bigcup_{(\Sigma,\leq)\in Comp^e(\Sigma,\prec)} Min(Cons(\Sigma, \leq), <_{lex})$ (selon la première partie de cette preuve).

$= CmpInc(\Sigma,_p rec) \cup \bigcup_{(\Sigma,\leq)\in Comp^e(\Sigma,\prec)} Min(Cons(\Sigma, \leq), <_{lex}).$

Mais comme $CmpLex(\Sigma, \prec) \subseteq CmpIncl(\Sigma, \prec)$, on déduit que $CmpLex(\Sigma, \prec) = CmpIncl(\Sigma, \prec)$. ■

Ainsi, en se basant essentiellement sur les lemmes précédents, nous montrons la proposition suivante :

Proposition 23 CmpIncl *est* Π_2^p*-complet.*

Preuve.

- **Appartenance à Π_2^p :**
  Étant donnée une base de croyances partiellement préordonnées $(\Sigma, \preceq)$, d'après le lemme 2, nous avons $CmpIncl(\Sigma, \preceq) = CmpIncl(\Sigma, \prec_I)$ où $(\Sigma, \prec_I)$ est une nouvelle base de croyances partiellement strictement ordonnées obtenue à partir de $(\Sigma, \preceq)$ en remplaçant toutes les égalités par des incomparabilités.
  De plus, nous savons selon le lemme 3 que $CmpIncl(\Sigma, \prec_I) = CmpLex(\Sigma, \prec_I)$ ce qui implique que $CmpIncl(\Sigma, \preceq) = CmpLex(\Sigma, \prec_I)$.
  Ceci signifie que le problème CmpIncl est réductible au problème CmpLex, et puisque ce dernier est Π_2^p-complet, on déduit que le problème CmpIncl appartient à Π_2^p car la classe Π_2^p est fermée par réduction polynomiale.
- **Complétude pour Π_2^p :**

D'abord, le problème CmpIncl généralise le problème Incl c'est-à-dire Incl se réduit à CmplIncl. En outre, le problème Incl est Π_2^p-complet. De ce fait, le problème CmpIncl est Π_2^p-difficile, et puisqu'il appartient à la classe Π_2^p, on conclut qu'il est Π_2^p-complet.

■

10.2.5 Complexité des inférences possibilistes forte et possibiliste faible

En ce qui concerne les inférences possibilistes forte et faible, nous avons prouvé l'appartenance des problèmes de décision associés à la classe Π_2^p. Néanmoins, nous n'avons pas réussi à démontrer leur complétude ou non relativement à cette classe. En revanche, nous avons affiné le résultat en fournissant une borne inférieure telle que décrite par la proposition suivante :

Proposition 24 *La complexité des inférences possibilistes forte est faible est comme suit :*

1. Pos-S $\in \Pi_2^p$ *et est* $\Delta_2^p[O(log\ n)]$*-difficile.*
2. Pos-W $\in \Pi_2^p$ *et est* $\Delta_2^p[O(log\ n)]$*-difficile.*

Preuve.

L'appartenance du problème Pos-S à Π_2^p peut être dérivée via le même principe que nous avons utilisé dans le cas de PLex. Quand à la difficulté relativement à la classe $\Delta_2^p[O(log\ n)]$, elle découle du fait que Pos qui est connu pour être $\Delta_2^p[O(log\ n)]$-complet est un cas particulier de Pos-S lorsque l'on a affaire à des bases de croyances stratifiées. Le même principe de raisonnement est applicable sur le problème Pos-W pour prouver son appartenance à Π_2^p ainsi que sa difficulté par rapport à $\Delta_2^p[O(log\ n)]$.

■

10.2.6 Synthèse des résultats de complexité

Les résultats de complexité pour le raisonnement à partir de bases de croyances partiellement préordonnées sont résumés dans le tableau 10.1

Remarquons que la plupart de ces problèmes de décision sont au plus au second niveau de la hiéarchie polynomiale PH.

10.3 Analyse de prudence

Le but de cette section est de comparer les relations d'inférence à partir de bases de croyances partiellement préordonnées considérées dans ce chapitre en termes de prudence.

Problem d'inférence	Complexité
PLex	Π_2^p-complet
CmpLex	Π_2^p-complet
Demo	Π_2^p-complete
CmpIncl	Π_2^p-complete
Pos-S	dans Π_2^p, $\Delta_2^p[O(logn)]$-difficile
Pos-W	dans Π_2^p, $\Delta_2^p[O(logn)]$-difficile

Tab. 10.1 – Complexité des relations d'inférence à partir de bases de croyances partiellement préordonnées

D'abord, l'inférence démocratique est plus prudente que l'inférence lexicographique partielle (p-lexicographique).

Proposition 25 *Soit $(\Sigma, \preceq)$ une base de croyances partiellement préordonnées et soit ψ une formule propositionnelle. Alors, on a*

$$(\Sigma, \preceq) \vdash_{demo} \psi \Rightarrow (\Sigma, \preceq) \vdash_{plex} \psi$$

et l'inverse n'est pas vérifié.

Preuve.

Pour prouver cette proposition, il suffit de démontrer que $Min(Cons(\Sigma, \preceq), \prec_{plex}) \subset Min(Cons(\Sigma, \preceq), \prec_{demo})$. Étant données deux sous-bases cohérentes A et B de $(\Sigma, \preceq)$, prouvons que $A \prec_{demo} B$ implique $A \preceq_{plex}$, et que l'inverse n'est pas vérifié.

D'une manière équivalente, montrons que $A \not\preceq_{plex} B \Rightarrow A \not\prec_{demo} B$ et que l'inverse n'est pas vérifié.

En effet, par définition, $A \not\preceq_{plex} B$ ssi $\exists i, 1 \leq i \leq m$ tel que $|E_i \cap B| > |E_i \cap A|$ et $\forall j, 1 \leq j \leq m$, on a $E_j \prec_s E_i \Rightarrow |E_j \cap A| \leq |E_j \cap B|$.

On peut alors distinguer deux cas :

– **Cas 1 :** $\forall j, E_j \prec_s E_i \Rightarrow E_j \cap A \subseteq E_j \cap B$.
Ceci signifie que $\forall a \in \bigcup_{E_j \prec_s E_i} E_j$, on a $a \in A \Rightarrow a \in B$.
Donc, $\nexists a \in \Sigma$ tel que $a \in A \setminus B$ et $a \in \bigcup_{E_j \prec_s E_i} E_j$.
Soit $b \in E_i$. Clairement, $\forall a \in \Sigma$, $a \prec b$ ssi $a \in \bigcup_{E_j \prec_s E_i} E_j$.
Ceci signifie que $\forall b \in E_i, \nexists a \in A \setminus B$ tel que $a \prec b$.
D'autre part, $E_i \cap (B \setminus A) \subseteq E_i$ ce qui signifie que $\forall b \in E_i \cap (B \setminus A), \nexists a \in A \setminus B$ tel que $a \prec b$.
Maintenant, $|E_i \cap B| > |E_i \cap A|$ implique que $E_i \cap (B \setminus A) \neq \emptyset$.

Donc, $\exists b \in E_i \cap (B \setminus A), \nexists a \in A \setminus B$ tel que $a \prec b$.

Par conséquent, $\exists b \in B \setminus A, \nexists a \in A \setminus B$ tel que $a \prec b$ c'est-à-dire $A \not\prec_{demo} B$.

– **Cas 2 :** $\exists j, E_j \prec_s E_i$ et $E_j \cap A \not\subseteq E_j \cap B$.

Soit $E_s \in Min(\{E_j \text{ tel que } E_j \prec_s E_i \text{ et } E_j \cap A \not\subseteq E_j \cap B\}, \prec_s)$.

Ceci signifie que $\forall k, E_k \prec_s E_s \Rightarrow (E_k \cap A) \subseteq (E_k \cap B)$.

Donc, $\forall b \in E_s, \nexists a \in A \setminus B$ tel que $a \prec b$.

Alors, $\forall b \in E_s \cap B \setminus A, \nexists a \in A \setminus B$ tel que $a \prec b$.

Par ailleurs, nous avons posé E_s tel que $E_s \cap A \not\subseteq E_s \cap B$. De plus, $|E_s \cap B| \geq |E_s \cap A|$ puisque $E_s \prec_s E_i$ si bien que $E_s \cap B \setminus A \neq \emptyset$.

Donc, $\exists b \in E_s \cap B \setminus A, \nexists a \in A \setminus B$ tel que $a \prec b$.

En conséquence, $\exists b \in B \setminus A, \nexists a \in A \setminus B$ tel que $a \prec b$ ce qui signifie que $A \not\prec_{demo} B$.

Donc, $A \prec_{demo} B \Rightarrow A \preceq_{plex} B$.

Maintenant, montrons que $Min((\Sigma, \preceq), \prec_{plex}) \subseteq Min((\Sigma, \preceq), \prec_{demo})$.

Soit $B \in Min(Cons(\Sigma, \preceq), \prec_{plex})$. Supposons que $B \notin Min(Cons(\Sigma, \preceq), \prec_{demo})$ ce qui signifie que $\exists A \in Cons(\Sigma, \preceq)$ telle que $A \neq B$ et $A \prec_{demo} B$.

Ceci implique que $A \preceq_{plex} B$.

Cependant, $B \in Min(Cons(\Sigma, \preceq), \prec^p_{lex})$ si bien que $A \approx^p_{lex} B$.

Donc, $\forall i, 1 \leq i \leq n$, on a $|E_i \cap A| = |E_i \cap B|$.

Puisque $A \neq B$, $\exists E_j$ tel que $E_j \cap A \neq E_j \cap B$.

Soit $E_s \in Min(\{E_j \text{ tel que } E_j \cap A \neq E_j \cap B\}, \prec_s)$.

Donc, $\forall j$, $E_j \prec_s E_s \Rightarrow E_j \cap A = E_j \cap B$.

Ceci revient à dire que $\forall j$, $E_j \prec_s E_s \Rightarrow E_j \cap (A \setminus B) = \emptyset$.

Dans ce cas, $\forall b \in E_s, \nexists a \in A \setminus B$ tel que $a \prec b$ ce qui implique que

$\forall b \in E_s \cap (B \setminus A), \nexists a \in A \setminus B$ tel que $a \prec b$.

Comme $E_s \cap A \neq E_s \cap B$ et $|E_s \cap A| = |E_s \cap B|$, on déduit que $E_s \cap B \setminus A \neq \emptyset$.

En conséquence, $\exists b \in E_s \cap (B \setminus A), \nexists a \in A \setminus B$ tel que $a \prec b$.

Donc on dérive que $\exists b \in B \setminus A, \nexists a \in A \setminus B$ tel que $a \prec b$.

Autrement dit $A \not\prec_{demo} B$ ce qui contredit l'hypothèse de départ disant que $A \prec_{demo} B$.

Enfin, $B \in Min(Cons(\Sigma, \preceq), \prec_{demo})$.

Par conséquent, $Min(Cons(\Sigma, \preceq), \prec_{plex}) \subseteq Min(Cons(\Sigma, \preceq), \prec_{demo})$.

Par ailleurs, en particulier lorsque $(\Sigma, \preceq)$ est totalement préordonnée, on $Min(Cons(\Sigma, \preceq), \prec_{demo}) = Min(Cons(\Sigma, \preceq), <_{incl})$ et $Min(Cons(\Sigma, \preceq), \prec_{plex}) = Min(Cons(\Sigma, \preceq), <_{lex})$. Etant donné que $Min(Cons(\Sigma, \preceq), <_{lex}) \subset Min(Cons(\Sigma, \preceq), <_{incl})$, on déduit que dans le cas général $Min(Cons(\Sigma, \preceq), \prec_{plex}) \subset Min(Cons(\Sigma, \preceq), \prec_{demo})$.

■

En ce qui concerne l'inférence lexicographique basée-compatibles et l'inférence lexicographique partielle, elles sont incomparables en termes de prudence.

Proposition 26 *Soit* $(\Sigma, \preceq)$ *une base de croyances partiellement préordonnées et soit* ψ *une formule propositionnelle. Alors,*

1. $(\Sigma, \preceq) \vdash_{cincl} \psi$ *n'implique pas* $(\Sigma, \preceq) \vdash_{plex} \psi$,
2. $(\Sigma, \preceq) \vdash_{plex} \psi$ *n'implique pas* $(\Sigma, \preceq) \vdash_{cincl} \psi$.

Preuve.

1. En effet, dans le cas particulier où $(\Sigma, \preceq)$ est totalement préordonnée, on a $CmpIncl(\Sigma, \preceq) = Min(Cons(\Sigma, \preceq), <_{incl})$ et $Min(Cons(\Sigma, \preceq), \prec_{plex}) = Min(Cons(\Sigma, \preceq), <_{lex})$. Comme on sait que $Min(Cons(\Sigma, \preceq), <_{incl}) \not\subset Min(Cons(\Sigma, \preceq), <_{lex})$ (car $Min(Cons(\Sigma, \preceq), <_{lex}) \subset Min(Cons(\Sigma, \preceq), <_{incl})$, on déduit que $CmpIncl(\Sigma, \preceq) \not\subset Min(Cons(\Sigma, \preceq), \prec_{plex})$.
2. L'exemple suivant montre que $Min(Cons(\Sigma, \preceq), \prec_{plex}) \not\subset CmpIncl(\Sigma, \preceq)$.

   Exemple 49 *Soit* $(\Sigma, \preceq)$ *une base de croyance partiellement préordonnées telle que* $\Sigma = \{a \wedge \neg d, \neg a, a \wedge f, d\}$ *avec* $a \wedge \neg d \prec a \wedge f$ *et* $\neg a \prec d$.
   D'abord, $MaxCons = \{A, B, C\}$ *tel que*
   (a) $A = \{a \wedge \neg d, a \wedge f\}$,
   (b) $B = \{\neg a, d\}$,
   (c) $C = \{a \wedge f, d\}$.
   Ensuite, d'une part, on a $CmpIncl(\Sigma, \preceq) = \{A, B\}$. *En effet, il n'existe aucune base totalement préordonnée* $(\Sigma, \leq)$ *compatible avec* $(\Sigma, \preceq)$ *telle que* $C \in Min(Cons(\Sigma, \leq), <_{lex})$.
   D'autre part, $A \sim^p_{lex} B$, $A \sim^p_{lex} C$ *et* $B \sim^p_{lex} C$ *ce qui signifie que* $Min(Cons(\Sigma, \preceq), \prec_{plex}) = \{A, B, C\}$.
   Par conséquent , $Min(Cons(\Sigma, \preceq), \prec_{plex}) \not\subset CmpIncl(\Sigma, \preceq)$.

   ■

Par ailleurs, l'inférence possibiliste faible est plus prudente que l'inférence démocratique.

Proposition 27 *Soit* $(\Sigma, \preceq)$ *une base de croyances partiellement préordonnées et soit* ψ *une formule propositionnelle. Alors,*

$$(\Sigma, \preceq) \vdash_{wpos} \psi \text{ implique que } (\Sigma, \preceq) \vdash_{demo} \psi$$

et l'inverse n'est pas vérifié.

Preuve.

Étant données $A, B \in Cons(\Sigma)$, montrons que $A \prec_{wbo} B \Rightarrow A \prec_{demo} B$.

Supposons que $A \prec_{wbo} B$ et $A \not\prec_{demo} B$. Donc,

– $\forall \delta \notin A, \exists \lambda \notin B$ tel que $\lambda \prec \delta \ldots$ **(h1)**

– $\exists \beta \in B \setminus A, \forall \alpha \in A \setminus B$, on a $\alpha \not\prec \beta \ldots$ **(h2)**

Soit $\beta \in B \setminus A$. Alors, $\beta \notin A$, et selon (h1), il doit exister $\lambda \notin B$ tel que $\lambda \prec \beta$. En outre, $\lambda \notin A \setminus B$ car $\forall \alpha \in A \setminus B$, on a $\alpha \not\prec \beta$ selon (h2). $\lambda \notin B$ et $\lambda \notin A \setminus B$ implique que $\lambda \notin A \cup B$ c'est-à-dire $\lambda \in \Sigma \cap \overline{(A \cup B)}$.

Maintenant, soit $\xi \in \Sigma \cap \overline{(A \cup B)}$ tel que $\xi \preceq \lambda$ et $\forall \theta \in \Sigma \cap \overline{(A \cup B)}$, on a $\theta \not\prec \xi$.

Comme $\xi \notin A$, selon (h1), il doit exister $\chi \notin B$ tel que $\chi \prec \xi$.

En plus, puisque $\chi \prec \xi \preceq \lambda \prec \beta$ c'est-à-dire $\chi \prec \beta$, on déduit à partir de (h1) que $\chi \notin A \setminus B$. Ceci signifie que $\chi \notin A \cup B$ c'est-à-dire $\chi \in \Sigma \cap \overline{(A \cup B)}$ mais $\chi \prec \xi$ ce qui contredit la définition de ξ.

En conclusion, $A \prec_{wbo} B$ implique que $A \prec_{demo} B$.

Quant à la réciproque, on peut démontrer qu'elle n'est pas vérifiée en utilisant la propriété de monotonie. En effet, on sait que d'une part, $A \subset B \Rightarrow A \prec_{demo} B$. D'autre part, il a été démontré dans [5] que $A \subset B$ n'implique pas $A \prec_{wbo} B$. En conséquence, $A \prec_{demo} B$ n'implique pas que $A \prec_{wbo} B$.

■

Enfin, l'inférence lexicographique basée-compatibles est moins prudente que l'inférence relative à la préférence de l'inclusion basée-compatibles.

Proposition 28 *Soit $(\Sigma, \preceq)$ une base de croyance partiellement préordonnée et soit ψ une formule propositionnelle. Alors,*

$$(\Sigma, \preceq) \vdash_{cincl} \psi \text{ implique que } (\Sigma, \preceq) \vdash_{clex} \psi,$$

et l'inverse n'est pas vérifié.

Preuve.

On sait que , $\forall (\Sigma, \leq) \in Comp(\Sigma, \preceq)$, on a $Min(Cons(\Sigma, \leq), <_{lex}) \subset Min(Cons(\Sigma, \leq), <_{incl})$.

Donc,

$$\bigcup_{(\Sigma,\leq) \in Comp(\Sigma,\preceq)} Min(Cons(\Sigma, \leq), <_{lex}) \subset \bigcup_{(\Sigma,\leq) \in Comp(\Sigma,\preceq)} Min(Cons(\Sigma, \leq), <_{incl}).$$

Ceci veut dire que $CmpLex(\Sigma, \preceq) \subset CmpIncl(\Sigma, \preceq)$.

■

La relation entre l'inférence p-lexicographique et l'inférence lexicographique basée-compatibles a été définie dans le chapitre 9. Rappelons que cette realtion stipule que la première est plus prudente que la dernière.

Par ailleurs, il a été démontré dans [28] que l'inférence démocratique est plus prudente que l'inférence relative à la préférence pour l'inclusion basée-compatible. Enfin, dans [5], les auteurs ont prouvé que l'inférence possibiliste forte est plus prudente que l'inférence possibiliste faible.

Ces résultats de prudence sont récapitulés à travers la figure 10.1 où $R_1 \rightarrow R_2$ signifie que R_1 est plus prudente que R_2.

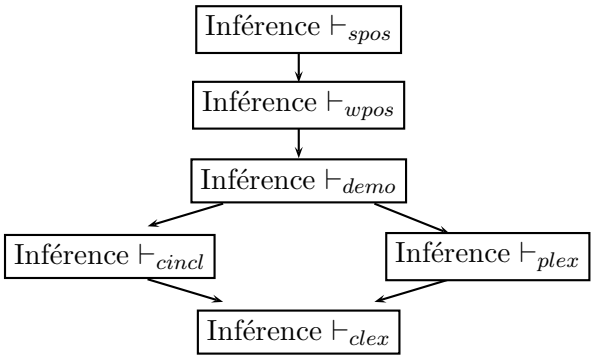

FIG. 10.1 – Prudence des relations d'inférence à partir de bases de croyances partiellement préordonnées.

10.4 Propriétés logiques

Nous étudions dans cette section les propriétés de déduction selon Kraus, Lehmann et Magidor telles que décrites dans [64] des relations d'inférence à partir de bases de croyances partiellement préordonnées considérées dans ce chapitre.

Afin de respecter la terminologie utilisée dans [64], nous commençons par les étendre de telle sorte qu'elles soient définies entre deux formules par rapport à une base de croyances partiellement préordonnées. L'extension que nous avons adoptée est similaire à celle décrite dans le chapitre 4 dans le cadre de bases de croyances totalement préordonnées et qui est bien connue dans la littérature.

En fait, l'idée est de dire qu'une formule α infère une autre formule β par rapport à une base de croyances partiellement préordonnées relativement à une relation d'inférence $\mid\sim_R$

est équivalent à dire que la nouvelle base de croyances partiellement préordonnées issue de la première base à laquelle on rajoute la formule α en lui affectant le plus haut degré de priorité infère la formule β selon la relation d'inférence $\vdash_R$. Plus formellement, cette extension est définie de la manière suivante :

Définition 142 *Soit $(\Sigma, \preceq)$ une base de croyances partiellement préordonnées et soient φ et ψ deux formules propositionnelles. Alors, $\varphi \mid\!\sim_R^{(\Sigma,\preceq)} \psi$ où $R \in \{spos, wpos, demo, cincl, plex, clex\}$ si et seulement si $(\Sigma \cup \{\varphi\}, \preceq^{\varphi}) \vdash_R \psi$ où $\preceq^{\psi}$ est un nouveau préordre qui étend le préordre $\preceq$ de telle sorte que la formule φ soit plus prioritaire que toutes les formules de Σ :*

- $\forall \alpha, \beta \in \Sigma : \alpha \preceq \beta$ *si et seulement si* $\alpha \preceq^{\varphi} \beta$,
- $\forall \alpha \in \Sigma : \varphi \prec^{\varphi} \alpha$.

10.4.1 Propriétés de l'inférence lexicographique partielle

Étant donnée une base de croyances partiellement préordonnées $(\Sigma, \preceq)$ et une formule propositionnelle φ, nous proposons dans un premier temps le lemme suivant qui décrit la relation entre les sous-bases cohérentes p-lexicographiquement préférées de $(\Sigma \cup \{\varphi\}, \preceq^{\varphi})$ et les sous-bases p-lexicographiquement préférées de $(\Sigma, \preceq)$ qui sont cohérentes avec la formule φ. Mais tout d'abord, donnons cette notation qui sera utilisée tout au long de cette section :

Notation 2 *Soit $(\Sigma, \preceq)$ une base de croyances partiellement préordonnées et soit φ une formule. L'ensemble de toutes les sous-bases cohérentes de $(\Sigma, \preceq)$ qui sont cohérentes avec φ est noté par $\varphi - Cons(\Sigma, \preceq)$.*

Lemme 4 *Soit $(\Sigma, \preceq)$ une base de croyances partiellement préordonnées et soit φ une formule cohérente. Alors, tous les éléments de $Min(Cons(\Sigma \cup \{\varphi\}, \preceq^{\varphi}), \prec_{plex})$ sont de la forme $B \cup \{\varphi\}$ où $B \in Min(\varphi - Cons(\Sigma, \preceq), \prec_{plex})$.*

Preuve.

Soit $(\Sigma, \preceq)$ une base de croyances partiellement préordonnées et soient φ et ψ deux formules propositionnelles.

D'abord, n'importe quelle sous-base cohérente de $(\Sigma \cup \varphi, \preceq^{\varphi})$ préférée par rapport à $\prec_{plex}$ contient au moins la formule $\{\varphi\}$. En effet, étant données deux sous-bases cohérentes A et B telles que $A = \{\varphi\}$ et $\varphi \notin B$, il est clair que $A \prec_{plex} B$.

Montrons maintenant que $\{\varphi\} \cup B \in Min(Cons(\Sigma \cup \varphi, \preceq^{\varphi}), \prec_{plex}$ si et seulement si B est préférée par rapport à $\prec_{plex}$ parmi les sous-bases φ-cohérentes de $(\Sigma, \preceq)$.

– Supposons que $\{\varphi\} \cup B \in Min(Cons(\Sigma \cup \varphi, \preceq^{\varphi}), \prec_{plex})$ et que B n'est pas préférée par rapport à $\prec_{plex}$ parmi les sous-bases φ-cohérentes de $(\Sigma, \preceq)$.
Donc, il existe une sous-base φ-cohérente A de $(\Sigma, \preceq)$ telle que $A \prec_{plex} B$ par rapport à $(\Sigma, \preceq)$.
Remarquons que $A \cup \{\varphi\}$ est une sous-base cohérente de $(\Sigma \cup \varphi, \preceq^{\varphi})$.
Soit $(\Sigma, \preceq) = (E_1, \ldots, E_n)$ $(n \geq 1)$ et $(\Sigma \cup \varphi, \preceq^{\varphi}) = (E_0, E_1, \ldots, E_n)$ avec $E_0 = \{\varphi\}$.
Nous montrons alors que :
  1. $A \cup \{\varphi\} \preceq_{plex} B \cup \{\varphi\}$. En effet, on a $\forall i, 0 \leq i \leq n$, si $|E_i \cap B| > |E_i \cap A|$, alors $\exists j, 0 \leq j \leq n$ tel que $E_j \prec_s E_i$ et $|E_i \cap A| > |E_i \cap B|$ puisque $A \prec_{plex} B$ et $|E_0 \cap A| = |E_0 \cap B|$.
  2. $B \cup \{\varphi\} \not\preceq_{plex} A \cup \{\varphi\}$. En effet, puisque $B \not\preceq_{plex} A$, $\exists i, 1 \leq i \leq n$ tel que $|E_i \cap A| > |E_i \cap B|$ et $\forall j, 1 \leq j \leq n$, si $E_j \prec_s E_i$ alors $|E_i \cap A| \leq |E_i \cap B|$. Comme $|E_0 \cap A| = |E_0 \cap B|$ on conclut que $\exists i, 0 \leq i \leq n$ tel que $|E_i \cap A| > |E_i \cap B|$ et $\forall j, 0 \leq j \leq n$, si $E_j \prec_s E_i$ alors $|E_i \cap A| \leq |E_i \cap B|$.

  Par conséquent, $A \cup \{\varphi\} \prec_{plex} B \cup \{\varphi\}$ ce qui contredit le fait que $\{\varphi\} \cup B \in Min(Cons(\Sigma \cup \varphi, \preceq^{\varphi}), \prec_{plex})$.
– Supposons q'il existe une sous-base B telle que $B \in Min(\varphi - Cons(\Sigma, \preceq), \prec_{plex})$ alors que $B \cup \{\varphi\} \notin Min(Cons(\Sigma \cup \varphi, \preceq^{\varphi}), \prec_{plex})$.
Ceci signifie qu'il existe une sous-base $A \cup \{\varphi\}$ de $(\Sigma \cup \varphi, \preceq^{\varphi})$ où A est une sous-base de $(\Sigma, \preceq)$ telle que $A \cup \{\varphi\} \prec_{plex} B \cup \{\varphi\}$.
  1. D'une part, on a $A \cup \{\varphi\} \preceq_{plex} B \cup \{\varphi\}$ si et seulement si $\forall i, 0 \leq i \leq n$, si $|E_i \cap (B \cup \{\varphi\})| > |E_i \cap (A \cup \{\varphi\})|$ alors $\exists j, 0 \leq j \leq n$ tel que $E_j \prec_s E_i$ et $|E_j \cap (B \cup \{\varphi\})| > |E_j \cap (B \cup \{\varphi\})|$.
  Comme $|E_0 \cap (A \cup \{\varphi\})| = |E_0 \cap (B \cup \{\varphi\})|$, on peut s'en passer de la classe E_0 au niveau de la propriété précédente. De plus, il est clair que $|E_i \cap (B \cup \{\varphi\})| > |E_i \cap (A \cup \{\varphi\})|$ si et seulement si $|E_i \cap B| > |E_i \cap A|$ et $|E_i \cap (B \cup \{\varphi\})| \leq |E_i \cap (A \cup \{\varphi\})|$ si et seulement si $|E_i \cap B| > |E_i \cap A|$.
  Ceci implique que $\forall i, 1 \leq i \leq n$, si $|E_i \cap B| > |E_i \cap A|$ alors $\exists j, 1 \leq j \leq n$ tel que $E_j \prec_s E_i$ et $|E_j \cap B| > |E_j \cap B|$ d'où $A \preceq_{plex} B$ par rapport à $(\Sigma, \preceq)$.
  2. D'autre part, $B \cup \{\varphi\} \not\prec_{plex} A \cup \{\varphi\}$ si et seulement si $\exists i, 0 \leq i \leq n$ tel que $|E_i \cap (A \cup \{\varphi\})| > |E_i \cap (B \cup \{\varphi\})|$ et $\forall j, 0 \leq j \leq n$, si $E_j \prec_s E_i$ alors $|E_i \cap (A \cup \{\varphi\})| \leq |E_i \cap (B \cup \{\varphi\})|$.
  Étant donné que $|E_0 \cap (A \cup \{\varphi\})| = |E_0 \cap (B \cup \{\varphi\})|$ et que $|E_i \cap (B \cup \{\varphi\})| > |E_i \cap (A \cup \{\varphi\})|$ si et seulement si $|E_i \cap B| > |E_i \cap A|$ et $|E_i \cap (B \cup \{\varphi\})| \leq |E_i \cap (A \cup \{\varphi\})|$ si et seulement si $|E_i \cap B| > |E_i \cap A|$, on conclut que $\exists i, 1 \leq i \leq n$ tel que $|E_i \cap A| > |E_i \cap B|$ et $\forall j, 1 \leq j \leq n$, si $E_j \prec_s E_i$ alors $|E_i \cap A| \leq |E_i \cap B|$.
  Ceci signifie que $B \not\prec_{plex} A$ par rapport à $(\Sigma, \preceq)$.

  En conclusion, on a $A \prec_{plex} B$ par rapport à $(\Sigma, \preceq)$. En outre A est cohérente avec φ. Néanmoins ceci est impossible car $B \in Min(\varphi - Cons(\Sigma, \preceq), \prec_{plex})$.

■

Nous proposons ensuite le lemme suivant qui définit une caractérisation sémantique de l'inférence $\mid\sim_{plex}$ en termes de modèles préférés.

Lemme 5 *Soit $(\Sigma, \preceq)$ une base de croyances partiellement préordonnées et soient φ et ψ deux formules propositionnelles. Alors,*

$$\varphi \mid\sim_{plex}^{(\Sigma,\preceq)} \psi \text{ ssi } Min(Mod(\varphi), \lhd_{plex}^{(\Sigma,\preceq)}) \subseteq Mod(\psi).$$

Preuve.

Par définition, $\varphi \mid\sim_{plex}^{(\Sigma,\preceq)} \psi$ si et seulement si $(\Sigma \cup \{\varphi\}, \preceq^{\varphi}) \vdash_{plex} \psi$ c'est-à-dire $\forall B' \in Min(Cons(\Sigma \cup \{\varphi\}, \preceq^{\varphi}), <_{plex}), B' \models \psi$.

Ceci équivaut à dire que $\bigcup_{B' \in Min(Cons(\Sigma\cup\{\varphi\},\preceq^{\varphi}),<_{plex})} Mod(B') \subseteq Mod(\psi)$.

Montrons alors que :

$B' \in Min(Cons(\Sigma\cup\{\varphi\}, \preceq^{\varphi}), \prec_{plex})$ ssi $\omega \in Min(Mod(\varphi), \lhd_{plex}^{(\Sigma,\preceq)})$ où $V(\omega, \Sigma\cup\{\varphi\}) = B'$ où $V(\omega, A)$ désigne l'ensemble des formules de A satisfaites par ω.

- Supposons que $B' \in Min(Cons(\Sigma\cup\{\varphi\}, \preceq^{\varphi}), \prec_{plex})$ et que $\omega \notin Min(Mod(\varphi), \lhd_{plex}^{(\Sigma,\preceq)})$ avec $V(\omega, \Sigma \cup \{\varphi\}) = B'$.
  Donc, $\exists \omega' \in Mod(\varphi)$ tel que $\omega' \prec_{plex}^{(\Sigma,\preceq)} \omega$.
  Par définition, ceci veut dire que $V(\omega', \Sigma) \prec_{plex}^{(\Sigma,\preceq)} V(\omega, \Sigma)$.
  Par ailleurs, selon le lemme 4, B' est de la forme $\{\varphi\}\cup B$ où $B \in Min(\varphi - Cons(\Sigma, \preceq), \prec_{plex})$.
  Clairement, B n'est rien d'autre que $V(\omega, \Sigma)$. De plus, $V(\omega', \Sigma)$ est une sous-base de $(\Sigma, \preceq)$ cohérente avec φ (car ω' qui la satisfait est un modèle de φ). Néanmoins, ceci est impossible car ça contredit le fait que $B \in Min(\varphi - Cons(\Sigma, \preceq), \prec_{plex})$.
- Supposons que $\omega \in Min(Mod(\varphi), \lhd_{plex}^{(\Sigma,\preceq)})$ avec $V(\omega, \Sigma \cup \{\varphi\}) = B'$ et que $B' \notin Min(Cons(\Sigma \cup \{\varphi\}, \preceq^{\varphi}), \prec_{plex})$.
  Comme $\omega \in Mod(\varphi)$, on déduit que $V(\omega, \Sigma\cup\{\varphi\}) = \{\varphi\}\cup B$ où B est une sous-base de $(\Sigma, \preceq)$ cohérente avec φ.
  Maintenant, $B' \notin Min(Cons(\Sigma\cup\{\varphi\}, \preceq^{\varphi}), \prec_{plex})$ implique qu'il existe une sous-base A de $(\Sigma, \preceq)$ cohérente avec φ telle que $A \prec_{plex} B$.
  Puisque A est cohérente avec φ, soit ω' tel que ω' satisfait A et satisfait φ ce qui implique que $\omega' \in Mod(\varphi)$.
  De plus, il est clair que $V(\omega, \Sigma) = B$ et $V(\omega', \Sigma) = A$ d'où $V(\omega', \Sigma) \prec_{plex}^{(\Sigma,\preceq)} V(\omega, \Sigma)$ ce qui par définition veut dire que $\omega' \prec plex^{(\Sigma,\preceq)} \omega'$.
  Cependant, ceci est incohérent avec l'hypothèse de départ qui dit que $\omega \in Min(Mod(\varphi), \lhd_{plex}^{(\Sigma,\preceq)})$.

■

Proposition 29 *L'inférence $\mid\sim_{plex}$ vérifie les propriétés du système P.*

Preuve.

Tout d'abord, on peut facilement voir que la relation $\lhd_{plex}^{(\Sigma,\preceq)}$ définit un ordre strict partiel. De plus, on a $\varphi \mid\sim_{plex}^{(\Sigma,\preceq)} \psi$ *ssi* $Min(Mod(\varphi), \lhd_{plex}^{(\Sigma,\preceq)}) \subseteq Mod(\psi)$, pour toute base de croyances partiellement préordonnées $(\Sigma, \preceq)$ et pour toute paire de formule (φ, ψ). De ce fait, d'après le théorème de caractérisation de Kraus, Lehmann et Magidor de l'inférence préférentielle, on conclut que l'inférence $\mid\sim_{plex}$ est bien une inférence préférentielle.

■

En revanche, l'inférence lexicographique partielle ne définit pas une relation d'inférence rationnelle.

Proposition 30 *L'inférence $\mid\sim_{plex}$ ne vérifie pas la monotonie rationnelle.*

Preuve.

Considérons le contre-exemple suivant :

Exemple 50 *Soit une base de croyances partiellement préordonnées $(\Sigma, \preceq)$ telle que $\Sigma = \{p, q, \neg r\}$ et $p \prec \neg r$, $q \prec \neg r$ et $p \sim q$.*

On peut alors la partitionner en 3 classes $E_1 = \{p\}$, $E_2 = \{q\}$ et $E_3 = \{\neg r\}$.

Considérons trois formules φ, ψ et δ telles que :

- $\varphi = (\neg p \vee \neg q) \wedge (\neg p \vee r)$,
- $\psi = p \vee \neg r$,
- $\delta = r$.

La base $\Sigma \cup \{\varphi\}$ admet 3 bases maximales cohérentes :

- $A = \{\varphi, p\}$,
- $B = \{\varphi, q, \neg r\}$,
- $C = \{p, q, \neg r\}$.

Alors que $A \prec_{plex} C$ et $B \prec_{plex} C$, on a $A \sim_{lex}^{p} B$ ce qui implique que $Min(Cons(\Sigma \cup \{\varphi\}, \preceq^{\varphi}), \prec_{plex}) = \{A, B\}$.

Clairement, $(\Sigma \cup \{\varphi\}, \preceq^{\varphi}) \vdash_{plex} \psi = p \vee \neg r$ (car $A \models \psi$ et $B \models \psi$) ce qui signifie que $\varphi \mid\sim_{plex}^{(\Sigma,\preceq)} \psi$.

Par contre, $(\Sigma \cup \{\varphi\}, \preceq^{\varphi}) \nvdash_{plex} \neg\delta = \neg r$ (puisque $B' \not\models \psi$) c'est-à-dire $\varphi \not\mid\sim_{plex}^{(\Sigma,\preceq)} \neg\delta$.

Considérons maintenant la base $\Sigma \cup \{\varphi \wedge \delta\}$. Dans ce cas, on a 3 nouvelles bases maximales cohérentes :

– $A' = \{\varphi \wedge \delta, p\}$,
– $B' = \{\varphi \wedge \delta, q\}$,
– $C' = \{p, q, \neg r\}$.

De plus, $Min(Cons(\Sigma \cup \{\varphi \wedge \delta\}, \preceq^{\varphi\wedge\delta}), \prec_{plex}) = \{A', B'\}$.

Néanmoins, $(\Sigma \cup \{\varphi\}, \preceq^{\varphi\wedge\delta}) \nvdash_{plex} \psi = p \vee \neg r$ *(car* $A \not\models \psi$*) ce qui signifie que* $\varphi \wedge \delta \ | \sim^{(\Sigma,\preceq)}_{plex} \psi$.

En conclusion, on a $\varphi \mid\sim^{(\Sigma,\preceq)}_{plex} \psi$, $\varphi \nmid\sim^{(\Sigma,\preceq)}_{plex} \neg\delta$ *alors que* $\varphi\wedge\delta \nmid\sim^{(\Sigma,\preceq)}_{plex} \psi$. *Par conséquent,* $\mid\sim_{plex}$ *ne vérifie pas la monotonie rationnelle.*

■

10.4.2 Propriétés de l'inférence lexicographique basée-compatibles

En ce qui concerne l'inférence lexicographique basée-compatibles, et contrairement à l'inférence lexicographique partielle, nous n'avons pas trouvé une caractérisation sémantique de forme préférentielle basée sur un ordre partiel strict. En revanche, nous fournissons dans un premier temps une caractérisation de l'inférence $\mid\sim^{(\Sigma,\preceq)}_{clex}$ relativement aux compatibles de $(\Sigma, \preceq)$ tel que décrit à travers le lemme suivant :

Lemme 6 *Soit* $(\Sigma, \preceq)$ *une base de croyances partiellement préordonnées et soient* φ *et* ψ *deux formules propositionnelles. Alors, on a :*

$$\varphi \mid\sim^{(\Sigma,\preceq)}_{clex} \psi \ ssi \ \forall(\Sigma, \leq) \in Comp(\Sigma, \preceq), \varphi \mid\sim^{(\Sigma,\leq)}_{lex} \psi.$$

Preuve.

Par définition, $\varphi \mid\sim^{(\Sigma,\preceq)}_{clex} \psi$ si et seulement si $(\{\varphi\} \cup \Sigma, \preceq^{\varphi}) \vdash_{clex} \psi$.

Ceci est équivalent à dire que $\forall(\{\varphi\} \cup \Sigma, \leq^{\varphi}) \in Comp(\{\varphi\} \cup \Sigma, \preceq^{\varphi})$, on a $(\{\varphi\} \cup \Sigma, \leq^{\varphi}) \vdash_{lex} \psi$.

Or, les compatibles de $(\{\varphi\} \cup \Sigma, \preceq^{\varphi})$ sont tous de la forme $(\{\varphi\} \cup \Sigma, \leq^{\varphi})$ où $(\Sigma, \leq) \in Comp(\Sigma, \preceq)$.

Par conséquent, $\forall(\{\varphi\} \cup \Sigma, \leq^{\varphi}) \in Comp(\{\varphi\} \cup \Sigma, \preceq^{\varphi})$, $(\{\varphi\} \cup \Sigma, \leq^{\varphi}) \vdash_{lex} \psi$ si et seulement si $\forall(\Sigma, \leq) \in Comp(\Sigma, \preceq)$, $(\{\varphi\} \cup \Sigma, \leq^{\varphi}) \vdash_{lex} \psi$.

En conclusion, $\varphi \mid\sim^{(\Sigma,\preceq)}_{clex} \psi \ ssi \ \forall(\Sigma, \leq) \in Comp(\Sigma, \preceq), \varphi \mid\sim^{(\Sigma,\leq)}_{lex} \psi$.

■

En se basant sur ce dernier lemme, nous montrons que l'inférence lexicographique basée-compatibles est une inférence préférentielle.

Proposition 31 *L'inférence* $\mid\sim_{clex}$ *vérifie les propriétés du Système P.*

Preuve.

Vérifions les six propriétés du système P :

1. **Réflexivité :**
   Selon le lemme 6, $\alpha \mid\sim_{clex}^{(\Sigma,\preceq)} \alpha$ si et seulement si $\forall(\Sigma,\leq) \in Comp(\Sigma,\preceq), \alpha \mid\sim_{lex}^{(\Sigma,\leq)} \alpha$. Mais puisque $\mid\sim_{lex}$ vérifie les propriétés du système P, entre autres la réflexivité, on déduit que $\forall(\Sigma,\leq) \in Comp(\Sigma,\preceq), \alpha \mid\sim_{lex}^{(\Sigma,\leq)} \alpha$ c'est-à-dire $\alpha \mid\sim_{clex}^{(\Sigma,\preceq)} \alpha$.
2. **Équivalence logique gauche :**
   Étant données trois formules α, β et γ, supposons que $\alpha \mid\sim_{clex}^{(\Sigma,\preceq)} \beta$ et que $\models \alpha \Leftrightarrow \gamma$ et montrons que $\gamma \mid\sim_{clex}^{(\Sigma,\preceq)} \beta$.
   D'après le lemme 6, on a $\alpha \mid\sim_{clex}^{(\Sigma,\preceq)} \beta$ si et seulement si $\forall(\Sigma,\leq) \in Comp(\Sigma,\preceq), \alpha \mid\sim_{lex}^{(\Sigma,\leq)} \beta$. Comme $\mid\sim_{lex}$ satisfait les propriétés du système P et donc satisfait l'équivalence logique gauche, on dérive que $\forall(\Sigma,\leq) \in Comp(\Sigma,\preceq), \gamma \mid\sim_{lex}^{(\Sigma,\leq)} \beta$ c'est-à-dire $\gamma \mid\sim_{clex}^{(\Sigma,\preceq)} \beta$.
3. **Affaiblissement droit :**
   Soient trois formules α, β et γ. Supposons que $\alpha \mid\sim_{clex}^{(\Sigma,\preceq)} \beta$ et $\beta \models \gamma$ et prouvons que $\alpha \mid\sim_{clex}^{(\Sigma,\preceq)} \gamma$.
   Selon le lemme 6, $\alpha \mid\sim_{clex}^{(\Sigma,\preceq)} \beta$ si et seulement si $\forall(\Sigma,\leq) \in Comp(\Sigma,\preceq), \alpha \mid\sim_{lex}^{(\Sigma,\leq)} \beta$. Puisque $\mid\sim_{lex}$ satisfait les propriétés du système P y compris l'affaiblissement droit et puisque $\beta \models \gamma$, on conclut que $\forall(\Sigma,\leq) \in Comp(\Sigma,\preceq), \alpha \mid\sim_{lex}^{(\Sigma,\leq)} \gamma$ c'est-à-dire $\alpha \mid\sim_{clex}^{(\Sigma,\preceq)} \gamma$.
4. **Monotonie prudente**
   Prenons trois formules α, β et γ. Supposons que $\alpha \mid\sim_{clex}^{(\Sigma,\preceq)} \beta$ et $\alpha \mid\sim_{clex}^{(\Sigma,\preceq)} \gamma$. Montrons que $\alpha \wedge \beta \mid\sim_{clex}^{(\Sigma,\preceq)} \gamma$. D'après le lemme 6, on a $\forall(\Sigma,\leq) \in Comp(\Sigma,\preceq), \alpha \mid\sim_{lex}^{(\Sigma,\leq)} \beta$ et $\alpha \mid\sim_{lex}^{(\Sigma,\leq)} \gamma$. Étant donné que l'inférence $\mid\sim_{lex}$ vérifie les propriétés du système P à l'image de la monotonie prudente, on déduit que $\forall(\Sigma,\leq) \in Comp(\Sigma,\preceq), \alpha \wedge \beta \mid\sim_{lex}^{(\Sigma,\leq)} \gamma$. Autrement dit, $\alpha \wedge \beta \mid\sim_{clex}^{(\Sigma,\preceq)} \gamma$.
5. **Coupure :**
   Étant données trois formules α, β et γ, supposons que $\alpha \wedge \beta \mid\sim_{clex}^{(\Sigma,\preceq)} \gamma$ et $\alpha \mid\sim_{clex}^{(\Sigma,\preceq)} \beta$ et montrons que $\alpha \mid\sim_{clex}^{(\Sigma,\preceq)} \gamma$. En utilisant le lemme 6 et le fait que l'inférence $\mid\sim_{lex}$ vérifie les propriétés du système P telles que la coupure, on a $\forall(\Sigma,\leq) \in Comp(\Sigma,\preceq), \alpha \mid\sim_{lex}^{(\Sigma,\leq)} \gamma$ ce qui est équivalent à dire que $\alpha \mid\sim_{clex}^{(\Sigma,\preceq)} \gamma$.
6. **Le OU :**
   Soient trois formules α, β et γ. Supposons que $\alpha \mid\sim_{clex}^{(\Sigma,\preceq)} \gamma$ et $\beta \mid\sim_{clex}^{(\Sigma,\preceq)} \gamma$ et montrons que $\alpha \vee \beta \mid\sim_{clex}^{(\Sigma,\preceq)} \gamma$. D'après le lemme 6, nous avons alors que $\forall(\Sigma,\leq) \in Comp(\Sigma,\preceq$

$), \alpha \mid\sim_{lex}^{(\Sigma,\leq)} \gamma$ et $\beta \mid\sim_{lex}^{(\Sigma,\leq)} \gamma$. La propriété du OU est satisfaite par l'inférence $\mid\sim_{lex}$ d'où $\forall(\Sigma,\leq) \in Comp(\Sigma,\preceq), \alpha \vee \beta \mid\sim_{lex}^{(\Sigma,\leq)} \gamma$. Ceci signifie que $\alpha \vee \beta \mid\sim_{clex}^{(\Sigma,\preceq)} \gamma$.

■

Par ailleurs, l'inférence lexicographique basée-compatibles n'est pas une relation d'inférence rationnelle.

Proposition 32 *L'inférence $\mid\sim_{clex}$ ne vérifie pas la monotonie rationnelle.*

Preuve.

Il suffit de considérer encore une fois la base $(\Sigma,\preceq)$ et les trois formules propositionnelles φ, ψ et δ données dans l'exemple 50.

En effet, $CmpLex(\Sigma \cup \{\varphi\}, \preceq^{\varphi}) = \{A, B\}$ avec :

- $A = \{\varphi, p\}$,
- $B = \{\varphi, q, \neg r\}$.

On a $(\Sigma \cup \{\varphi\}, \preceq^{\varphi}) \vdash_{clex} \psi$ car $A \vee B \models \psi$, et donc $\varphi \mid\sim_{clex}^{(\Sigma,\preceq)} \psi$. Par contre, $(\Sigma \cup \{\varphi\}, \preceq^{\varphi}) \nvdash_{clex} \neg\delta$ parce que $A \not\models \neg\delta$ ce qui signifie que $\varphi \mid\not\sim_{clex}^{(\Sigma,\preceq)} \neg\delta$.

Par ailleurs, $CmpLex(\Sigma \cup \{\varphi \wedge \delta\}, \preceq^{\varphi\wedge\delta}) = \{A', B'\}$ avec :

- $A' = \{\varphi \wedge \delta, p\}$,
- $B' = \{\varphi \wedge \delta, q\}$.

On a $B' \not\models \psi$ ce qui implique que $(\Sigma \cup \{\varphi \wedge \delta\}, \preceq^{\varphi\wedge\delta}) \nvdash_{clex} \psi$ c'est-à-dire $\varphi \wedge \delta \mid\not\sim_{clex}^{(\Sigma,\preceq)} \psi$.

Donc, $\mid\sim_{clex}$ ne vérifie pas la monotonie rationnelle.

■

10.4.3 Propriétés de l'inférence démocratique

Nous commençons par donner une caractérisation sémantique de l'inférence démocratique et ce en définissant d'abord une relation de préférence entre les interprétions comme suit :

Définition 143 *Soit $(\Sigma,\preceq)$ une base de croyances partiellement préordonnées. Soient ω et ω' deux interprétations. Alors, ω est dite démocratiquement préférée à ω', noté $\omega \lhd_{demo} \omega'$, si et seulement si $V(\omega,\Sigma) \prec_{demo} V(\omega',\Sigma)$ où $V(\omega,\Sigma)$ désigne l'ensemble des formules de Σ satisfaites par ω.*

Lemme 7 *Soit $(\Sigma,\preceq)$ une base de croyances partiellement préordonnées et soit φ une formule cohérente. Alors, tous les éléments de $Min(Cons(\Sigma \cup \{\varphi\}, \preceq^{\varphi}), \prec_{demo})$ sont de la forme $B \cup \{\varphi\}$ où $B \in Min(\varphi - Cons(\Sigma,\preceq), \prec_{demo})$.*

Preuve.

Pour commencer, remarquons que n'importe quelle sous-base cohérente de $(\Sigma \cup \varphi, \preceq^{\varphi})$ préférée par rapport à $\prec_{demo}$ contient au moins la formule $\{\varphi\}$. En effet, étant données deux sous-bases cohérentes A et B telles que $A = \{\varphi\}$ et $\varphi \notin B$, il est clair que $A \prec_{demo} B$.

Maintenant, montrons que $\{\varphi\} \cup B \in Min(Cons(\Sigma \cup \varphi, \preceq^{\varphi}), \prec_{demo})$ si et seulement si B est préférée par rapport à $\prec_{demo}$ parmi les sous-bases φ-cohérentes de $(\Sigma, \preceq)$.

- Supposons que $\{\varphi\} \cup B \in Min(Cons(\Sigma \cup \varphi, \preceq^{\varphi}), \prec_{demo})$ et que $B \notin Min(\varphi - Cons(\Sigma, \preceq), \prec_{demo})$.
  Donc, il existe une sous-base A de $(\Sigma, \preceq)$ cohérente avec φ telle que $A \prec_{demo} B$ c'est-à-dire $\forall \delta \in B \setminus A, \exists \gamma \in A \setminus B$ tel que $\gamma \prec \delta$ et $B \setminus A \neq \emptyset$.
  Étant donné que $B \cup \{\varphi\} \setminus A \cup \{\varphi\} = B \setminus A$ et $A \cup \{\varphi\} \setminus B \cup \{\varphi\} = A \setminus B$, on déduit que $\forall \delta \in (B \cup \{\varphi\}) \setminus (A \cup \{\varphi\}), \exists \gamma \in (A \cup \{\varphi\}) \setminus (B \cup \{\varphi\})$ tel que $\gamma \prec \delta$ et $B \cup \{\varphi\} \setminus A \cup \{\varphi\} \neq \emptyset$. Autrement dit $A \cup \{\varphi\} \prec_{demo} B \cup \{\varphi\}$ ce qui est incohérent avec le fait que $\{\varphi\} \cup B \in Min(Cons(\Sigma \cup \varphi, \preceq^{\varphi}), \prec_{demo})$.
- Supposons maintenant que $B \in Min(\varphi - Cons(\Sigma, \preceq), \prec_{demo})$ et que $B \cup \{\varphi\} \notin Min(Cons(\Sigma \cup \{\varphi\}, \preceq^{\varphi}), \prec_{demo})$.
  Alors, il existe une sous-base $A \cup \{\varphi\}$ de $(\Sigma \cup \{\varphi\}, \preceq^{\varphi})$ telle que $A \cup \{\varphi\} \prec_{demo} B \cup \{\varphi\}$ avec A une sous-base de $(\Sigma, \preceq)$ cohérente avec φ.
  Ceci veut dire que $\forall \delta \in (B \cup \{\varphi\}) \setminus (A \cup \{\varphi\}), \exists \gamma \in (A \cup \{\varphi\}) \setminus (B \cup \{\varphi\})$ tel que $\gamma \prec \delta$ et $B \cup \{\varphi\} \setminus A \cup \{\varphi\} \neq \emptyset$.
  Mais puisque $B \cup \{\varphi\} \setminus A \cup \{\varphi\} = B \setminus A$ et $A \cup \{\varphi\} \setminus B \cup \{\varphi\} = A \setminus B$, on conclut que $\forall \delta \in B \setminus A, \exists \gamma \in A \setminus B$ tel que $\gamma \prec \delta$ et $B \setminus A \neq \emptyset$ c'est-à-dire $A \prec_{demo} B$ ce qui contredit l'hypothèse de départ qui suppose que $B \in Min(\varphi - Cons(\Sigma, \preceq), \prec_{demo})$.

■

Lemme 8 *Soit $(\Sigma, \preceq)$ une base de croyances partiellement préordonnées et soient φ et ψ deux formules propositionnelles. Alors,*

$$\varphi \mid\!\sim^{(\Sigma,\preceq)}_{demo} \psi \text{ ssi } Min(Mod(\varphi), \lhd^{(\Sigma,\preceq)}_{demo}) \subseteq Mod(\psi).$$

Preuve.

Par définition, $\varphi \mid\!\sim^{(\Sigma,\preceq)}_{demo} \psi$ si et seulement si $(\Sigma \cup \{\varphi\}, \preceq^{\varphi}) \vdash_{demo} \psi$ c'est-à-dire $\forall B' \in Min(Cons(\Sigma \cup \{\varphi\}, \preceq^{\varphi}), \prec_{demo}), B' \models \psi$.

Ceci équivaut à dire que $\bigcup_{B' \in Min(Cons(\Sigma \cup \{\varphi\}, \preceq^{\varphi}), \prec_{demo})} Mod(B') \subseteq Mod(\psi)$.

Montrons alors que :

$B' \in Min(Cons(\Sigma \cup \{\varphi\}, \preceq^{\varphi}), \prec_{demo})$ ssi $\omega \in Min(Mod(\varphi), \lhd^{(\Sigma,\preceq)}_{demo})$ où $V(\omega, \Sigma \cup \{\varphi\}) = B'$.

- Supposons que $B' \in Min(Cons(\Sigma \cup \{\varphi\}, \preceq^{\varphi}), \prec_{demo})$ et que $\omega \notin Min(Mod(\varphi), \lhd^{(\Sigma,\preceq)}_{demo}$

) avec $V(\omega, \Sigma \cup \{\varphi\}) = B'$.
Donc, $\exists \omega' \in Mod(\varphi)$ tel que $\omega' \lhd_{demo}^{(\Sigma,\preceq)} \omega$.
Par définition, ceci veut dire que $V(\omega', \Sigma) \prec_{demo}^{(\Sigma,\preceq)} V(\omega, \Sigma)$.
Par ailleurs, selon le lemme 7, B' est de la forme $\{\varphi\} \cup B$ où $B \in Min(\varphi - Cons(\Sigma, \preceq), \prec_{demo})$.
Clairement, B n'est rien d'autre que $V(\omega, \Sigma)$. De plus, $V(\omega', \Sigma)$ est une sous-base de $(\Sigma, \preceq)$ cohérente avec φ (car ω' qui la satisfait est un modèle de φ). Néanmoins, ceci est impossible car ça contredit le fait que $B \in Min(\varphi - Cons(\Sigma, \preceq), \prec_{demo})$.

– Supposons que $\omega \in Min(Mod(\varphi), \lhd_{demo}^{(\Sigma,\preceq)})$ avec $V(\omega, \Sigma \cup \{\varphi\}) = B'$ et que $B' \notin Min(Cons(\Sigma \cup \{\varphi\}, \preceq^{\varphi}), \prec_{demo})$.
Comme $\omega \in Mod(\varphi)$, on déduit que $V(\omega, \Sigma \cup \{\varphi\}) = \{\varphi\} \cup B$ où B est une sous-base de $(\Sigma, \preceq)$ cohérente avec φ.
Maintenant, $B' \notin Min(Cons(\Sigma \cup \{\varphi\}, \preceq^{\varphi}), \prec_{demo})$ implique qu'il existe une sous-base A de $(\Sigma, \preceq)$ cohérente avec φ telle que $A \prec_{demo} B$.
Puisque A est cohérente avec φ, soit ω' tel que ω' satisfait A et satisfait φ ce qui implique que $\omega' \in Mod(\varphi)$.
De plus, il est clair que $V(\omega, \Sigma) = B$ et $V(\omega', \Sigma) = A$ d'où $V(\omega', \Sigma) \prec_{demo}^{(\Sigma,\preceq)} V(\omega, \Sigma)$ ce qui par définition veut dire que $\omega' \lhd_{demo}^{(\Sigma,\preceq)} \omega'$.
Cependant, ceci contredit l'hypothèse initiale disant que $\omega \in Min(Mod(\varphi), \lhd_{demo}^{(\Sigma,\preceq)})$.
■

Ce dernier lemme nous permet de dire que l'inférence $\mid\!\sim_{demo}$ définit une inférence préférentielle.

Proposition 33 *L'inférence $\mid\!\sim_{demo}$ vérifie les propriétés du système P.*

Preuve.

La preuve de cette proposition découle directement du théorème de caractérisation de Kraus, Lehmann et Magidor de l'inférence préférentielle, et ce en se basant sur le lemme 8 et du fait que la relation $\lhd_{demo}^{(\Sigma,\preceq)}$ est clairement un ordre strict partiel. ■

En ce qui concerne la monotonie rationnelle, le fait qu'elle ne soit pas satisfaite par l'inférence démocratique découle directement du fait qu'elle n'est pas vérifiée par l'inférence relative à la préférence pour l'inclusion.

Proposition 34 *L'inférence $\mid\!\sim_{demo}$ ne vérifie pas la monotonie rationnelle.*

Preuve.

Comme $\mid\!\sim_{demo}$ est une généralisation de $\mid\!\sim_{incl}$, et puisque, selon [9], $\mid\!\sim_{incl}$ ne vérifie pas la monotonie rationnelle, on déduit que $\mid\!\sim_{demo}$ ne vérifie pas la monotonie rationnelle.
■

10.4.4 Propriétés de l'inférence relative à la préférence pour l'inclusion basée-compatibles

A l'instar de l'inférence lexicographique basée-compatibles, nous introduisons dans un premier temps une caractérisation de l'inférence relative à la préférence pour l'inclusion basée-compatibles par rapport aux compatibles de la base de croyances partiellement préordonnées en question comme le montre le lemme suivant :

Lemme 9 *Soit* $(\Sigma, \preceq)$ *une base de croyances partiellement préordonnées et soient* φ *et* ψ *deux formules propositionnelles. Alors, nous montrons dans un premier temps que :*

$$\varphi \mid\!\sim_{cincl}^{(\Sigma,\preceq)} \psi \text{ ssi } \forall (\Sigma, \leq) \in Comp(\Sigma, \preceq), \varphi \mid\!\sim_{incl}^{(\Sigma,\leq)} \psi.$$

Preuve.

Par définition, $\varphi \mid\!\sim_{cincl}^{(\Sigma,\preceq)} \psi$ si et seulement si $(\{\varphi\} \cup \Sigma, \preceq^{\varphi}) \vdash_{cincl} \psi$.

Alors, d'une manière équivalente, on a $\forall (\{\varphi\} \cup \Sigma, \leq^{\varphi}) \in Comp(\{\varphi\} \cup \Sigma, \preceq^{\varphi})$, $(\{\varphi\} \cup \Sigma, \leq^{\varphi}) \vdash_{incl} \psi$.

Tous les compatibles de $(\{\varphi\} \cup \Sigma, \preceq^{\varphi})$ ont la forme $(\{\varphi\} \cup \Sigma, \leq^{\varphi})$ avec $(\Sigma, \leq) \in Comp(\Sigma, \preceq)$.

Donc, on déduit que $\forall (\{\varphi\} \cup \Sigma, \leq^{\varphi}) \in Comp(\{\varphi\} \cup \Sigma, \preceq^{\varphi})$, $(\{\varphi\} \cup \Sigma, \leq^{\varphi}) \vdash_{incl} \psi$ si et seulement si $\forall (\Sigma, \leq) \in Comp(\Sigma, \preceq)$, $(\{\varphi\} \cup \Sigma, \leq^{\varphi}) \vdash_{incl} \psi$.

Par conséquent, $\varphi \mid\!\sim_{cincl}^{(\Sigma,\preceq)} \psi$ *ssi* $\forall (\Sigma, \leq) \in Comp(\Sigma, \preceq), \varphi \mid\!\sim_{incl}^{(\Sigma,\leq)} \psi$.

■

Ainsi, l'inférence relative à la préférence pour l'inclusion basée-compatibles est une relation préférentielle.

Proposition 35 *L'inférence* $\mid\!\sim_{cincl}$ *vérifie les propriétés du Système P.*

Preuve.

La démonstration de cette proposition est similaire à celle de la proposition 31 et ce en se basant cette fois ci sur le lemme 9. ■

Proposition 36 *L'inférence* $\mid\!\sim_{cincl}$ *ne vérifie pas la monotonie rationnelle.*

L'inférence relative à la préférence pour l'inclusion basée-compatibles n'est pas une relation rationnelle.

Preuve.

A l'image de l'inférence $\mid\!\sim_{demo}$, l'inférence $\mid\!\sim_{cincl}$ est une généralisation de $\mid\!\sim_{incl}$ dans le cadre de bases de croyances partiellement préordonnées. Or, l'inférence $\mid\!\sim_{incl}$ ne vérifie pas

la monotonie rationnelle [9]. De ce fait, on conclut que $\mid\!\sim_{cincl}$ ne vérifie pas la monotonie rationnelle. ■

10.4.5 Propriétés des inférences possibilistes faible et forte

Une caractérisation sémantique de la préférence $\vdash_{wpos}$ a été proposée dans [5] et ce en se basant sur la relation de préférence suivante entre deux interprétations.

Définition 144 *Soit $(\Sigma, \preceq)$ une base de croyances partiellement préordonnées. Soient ω et ω' deux interprétations. Alors, ω est dite préférée à ω' par rapport à la préférence best-out, noté $\omega \lhd_{wbo}^{(\Sigma,\preceq)} \omega'$, si et seulement si $\forall \delta \in F(\omega, \Sigma), \exists \gamma \in F(\omega', \Sigma)$, tel que $\gamma \prec \delta$, où $F(\omega, \Sigma)$ désigne l'ensemble des formules de Σ falsifiées par ω.*

Lemme 10 *Soit $(\Sigma, \preceq)$ une base de croyances partiellement préordonnées et soit φ une formule cohérente. Alors, tous les éléments de $Min(Cons(\Sigma \cup \{\varphi\}, \preceq^{\varphi}), \prec_{wbo})$ sont de la forme $B \cup \{\varphi\}$ où $B \in Min(\varphi - Cons(\Sigma, \preceq), \prec_{wbo})$.*

Preuve.

Tout d'abord, il est à noter que toute sous-base cohérente de $(\Sigma \cup \varphi, \preceq^{\varphi})$ préférée par rapport à $\prec_{wbo}$ contient au moins la formule $\{\varphi\}$ c'est-à-dire de la forme $\{\varphi\} \cup B$ où B est une sous-base cohérente de $(\Sigma, \preceq)$ cohérente avec φ. En effet, étant données deux sous-bases cohérentes A et B telles que $A = \{\varphi\}$ et $\varphi \notin B$, il est clair que $A \prec_{wbo} B$.

Montrons ensuite que $\{\varphi\} \cup B \in Min(Cons(\Sigma \cup \varphi, \preceq^{\varphi}), \prec_{wbo})$ si et seulement si B est préférée par rapport à $\prec_{wbo}$ parmi les sous-bases φ-cohérentes de $(\Sigma, \preceq)$.

- Supposons que $\{\varphi\} \cup B \in Min(Cons(\Sigma \cup \varphi, \preceq^{\varphi}), \prec_{wbo})$ et que $B \notin Min(\varphi - Cons(\Sigma, \preceq), \prec_{wbo})$.
  Le fait que $B \notin Min(\varphi - Cons(\Sigma, \preceq), \prec_{wbo})$ implique qu'il existe une sous-base cohérente A de $(\Sigma, \preceq)$ qui est cohérente avec φ telle que $A \prec_{wbo} B$ par rapport à $(\Sigma, \preceq)$.
  Par définition, ceci revient à dire que $\forall \delta \notin A, \exists \gamma \notin B$, tel que $\gamma \prec \delta$.
  Autrement dit, $\forall \delta \in \Sigma \setminus A, \exists \gamma \in \Sigma \setminus B$, tel que $\gamma \prec \delta$.
  Comme $\varphi \notin \Sigma$, on a $\Sigma \cup \{\varphi\} \setminus A \cup \{\varphi\} = \Sigma \setminus A$ et $\Sigma \cup \{\varphi\} \setminus B \cup \{\varphi\} = \Sigma \setminus B$.
  On déduit alors que $\forall \delta \in \Sigma \cup \{\varphi\} \setminus A \cup \{\varphi\}, \exists \gamma \in \Sigma \cup \{\varphi\} \setminus B \cup \{\varphi\}$, tel que $\gamma \prec \delta$.
  Ceci équivaut à dire que $A \cup \{\varphi\} \prec_{wbo} B \cup \{\varphi\}$ par rapport à $(\Sigma \cup \{\varphi\}, \preceq^{\varphi})$ ce qui est impossible car $\{\varphi\} \cup B \in Min(Cons(\Sigma \cup \varphi, \preceq^{\varphi}), \prec_{wbo})$.
- Supposons que $B \in Min(\varphi - Cons(\Sigma, \preceq), \prec_{wbo})$ et que $\{\varphi\} \cup B \notin Min(Cons(\Sigma \cup \varphi, \preceq^{\varphi}), \prec_{wbo})$.
  $\{\varphi\} \cup B \notin Min(Cons(\Sigma \cup \varphi, \preceq^{\varphi}), \prec_{wbo})$ implique qu'il existe une sous-base cohérente A' de $(\Sigma \cup \varphi, \preceq^{\varphi})$ telle que $A' \prec_{wbo} \{\varphi\} \cup B$.

Puisque $A' \prec_{wbo} \{\varphi\} \cup B$, on conclut que A' contient au moins la formule φ c'est-à-dire $A' = \{\varphi\} \cup A$ où A est une sous-base cohérente de $(\Sigma, \preceq)$ cohérente avec φ.
Par conséquent, on a $\forall \delta \in \Sigma \cup \{\varphi\} \setminus A \cup \{\varphi\}, \exists \gamma \in \Sigma \cup \{\varphi\} \setminus B \cup \{\varphi\}$, tel que $\gamma \prec \delta$. Mais comme $\Sigma \cup \{\varphi\} \setminus A \cup \{\varphi\} = \Sigma \setminus A$ et $\Sigma \cup \{\varphi\} \setminus B \cup \{\varphi\} = \Sigma \setminus B$ car $\varphi \notin \Sigma$, on déduit que $\forall \delta \in \Sigma \setminus A, \exists \gamma \in \Sigma \setminus B$, tel que $\gamma \prec \delta$.
En conclusion, $A \prec_{wbo} B$ ce qui est incohérent avec le fait que $B \in Min(\varphi - Cons(\Sigma, \preceq), \prec_{wbo})$.

■

Lemme 11 *Soit $(\Sigma, \preceq)$ une base de croyances partiellement préordonnées et soient φ et ψ deux formules propositionnelles. Alors,*

$$\varphi \mid\sim_{wpos}^{(\Sigma,\preceq)} \psi \text{ ssi } Min(Mod(\varphi), \lhd_{wbo}^{(\Sigma,\preceq)}) \subseteq Mod(\psi).$$

Preuve.

Tout d'abord, on a $\omega \lhd_{wbo}^{(\Sigma,\preceq)} \omega'$ si et seulement si $V(w, \Sigma) \prec_{wbo} V(w, \Sigma)$. En effet, par définition, $\omega \lhd_{wbo}^{(\Sigma,\preceq)} \omega'$ si et seulement si $\forall \delta \in F(\omega, \Sigma), \exists \gamma \in F(\omega', \Sigma)$, tel que $\gamma \prec \delta$. Ceci est équivalent à dire que $\forall \delta \notin V(\omega, \Sigma), \exists \gamma \notin V(\omega', \Sigma)$, tel que $\gamma \prec \delta$ c'est-à-dire $V(w, \Sigma) \prec_{wbo} V(w, \Sigma)$.

En se basant sur cette dernière propriété ainsi que sur le lemme 10, cette preuve est similaire par exemple à celle du lemme 5 en remplaçant la préférence $\prec_{plex}$ par la préférence $\prec_{wbo}$.

■

Proposition 37 *L'inférence $\mid\sim_{spos}$ vérifie les propriétés du système P.*

Preuve.

La preuve de cette proposition découle directement du théorème de caractérisation de Kraus, Lehmann et Magidor de l'inférence préférentielle, et ce en se basant sur le lemme 11. ■

De manière similaire, nous montrons que l'inférence possibiliste forte est une relation préférentielle.

Proposition 38 *L'inférence $\mid\sim_{spos}$ vérifie les propriétés du système P.*

Quant à la monotonie rationnelle, elle n'est pas vérifiée par l'inférence possibiliste faible.

Proposition 39 *L'inférence $\mid\sim_{wpos}$ ne vérifie pas la monotonie rationnelle.*

Preuve.

Considérons le contre-exemple suivant :

Exemple 51 *Soit $(\Sigma, \preceq)$ une base de croyances partiellement préordonnées telle que $\Sigma = \{a, \neg a, \neg b\}$ avec $a \prec \neg b$.*

Considérons 3 formules φ, ψ et δ telles que :

- $\varphi = c$,
- $\psi = \neg b$,
- $\delta = \neg a \vee b$.

La base $\Sigma \cup \{c\}$ admet deux sous-bases maximales cohérentes :

- $A = \{a, \neg b, c\}$,
- $B = \{\neg a, \neg b, c\}$.

Puisque $A \sim_{wbo} B$, on a $Min(Cons(\Sigma \cup \{\varphi\}, \preceq^{\varphi}), \prec_{wbo}) = \{A, B\}$.

Comme $A \vee B \models \neg b$, $(\Sigma \cup \{\varphi\}, \preceq^{\varphi}) \vdash_{wbo} \psi$ ce qui signifie que $\varphi \mid\sim_{wpos}^{(\Sigma,\preceq)} \psi$.

En ce qui concerne $\neg\delta$, on a $B \not\models \neg(\neg a \vee b)$ d'où $(\Sigma \cup \{\varphi\}, \preceq^{\varphi}) \nvdash_{wpos} \neg\delta$. Autrement dit, $\varphi \not\mid\sim_{wpos}^{(\Sigma,\preceq)} \neg\delta$.

Prenons maintenant la base $\Sigma \cup \{\varphi \wedge \delta\}$. Cette dernière admet deux sous-bases maximales cohérentes :

- $A' = \{a, c \wedge (\neg a \vee b)\}$,
- $B' = \{\neg a, \neg b, c \wedge (\neg a \vee b)\}$.

Comme $A' \sim_{wbo} B'$, on déduit que $Min(Cons(\Sigma \cup \{\varphi \wedge \delta\}, \preceq^{\varphi\wedge\delta}), \prec_{wbo}) = \{A', B'\}$.

$B' \not\models \neg b$ ce qui veut dire que $(\Sigma \cup \{\varphi \wedge \delta\}, \preceq^{\varphi\wedge\delta}) \nvdash_{wpo} \psi$ et donc $\varphi \wedge \delta \not\mid\sim_{wpo}^{(\Sigma,\preceq)} \psi$.

D'où, $\mid\sim_{wpo}$ ne vérifie pas la monotonie rationnelle.

■

Il est de même, la monotonie rationnelle n'est pas satisfaite par l'inférence best-out forte.

Proposition 40 *L'inférence $\mid\sim_{spos}$ ne vérifie pas la monotonie rationnelle.*

Preuve.

Il suffit de le vérifier sur la base et les formules de l'exemple 51. ■

10.5 Conclusion

Nous avons présenté dans ce chapitre les résultats d'une étude comparative que nous avons menée autour de relations d'inférence à partir de bases de croyances partiellement préordonnées. Les relations d'inférence concernées par cette étude sont les relations d'inférence lexicographique partielle et basée-compatibles que nous avons introduites dans le chapitre 9, la relation d'inférence démocratique, la relation d'inférence relative à la préférence pour l'inclusion basée-compatibles ainsi que les relations d'inférence possibilistes faible et forte rappelées dans le chapitre 4.

D'après notre étude, toutes ces relations sont heureusement préférentielles. Néanmoins, aucune d'entre elles ne satisfait la monotonie rationnelle, même celles qui la satisfont dans le cadre de bases de croyances stratifiées, à savoir les extensions de l'inférence lexicographique et possibiliste. Cependant, nous ne considérons pas le dernier résultat comme étant un résultat négatif car la propriété de monotonie rationnelle reste discutable. Par exemple, elle a été proposée dans l'optique de résoudre le problème de non-pertinence alors que finalement, elle ne le résout que partiellement.

En ce qui concerne les résultats de complexité, on constate que ces relations d'inférence sont typiquement au deuxième niveau de la hiérarchie polynomiale, tandis que dans le cadre totalement préordonné, elles sont en général au premier niveau de cette hiérarchie. Ce passage d'un niveau dans la hiérarchie polynomiale n'est rien d'autre que le prix à payer pour gagner plus en termes de flexibilité.

Enfin, on peut dire que ces relations d'inférence diffèrent essentiellement par rapport au critère de prudence. Ainsi, les plus prudentes sont les inférences possibilistes forte ensuite faible, et la moins prudente, ou d'une manière équivalente, la plus productive, est l'inférence lexicographique basée-compatibles. Cette dernière est plus productive que l'inférence relative à la préférence de l'inclusion basée-compatibles, qui est plus productive à son tour que l'inférence démocratique. Par ailleurs, l'inférence lexicographique partielle se trouve entre l'inférence démocratique et l'inférence lexicographique basée-compatibles, en étant incomparable à l'inférence relative à la préférence de l'inclusion basée-compatibles. Donc, en général, ces résultats de prudence sont fidèles à ceux connus dans le cadre totalement préordonné, hormis la relation entre l'inférence lexicographique partielle et l'inférence de l'inclusion basée-compatibles, qui sont incomparables par rapport à des bases de croyances partiellement préordonnées, alors que la première est plus productive que la seconde dans le cas de bases de croyances totalement préordonnées.

Chapitre 11

Gestion de l'incohérence en détection d'intrusions

11.1 Introduction

La sécurisation d'un système d'informations, qui est parmi les enjeux majeurs de nos jours, revient à garantir la confidentialité et l'intégrité des données de ce système, ainsi que la disponibilité de ses services. Parmi les mécanismes mis au point en vue de sécuriser les systèmes d'informations, on peut distinguer les systèmes de prévention tels que l'authentification, dont l'objectif est de prouver l'identité des utilisateurs, le contrôle d'accès, qui consiste à définir les droits accordés aux utilisateurs sur les données et les pare-feux dont le rôle est de filtrer l'accès aux services du système d'informations vis-à-vis du monde extérieur.

Cependant, ces mécanismes souffrent de certaines limitations. En effet, les systèmes informatiques présentent souvent des vulnérabilités, c'est-à-dire des failles de conception, d'implémentation et de configuration, ce qui permet à des attaquants de contourner les mécanismes de prévention. De plus, un certain nombre de ces systèmes se focalisent sur la protection des attaques extérieures, alors qu'une grande partie des attaques proviennent de l'intérieur. Étant donné que les systèmes de prévention ne suffisent pas, une seconde couche de sécurisation s'avère nécessaire, à savoir la détection d'intrusions.

La détections d'intrusions vise à détecter les attaques contre le système surveillé. Néanmoins, à son tour, elle souffre de certains problèmes. En effet, certaines attaques peuvent passer inaperçues, et dans ce cas, on parle de faux négatifs. En revanche, certaines alertes sont générées par rapport à des attaques qui n'ont pas eu lieu, ce qui correspond à des faux positifs.

Par ailleurs, la détection d'intrusion coopérative, qui implique plusieurs systèmes de détection d'intrusions ou IDSs (pour Intrusion Detection System) et d'autres systèmes (tels que les analyseurs réseaux, scanners de vulnérabilités, etc), offre de nombreux avantages. En effet, elle permet une vision globale du système surveillé, et donc des points de vue complémentaires. Toutefois, étant donné que les systèmes utilisés dans la coopération ne sont pas totalement fiables, souvent des conflits et des incohérences surviennent. De ce fait, il est indispensable de gérer ces incohérences afin d'exploiter cette coopération.

Nous introduisons dans ce chapitre une nouvelle approche de corrélation d'alertes dans un contexte de détection d'intrusion coopérative qui permet de corréler les informations issues des différents systèmes utilisés, en vue de réduire le nombre d'alertes, en particulier les faux positifs.

L'idée de cette approche de corrélation est de raisonner, sans trivialisation, à partir des informations issues des différents systèmes utilisés conjointement dans le processus de surveillance. Par ailleurs, étant donné que les informations manipulées en détection d'intrusion sont de nature structurée (souvent exprimées en XML), nous proposons de les représenter en logiques de description. Les logiques de description ou DLs (pour Description Logics) sont bien adaptées pour représenter des informations structurées, et de plus,

elles garantissent la décidabilité du raisonnement. Avant de décrire notre approche de corrélation, nous rappelons le principe de la détection d'intrusions. Ensuite, nous donnons une introduction aux logiques de description.

11.2 Principe de la détection d'intrusions

La détection d'intrusions est apparue au début des années 80, suite aux travaux de Anderson [1] et à ceux de Denning [44]. Un système de détection d'intrusions (IDS pour Intrusion Detection System) est destiné à détecter automatiquement des actions non autorisées ou malicieuses menées par des personnes internes ou externes au système d'informations surveillé.

Un IDS regroupe deux types de composants : un capteur et un analyseur. Alors que le capteur est chargé de collecter les données où se manifeste l'activité des utilisateurs, l'analyseur se charge de l'analyse, comme son nom l'indique, des événements produits par les capteurs afin de déterminer si une activité est intrusive ou non, autrement dit s'il s'agit d'une attaque.

Les IDSs peuvent être classifiés en général par rapport à deux critères, à savoir la nature des données analysées et la méthode d'analyse [42].

Ainsi, relativement à la source des données, on distingue trois catégories d'IDSs :
- les **IDSs réseaux ou NIDS** (pour Network-Based IDS) munis d'un dispositif matériel qui permet de capturer le trafic du réseau qu'ils surveillent,
- **les IDS systèmes ou HIDS** (pour Host-based IDS) qui analysent des données d'audit produites par le système d'exploitation des hôtes sur lesquels ils sont installés,
- **les IDS applicatifs** dont la source est les audits applicatifs. Donc, les données à analyser sont produites directement par une application, par exemple un serveur web, ftp, de base de données.

En ce qui concerne les méthodes d'analyse, on distingue d'une manière générale, deux catégories : les approches par signatures et les approches comportementales.
- Les **IDSs basés-approche par signatures** nécessitent une connaissances à priori sur les attaques détectées. Autrement dit, tout ce qui n'est pas explicitement interdit est autorisé. L'approche la plus fréquemment utilisée s'appuie sur des algorithmes de pattern matching. Les motifs recherchés dans les sources de données (chaîne de caractères, taille anormale, accès à un fichier système), censés caractériser des attaques sont appelés des signatures. D'autres approches ont été proposées, par exemple, Mé [75] propose d'appliquer des algorithmes génétiques pour analyser les audits systèmes. Lunt et al [72] introduisent une approche basée sur des systèmes

experts. Parmi de tels IDSs, citons le NIDS Snort[1] qui est actuellement le plus utilisé en détection d'intrusions.

– Pour les **IDSs basés-approche comportementale**, on définit préalablement un modèle de comportement normal ou autorisé du système surveillé. Au cours de la surveillance, le comportement courant de l'entité est mesuré, et toute déviation significative du comportement courant par rapport au comportement de référence génère une alerte. Contrairement à l'approche par signature, tout ce qui n'est pas explicitement autorisé est interdit. La construction du modèle de référence se fait le plus souvent par apprentissage. Debar [41] propose par exemple d'utiliser des réseaux de neurones, Forrest et al [59] proposent pour leur part d'appliquer une approche immunologique. Des méthodes statistiques ont aussi été suggérées [58]. Le modèle peut aussi être construit à partir de spécifications de comportement d'applications ou par spécification de politiques de sécurité, comme dans l'approche de Zimmerman et al. [104].

11.3 Introduction aux logiques de description

Les logiques de description sont une famille de formalismes de représentation de connaissances qui permettent de représenter les connaissance d'un domaine d'application d'une manière structurée et formelle. Ces logiques sont issues de modèles graphiques de représentation de connaissances, notamment les réseaux sémantiques. En général, dans un réseau sémantique, on distingue des concepts (dénotés par des nœuds génériques), des individus (dénotés par des nœuds d'individus), des liens classe-fille/classe-mère et des liens de propriétés. En utilisant les liens classe-fille/classe-mère, les concepts peuvent être classés dans une hiérarchie, et en utilisant les liens de propriétés, des propriétés peuvent être assignées aux concepts. Toutefois, ces liens sont dépourvus de sémantique. De ce fait, ils peuvent être interprétés de différentes manières. Afin de pallier cette ambiguïté, d'autre formalismes ont été développés tels que les réseaux d'héritage structuré qui ont été implémentés dans le système KL-ONE [20]. Par la suite, KL-ONE a été doté d'une sémantique ce qui a fixé de manière précise la signification de ses constructeurs graphiques, et a conduit à la définition de la première logique de description [70], appelée aussi à cette époque langage terminologique, langage conceptuel ou encore langage basés KL-ONE.

D'un point de vue pratique, les logiques de description sont assez prometteuses. En fait, elles ont été utilisées dans l'implémentation de nombreuses applications dans divers domaines. Sans être exhaustifs, nous citons par exemple le Web Sémantique pour la représentation d'ontologies et la recherche d'information basée sur la logique, la médecine où l'objectif est la construction et la maintenance de très grandes ontologies médicales, les librairies numériques, les systèmes d'informations basés web, le traitement du langage

[1] http://www.snort.org/

naturel, la gestion de bases de données, le génie logiciel, etc.

11.3.1 Logique de description $\mathcal{AL}$

Une base de connaissances basée sur la logique de description comprend deux composants, la TBox et la ABox. La TBox introduit la terminologie, c'est-à-dire le vocabulaire du domaine d'une application. Quant à la ABox, elle contient des assertions par rapport à des individus nommés en termes de ce vocabulaire.

Le vocabulaire consiste en des concepts, qui dénotent des ensembles d'individus, et des rôles qui désignent des relations binaires entre des individus. En plus des concepts atomiques et des rôles atomiques (les noms des concepts et des règles), tous les systèmes DL permettent de construire des descriptions de concepts et de rôles plus complexes.

Les descriptions élémentaires sont les concepts atomiques et les rôles atomiques à partir desquels des descriptions complexes peuvent être générées via les constructeurs de concepts et les constructeurs de rôles. Dans ce qui suit, on utilisera les lettres A et B pour designer des concepts atomiques, la lettre R pour un rôle atomique et les lettres C et D pour les descriptions de concepts. Les logiques de description sont distinguées par rapport aux constructeurs qu'elles utilisent. La logique de description $\mathcal{AL}$ (pour Attribuitive Langage) a été introduite dans [87] comme étant la logique de description la moins expressive ayant un intérêt pratique. Les descriptions de concepts dans la logique $\mathcal{AL}$ sont générées selon la règle syntaxique suivante :

$$
\begin{array}{lll}
C, D \rightarrow & A \mid & \text{(concept atomique)} \\
& \top \mid & \text{(concept universel)} \\
& \bot \mid & \text{(concept bottom)} \\
& \neg A \mid & \text{(négation atomique)} \\
& C \sqcap D \mid & \text{(intersection)} \\
& \forall R.C \mid & \text{(restriction de valeur)} \\
& \exists R.\top & \text{(quantificateur existentiel limité)}
\end{array}
$$

Afin d'illustrer cette règle, prenons l'exemple suivant qui nous donne une idée sur ce qui peut être exprimé dans la logique $\mathcal{AL}$:

Exemple 52 *Soient* `Person` *et* `Female` *deux concepts atomiques. Donc,* `Person`$\sqcap$`Female` *et* `Person` $\sqcap \neg$`Female` *sont des concepts décrivant intuitivement les personnes qui sont femelle et les personnes qui ne sont pas femelle. En outre, étant donné un rôle atomique* `hasChil`*, on peut construire les concepts* `Person` $\sqcap \exists$`hasChild`.$\top$ *et* `Person` $\sqcap \forall$ `hasChild.Female` *dénotant respectivement les personnes qui ont au moins un enfant et les personnes dont tous les enfants sont une femelle. Enfin, en utilisant le concept* $\bot$*, on peut également décrire les personnes qui n'ont pas d'enfants par* `Person` $\sqcap \forall$`hasChild`.$\bot$.

En vue de définir une sémantique formelle des concepts $\mathcal{AL}$, on considère une interprétation $\mathcal{I}$ définie par la donnée d'un ensemble non vide $\Delta^{\mathcal{I}}$ (le domaine d'interprétation) et d'une fonction d'interprétation, qui assigne à chaque concept atomique A un ensemble $A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$ et à chaque rôle atomique R une relation binaire $R^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$. Une fonction d'interprétation est étendue aux descriptions de concepts par la définition inductive suivante :

$$\begin{array}{rcl} \top^{\mathcal{I}} & = & \Delta^{\mathcal{I}} \\ \bot^{\mathcal{I}} & = & \emptyset \\ (\neg A)^{\mathcal{I}} & = & \Delta^{\mathcal{I}} - A^{\mathcal{I}} \\ (C \sqcap D)^{\mathcal{I}} & = & C^{\mathcal{I}} \cap D^{\mathcal{I}} \\ (\forall R.C)^{\mathcal{I}} & = & \{a \in \Delta^{\mathcal{I}} | \forall b, (a,b) \in R^{\mathcal{I}} \rightarrow b \in C^{\mathcal{I}}\} \\ (\exists R.\top)^{\mathcal{I}} & = & \{a \in \Delta^{\mathcal{I}} | \exists b, (a,b) \in R^{\mathcal{I}}\} \end{array}$$

Deux concepts C et D sont dits équivalents, $C \equiv D$, si et seulement si $C^{\mathcal{I}} = D^{\mathcal{I}}$ pour toute interprétation $\mathcal{I}$. Par exemple, on peut aisément vérifier que les concepts $\forall$`hasChild.Female`$\sqcap\forall$`hasChild.Student` et $\forall$ `hasChild.(Female`$\sqcap$`Student`) sont équivalents.

11.3.2 Au delà de la logique $\mathcal{AL}$

D'autres logiques, plus expressives, peuvent être définies en rajoutant d'autres constructeurs à la logique $\mathcal{AL}$ à savoir :

– L'union de concepts, désignée par la lettre $\mathcal{U}$, notée par $C \sqcup D$ et interprétée par :

$$(C \sqcup D)^{\mathcal{I}} = C^{\mathcal{I}} \cup D^{\mathcal{I}}.$$

– La quantification existentielle complète, désignée par la lettre $\mathcal{E}$, notée par $\exists R.C$ et interprétée par :

$$(\exists R.C)^{\mathcal{I}} = \{a \in \Delta^{\mathcal{I}} | \exists b, (a,b) \in R^{\mathcal{I}} \wedge b \in C^{\mathcal{I}}\}.$$

– Les restrictions de nombres, désignées par la lettre $\mathcal{N}$ et notées par $\geq nR$ (restriction au moins) et par $\leq nR$ (restriction au plus) où n représente un entier positif. Ces restrictions de valeurs sont interprétées respectivement comme suit :

$$(\geq nR)^{\mathcal{I}} = \{a \in \Delta^{\mathcal{I}} : |\{b|(a,b) \in R^{\mathcal{I}}\}| \geq n\},$$

et

$$(\leq nR)^{\mathcal{I}} = \{a \in \Delta^{\mathcal{I}} : |\{b|(a,b) \in R^{\mathcal{I}}\}| \leq n\}.$$

– La négation de concepts arbitraires, désignée par la lettre (C), notée par $\neg C$ et interprétée de la façon suivante :

$$(\neg C)^{\mathcal{I}} = \Delta^I - (C)^{\mathcal{I}}.$$

Exemple 53 *Avec ces nouveaux constructeurs, on peut par exemple décrire les personnes ayant pas plus d'un enfant ou bien au moins trois enfants dont un est une femelle comme suit :*

$$\texttt{Person} \sqcap (\leq 1\ \texttt{hasChild} \sqcup (\geq 3\ \texttt{hasChild} \sqcap \exists \texttt{hasChild.Female})).$$

Étendre la logique $\mathcal{AL}$ par n'importe quel sous ensemble des constructeurs précédents résulte en une nouvelle logique désignée par une chaîne de caractères de la forme :

$$\mathcal{AL}[\mathcal{U}][\mathcal{E}][\mathcal{N}][\mathcal{C}],$$

où une lettre dans l'appellation reflète la présence du constructeur correspondant. Par exemple, $\mathcal{ALEN}$ est l'extension de $\mathcal{AL}$ par la quantification existentielle complète et les restrictions de nombres.

D'un point de vue sémantique, on a $C \sqcup D \equiv \neg(\neg C \sqcap \neg D)$ et $\exists R.C \equiv \neg\forall R.\neg C$. Par conséquent, l'union et la quantification universelle complète peuvent être exprimées via la négation et vice versa. Ainsi, on utilisera la lettre $\mathcal{C}$ au lieu des lettres $\mathcal{UE}$ dans les noms des logiques. Par exemple, on écrit $\mathcal{ALCN}$ au lieu de $\mathcal{ALUEN}$.

11.3.3 Logiques de description et la logique des prédicats du premier ordre

La sémantique des concepts identifie les logiques de description comme étant des fragments de la logique des prédicats du premier ordre. En fait, puisqu'une interprétation $\mathcal{I}$ assigne à chaque concept atomique et à chaque rôle une relation unaire et une relation binaire respectivement, les concepts et les rôles peuvent être vus comme étant des prédicats unaires et binaires. Donc, n'importe quel concept C peut être traduit par une formule logique $\phi_C(x)$ avec une seule variable libre x telle que pour toute interprétation $\mathcal{I}$, l'ensemble des éléments de $\Delta^{\mathcal{I}}$ qui satisfont $\phi_C(x)$ est exactement $C^{\mathcal{I}}$:

– Un concept atomique A est traduit par la formule $A(x)$,
– Les constructeurs d'intersection, d'union et de négation sont traduits par la conjonction, la disjonction et la négation respectivement,
– si $\phi_C(x)$ est la traduction de C et R est un rôle atomique, alors la restriction de valeurs et la quantification universelle correspondent aux formules :

$$\phi_{\exists R.C}(y) \equiv \exists x.R(y,x) \wedge \phi_C(x)$$

$$\phi_{\forall R.C}(y) \equiv \forall x.R(y,x) \rightarrow \phi_C(x)$$

où y est une nouvelle variable,

– Enfin, les restrictions de nombres sont exprimées par les formules :

$$\phi_{\geq nR}(x) \equiv \exists y_1, \ldots, y_n.R(x,y_1) \wedge \ldots R(x,y_n) \wedge \bigwedge_{i<j} y_i \neq y_j$$

$$\phi_{\leq nR}(x) \equiv \forall y_1, \ldots, y_{n+1}.R(x,y_1) \wedge \ldots R(x,y_{n+1}) \rightarrow \bigvee_{i<j} y_i = y_j$$

De ce fait, on peut dire qu'une syntaxe spéciale est dénuée d'intérêts. Néanmoins, les translations précédentes montrent, en particulier pour la restriction de nombres, que la syntaxe à variables libres des logiques de description est vraiment plus concise. Ceci facilite également le développement d'algorithmes.

11.3.4 Les terminologies TBox

Une terminologie ou TBox en abrégé est un ensemble d'axiomes terminologiques. En général, les axiomes terminologiques sont de la forme

$$C \sqsubseteq D \ (R \sqsubseteq S)$$

ou

$$C \equiv D \ (R \equiv S)$$

où C, D sont des concepts (et R, S sont des rôles). Les axiomes du premier type sont appelés inclusions alors que ceux du second type sont dits équivalence.

Sémantiquement, une interprétation $\mathcal{I}$ satisfait une inclusion $C \sqsubseteq D$ si et seulement si $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ et satisfait une équivalence $C \equiv D$ si et seulement si $C^{\mathcal{I}} = D^{\mathcal{I}}$. Si Σ est un ensemble d'axiomes, alors $\mathcal{I}$ satisfait Σ si et seulement si $\mathcal{I}$ satisfait chaque élément de Σ. Si $\mathcal{I}$ satisfait un axiome (resp. un ensemble d'axiomes), alors $\mathcal{I}$ est dite modèle de cet axiome (resp. l'ensemble d'axiomes). Deux axiomes ou deux ensembles d'axiomes sont équivalents si et seulement s'ils ont les mêmes modèles.

Une équivalence dont le membre gauche est un concept atomique est une définition. Les définitions sont utilisées pour affecter des noms symboliques à des descriptions complexes. Par exemple, par l'axiome :

$$\texttt{Mother} \equiv \texttt{Woman} \sqcap \exists \texttt{hasChild.Person}$$

on associe à la description $\texttt{Woman} \sqcap \exists \texttt{hasChild.Person}$ le nom `Mother`. Les noms symboliques peuvent être utilisés comme des abréviations dans d'autres descriptions. Si, par exemple, on définit le concept `Father` par analogie au concept `Mother`, on peut définir le concept `Parent` comme suit :

$$\texttt{Parent} \equiv \texttt{Mother} \sqcup \texttt{Father}.$$

On appelle un ensemble de définitions Σ une terminologie ou une TBox si aucun nom symbolique n'est défini pas plus d'une fois, c'est-à-dire que pour chaque concept atomique A, il existe au plus un axiome dans Σ dont le membre gauche est A.

Exemple 54 *La terminologie suivante exprime des concepts liés à des relations familiales :*

$$\begin{aligned}
\texttt{Woman} &\equiv \texttt{Person} \sqcap \texttt{Female} \\
\texttt{Man} &\equiv \texttt{Person} \sqcap \neg\texttt{Woman} \\
\texttt{Mother} &\equiv \texttt{Woman} \sqcap \exists\texttt{hasChild.Person} \\
\texttt{Father} &\equiv \texttt{Man} \sqcap \exists\texttt{hasChild.Person} \\
\texttt{Parent} &\equiv \texttt{Mother} \sqcup \texttt{Father} \\
\texttt{GrandMother} &\equiv \texttt{Mother} \sqcap \exists\texttt{hasChild.Parent} \\
\texttt{MotherWithoutDaughter} &\equiv \texttt{Mother} \sqcap \forall\texttt{hasChild.}\neg\texttt{Woman}
\end{aligned}$$

Par ailleurs, les concepts d'inclusion sont utilisés pour exprimer l'appartenance d'un concept à un autre. On appelle une inclusion où le membre gauche est un concept atomique une spécialisation telle que $\texttt{Woman} \sqsubseteq \texttt{Person}$.

11.3.5 Les assertions ABox

Le second composant d'une base de connaissances exprimée en logiques de description est la description du monde ou la ABox. La ABox décrit un état spécifique du domaine en termes de concepts et de rôles. Plus précisément, dans une ABox, on introduit des individus (en leur donnant des noms) ainsi que leur propriétés. Les individus sont notés par a, b, c. Soit C un concept et R un rôle. On peut former des assertions comme suit : $C(a)$ et $R(b, c)$.

La première assertion, dite assertion de concept, signifie que a appartient à (l'interprétation de) C. La deuxième, appelée assertion de rôle, signifie que c est un remplisseur du rôle R pour b.

Par exemple, si `PETER`, `PAUL` et `MARY` sont des noms d'individus, alors `Father (PETER)` signifie que `PETER` est un père et `hasChild(MARY, PAUL)` signifie que `PAUL` est enfant de `MARY`. Une ABox notée $\mathcal{A}$ est un ensemble fini de telles assertions.

Une ABox peut être vue comme étant une instance d'une base de données relationnelle avec seulement des relations unaires et binaires. Toutefois, contrairement à la sémantique du monde clos des bases de données classiques, la sémantique des ABox est une sémantique monde ouvert étant donné que les systèmes de représentation de connaissances sont appliqués dans des situations où l'on peut supposer que l'information est incomplète.

La sémantique des ABox est définie par l'extension des interprétations aux noms d'individus. Donc une interprétation $\mathcal{I} = (\Delta^{\mathcal{I}}, .^{\mathcal{I}})$ n'assigne pas uniquement des ensembles et des relations binaires aux concepts et au rôles mais aussi assigne à chaque nom d'individu a un élément $a^{\mathcal{I}}$ de $\Delta^{\mathcal{I}}$.

D'autre part, une interprétation $\mathcal{I}$ satisfait l'assertion de concept $C(a)$ si $a^{\mathcal{I}} \in C^{\mathcal{I}}$. Elle satisfait l'assertion de rôle $R(a, b)$ si $(a^{\mathcal{I}}, b^{\mathcal{I}}) \in R^{\mathcal{I}}$ et elle satisfait une ABox $\mathcal{A}$ si elle satisfait chaque assertion dans $\mathcal{A}$. Dans ce cas, $\mathcal{I}$ est dite modèle de l'assertion ou de la ABox. Enfin, une interprétation $\mathcal{I}$ satisfait une assertion ou une ABox par rapport à une terminologie Σ si en plus d'être modèle de l'assertion ou de la base, elle est également modèle de la terminologie Σ.

11.3.6 Raisonnement en logiques de description

Intuitivement, au lieu de concevoir de nouveaux algorithmes en DLs, on peut essayer de réduire le problème qu'on veut résoudre à un problème d'inférence connu en logique classique. Par exemple, la décidabilité du problème d'inférence en logique $\mathcal{ALC}$ peut être obtenu en ayant la décidabilité de la logique $\mathcal{L}^2$ (un fragment de la logique des prédicats du premier ordre à deux variables) en sachant que tout concept $\mathcal{ALC}$ peut être traduit dans la logique $\mathcal{L}^2$. Prenons l'exemple suivant :

Exemple 55 *Une traduction directe du concept $\forall R.(\exists R.A)$ génère la formule*

$$\forall y.(R(x, y) \rightarrow (\exists z.(R(y, z) \wedge A(z)))).$$

Comme la sous formule $\exists z.(R(y, z) \wedge A(z))$ ne contient pas la variable x, cette dernière peut être réutilisée à la place de z. Ce renommage génère la formule équivalente suivante :

$$\forall y.(R(x, y) \rightarrow (\exists x.(R(y, x) \wedge A(x))))$$

qui utilise uniquement deux variables.

Toutefois, la complexité des procédures de décision obtenues ainsi est souvent plus élevée qu'il en faut ce qui montre l'intérêt de développer de nouveaux algorithmes spécifiques aux logiques de description.

Par ailleurs, tous les problèmes d'inférence peuvent être réduits au problème de satisfiabilité en sachant que la logique de description en question permet la négation et la disjonction. Pour les logiques de description qui n'utilisent pas ces deux opérateurs en même temps la subsomption de concepts peut être calculée par des algorithmes de subsomption structurelle, c'est-à-dire des algorithmes qui comparent la structure syntaxique des descriptions de concepts.

Bien que ces algorithmes soient très efficaces, ils ne sont complets que pour des logiques simples peu expressives. En particulier, les logiques de description avec négation et disjonction ne peuvent être pas traitées par ce type d'algorithmes. En revanche, pour de telles logiques, les algorithmes basés-tableaux se trouvent très utiles. Le premier algorithme basé-tableaux en logique de description a été proposé par [87] dans l'optique de tester la satisfiabilité des concepts $\mathcal{ALC}$. Depuis, cette approche a été utilisée pour tester la satisfiabilité de nombreuses logiques de description qui étendent $\mathcal{ALC}$.

Les algorithmes de tableaux réduisent le problème de subsomption au problème de satisfiabilité. En fait, on sait que $C \sqsubseteq D$ si et seulement si $C \sqcap \neg D$ est insatisfiable. Le lecteur pourra être référé utilement à [3] pour avoir plus de détails quant aux algorithmes de tableaux. Enfin, citons quelques moteurs d'inférences basés sur les DLs tels que FaCT [60], Racer [57], Pellet [92], FaCT++ [95], F-OWL [105].

11.4 Représentation en DLs des connaissances en détection d'intrusions

En détection d'intrusions, les informations que l'on manipule sont de nature très structurée. Souvent, ces informations sont représentées en XML. Par exemple, le format d'alertes IDMEF (pour Intrusion Detection Message Exchange Format) décrit les alertes en XML. Il est de même pour les descriptions des vulnérabilités données par OVAL (Open Vulnerability and Assessment Language). Néanmoins, XML est limité à une représentation syntaxique. Étant donné que cette représentation est dénuée de sémantique, un formalise logique s'impose. Dans ce cas, la logique propositionnelle n'est pas vraiment adéquate, car elle ne permet pas de représenter les informations de manière structurée. D'où la nécessité d'aller au delà de la cette logique en termes d'expressivité.

Nous proposons alors de considérer un fragment de la logique du premier ordre, à savoir les logiques de description. Le choix de telles logiques est justifié tout d'abord par le fait qu'elles conviennent à la représentation des informations structurées. Aussi, elles sont décidables. De plus, on dispose actuellement d'un nombre considérable de logiques de description variant en termes d'expressivité, et pour lesquelles la complexité du raisonnement est bien connue. Par ailleurs, de nombreux raisonneurs en DLs ont été développés comme Fact ++. La plupart de ces derniers utilisent des techniques d'optimisation sophistiquées.

De ce fait, ces raisonneurs sont général efficaces en pratique, notamment par rapport à des problèmes réels.

Les informations nécessaires à notre approche de corrélation impliquent tout d'abord les alertes générées par les IDSs. Pour la description d'alertes, nous nous sommes basés sur la description des alertes selon le format IDMEF, qui est largement utilisé en détection d'intrusions. Aussi, nous avons besoin d'informations contextuelles à savoir la topologie ainsi que la cartographie. Pour ce dernier point, nous nous sommes essentiellement inspirés du modèle M4D4 [76]. Enfin, notre approche de corrélation requiert la description des vulnérabilités. Pour ce faire, nous avons partiellement utilisé le modèle M4D4, tout en considérant d'autres sources de description de vulnérabilités, en particulier OVAL.

11.4.1 Représentation de l'IDMEF en logiques de description

L'IDMEF (pour Instrusion Detection Message Exchange Format) est un format d'alertes proposé pour le groupe IDWG (Intrusion Detection Working Group). Il s'agit d'un modèle qui a été implémenté en XML. Une description détaillée de ce format et sur laquelle nous nous sommes basés est fournie par le RFC 4765[2]. En particulier, une alerte en IDMEF admet les caractéristiques suivantes :

- Identifier : son identifiant,
- CreateTime : l'instant de la création de l'alerte par l'IDS,
- DetectTime : l'instant de la détection de l'attaque,
- AnalyserTime : l'heure courante de l'IDS,
- Analyser : l'IDS ayant généré cette alerte
- Source : désigne l'attaquant,
- Target : correspond à la victime,
- Classification : représente l'attaque,
- Assessement : indique par exemple la gravité de l'attaque,
- AdditionalData : ce champ peut comprendre tout type d'informations en dehors des informations précédentes.

Nous proposons de représenter le vocabulaire de l'IDMEF en logiques de description. Pour ce faire, nous utilisons une TBox construite à partir d'une vingtaine de concepts et d'une soixantaine de rôles. Cette TBox comprend des axiomes de définitions ainsi que des axiomes d'inclusion.

Par exemple, le concept d'une alerte est donné par la figure 11.1. Un tel axiome signifie qu'une alerte admet un seul identifiant qui est de type chaîne de caractères, un seul champ "detecttime" de type "time", un seul champ "createtime" de type "time", et au plus un seul champ "analysertime" de type "time". De plus, à une alerte peut correspondre une source, voire même plusieurs. Il est de même, il peut lui correspondre une cible ou plusieurs. En

[2]http ://www.ietf.org/rfc/rfc4765.txt

outre, une alerte admet une seule classification, au plus un élément "assement" et un champ "additional data".

$$
\begin{aligned}
\texttt{Alert} \sqsubseteq\ & \forall \texttt{messageId.String} \sqcap = 1\ \texttt{messageId} \sqcap \\
& \forall \texttt{hasCreateTime.Time} \sqcap = 1\ \texttt{hasCreateTime} \sqcap \\
& \forall \texttt{hasDetectTime.Time} \sqcap \leq 1\ \texttt{hasDetectTime} \sqcap \\
& \forall \texttt{hasAnalyserTime.Time} \sqcap \leq 1\ \texttt{hasAnalyserTime} \sqcap \\
& \forall \texttt{hasAnalyser.Analyser} \sqcap = 1\ \texttt{hasAnalyser} \sqcap \\
& \forall \texttt{hasSource.Source} \sqcap \\
& \forall \texttt{hasTarget.Target} \sqcap \\
& \forall \texttt{hasClassification.Classification} \sqcap = 1\ \texttt{hasClassification} \sqcap \\
& \forall \texttt{hasAssessement.Assessment} \sqcap \leq 1\ \texttt{hasAssessment} \sqcap \\
& \forall \texttt{hasAdditionalData.AdditionalData}
\end{aligned}
$$

FIG. 11.1 – Concept Alert

De la même manière, nous définissons par exemple le concept d'une source tel que décrit par la figure 11.2.

$$
\begin{aligned}
\texttt{Source} \sqsubseteq\ & \forall \texttt{sourceId.String} \sqcap \leq 1\ \texttt{sourceId} \sqcap \\
& \forall \texttt{spoofed.\{yes,no,unknown\}} \sqcap \leq 1\ \texttt{spoofed} \sqcap \\
& \forall \texttt{interface.String} \sqcap \leq 1\ \texttt{interface} \sqcap \\
& \forall \texttt{hasNode.Node} \sqcap \leq 1\ \texttt{hasNode} \sqcap \\
& \forall \texttt{hasProcess.Process} \sqcap \leq 1\ \texttt{hasProcess} \sqcap \\
& \forall \texttt{hasUser.User} \sqcap \leq 1\ \texttt{hasUser} \sqcap \\
& \forall \texttt{hasService.Service} \sqcap \leq 1\ \texttt{hasService}
\end{aligned}
$$

FIG. 11.2 – Concept Source

Considérons par exemple la figure 11.3 qui décrit une description IDMEF d'une alerte dans le format XML. Cet exemple est tiré du RFC 4765[3] de l'IDMEF. La description de cet exemple en logiques de description génère une ABox qui comprend les assertions suivantes :

- **Différents concepts**
  - Alert(ALR1)
  - Analyser(ANL1)
  - Source(SRC1)
  - Target(TRG1)
  - Classification(CLS1)
  - Node(NOD1)

[3] http ://www.ietf.org/rfc/rfc4765.txt

```
<?xml version="1.0" encoding="UTF-8"?>

 <idmef:IDMEF-Message xmlns:idmef="http://iana.org/idmef"
                      version="1.0">
  <idmef:Alert messageid="abc123456789">
   <idmef:Analyzer analyzerid="hq-dmz-analyzer01">
    <idmef:Node category="dns">
     <idmef:location>Headquarters DMZ Network</idmef:location>
     <idmef:name>analyzer01.example.com</idmef:name>
    </idmef:Node>
   </idmef:Analyzer>
   <idmef:CreateTime ntpstamp="0xbc723b45.0xef449129">
    2000-03-09T10:01:25.93464-05:00
   </idmef:CreateTime>
   <idmef:Source ident="a1b2c3d4">
    <idmef:Node ident="a1b2c3d4-001" category="dns">
     <idmef:name>badguy.example.net</idmef:name>
     <idmef:Address ident="a1b2c3d4-002"
                    category="ipv4-net-mask">
      <idmef:address>192.0.2.50</idmef:address>
      <idmef:netmask>255.255.255.255</idmef:netmask>
     </idmef:Address>
    </idmef:Node>
   </idmef:Source>
   <idmef:Target ident="d1c2b3a4">
    <idmef:Node ident="d1c2b3a4-001" category="dns">
     <idmef:Address category="ipv4-addr-hex">
      <idmef:address>0xde796f70</idmef:address>
     </idmef:Address>
    </idmef:Node>
   </idmef:Target>
   <idmef:Classification text="Teardrop detected">
    <idmef:Reference origin="bugtraqid">
     <idmef:name>124</idmef:name>
     <idmef:url>http://www.securityfocus.com/bid/124</idmef:url>
    </idmef:Reference>
   </idmef:Classification>
  </idmef:Alert>
```

FIG. 11.3 – Exemple d'alerte

  - `Node(NOD2)`
  - `Node(NOD3)`
  - `Reference(RFC1)`
  - `Address(ADR1)`
  - `Address(ADR2)`
- **L'alerte**
  - `messageId(ALR1, "abc123456789")`
  - `hasCreateTime(ALR1, 2000-03-09T10 :01 :25.93464-05 :00)`
  - `hasAnalyzer(ALR1, ANL1)`
  - `hasSource(ALR1, SRC1)`
  - `hasTarget(ALR1, TRG1)`
  - `hasClassification(ALR1, CLS1)`
- **L'analyser**
  - `analyzerId(ANL1, "hq-dmz-analyzer01")`

  - hasNode(ANL1, NOD1)
- **Le noeud N1**
  - category(NOD1,"dns")
  - location(NOD1,"Headquarters DMZ Network")
  - name(NOD1,analyzer01.example.com)
- **La source**
  - hasNode(SRC1,NOD2)
  - hasIdent(SRC1, a1b2c3d4-002)
- **Le noeud N2**
  - hasIdent(NOD2,"a1b2c3d4-001")
  - category(NOD2,"dns")
  - name(NOD1,badguy.example.net)
  - hasAddress(NOD2, ADR1)
- **L'adresse ADR1**
  - hasIdent(ADR1,"a1b2c3d4-002")
  - category(ADR1,"ipv4-net-mask")
  - address(ADR1,192.0.2.50)
  - netmask(ADR1, 255.255.255.255)
- **Le nœud N3**
  - hasIdent(NOD3,"d1c2b3a4")
  - category(NOD3,"dns")
  - hasAddress(NOD3, ADR2)
- **La cible**
  - hasIdent(TRG1,"d1c2b3a4")
  - hasNode(TRG1,NOD3)
- **L'adresse ADR2**
  - category(ADR2,"ipv4-addr-hex")
  - address(ADR2,0xde796f70)
- **La classif**
  - text(CLS1,"Teardrop detected")
  - hasReference(CLS1,RFC1)
- **La reference**
  - origin(RFC1, "bugtraqid")
  - name(RFC1,124)
  - url(RFC, http ://www.securityfocus.com/bid/124)

11.4.2 Topologie

Décrire la topologie permet par exemple de déduire si un IDS est capable ou non de détecter une alerte. La topologie concerne les nœuds ainsi que leurs interconnexions. Dans

le modèle M4D4, on considère que chaque réseau admet une seule adresse. Nous traduisons cette information en DLs de la façon suivante :

$$\texttt{Network} \sqsubseteq \forall \texttt{netaddress.String} \sqcap = 1\ \texttt{netaddress}$$

Les nœuds représentent n'importe quelle machine connectée au réseau. Un nœud admet une adresse et appartient à un réseaux.

$$\begin{aligned} \texttt{Node} \quad \sqsubseteq \quad & \forall \texttt{nodeaddress.String} \sqcap = 1\ \texttt{nodeaddress} \sqcap \\ & \forall\ \texttt{hasNodeNet.Network} \end{aligned}$$

Les passerelles sont des nœuds particuliers dont l'objectif est de connecter des réseaux. Clairement, une passerelle appartient à plus d'un réseau.

$$\begin{aligned} \texttt{Gateway} \quad & \sqsubseteq \quad \texttt{Node} \sqcap > 1\ \texttt{hasNodeNet} \\ \texttt{Node} \sqcap \neg\texttt{Gateway} \quad & \sqsubseteq \quad = 1\ \texttt{hasNodeNet} \end{aligned}$$

Les passerelles directement accessibles par un nœud sont désignées par :

$$\begin{aligned} \texttt{Node} \quad \sqsubseteq \quad & \forall\ \texttt{hasNodeGateway.Gateway} \sqcap \\ & \forall\ \texttt{hasNodeSystemname.String} \sqcap = 1\ \texttt{hasNodeSystemname} \end{aligned}$$

Enfin, un nœud peut être en plus caractérisé par son nom système :

$$\texttt{Node} \quad \sqsubseteq \quad \forall\ \texttt{hasNodeSystemName.String} \sqcap = 1\ \texttt{hasNodeSystemName}$$

11.4.3 Cartographie en DL

Selon le modèle M4D4, un produit est caractérisé par un seul nom, une seule version, un seul type et par une seule architecture. En logiques de description, ceci correspond au concept produit que nous exprimons en DLs de la manière suivante :

$$\begin{aligned} \texttt{Software} \quad \sqsubseteq \quad & \forall \texttt{softwareName.String} \sqcap = 1\ \texttt{softwareName} \sqcap \\ & \forall \texttt{softwareVersion.String} \sqcap = 1\ \texttt{softwareVersion} \sqcap \\ & \forall \texttt{softwareType.String} \sqcap = 1\ \texttt{softwareType} \sqcap \\ & \forall \texttt{softwareArchitecture.String} \sqcap = 1\ \texttt{softwareArchitecture} \sqcap \end{aligned}$$

Un nœud héberge un logiciel :

$$\texttt{Node} \quad \sqsubseteq \quad \forall\ \texttt{hosts.Software}$$

Un processus est un produit exécuté par un utilisateur.

$$\begin{aligned}\texttt{Process} \quad \sqsubseteq \quad & \forall \texttt{hasSoftware.Software} \sqcap = 1\ \texttt{hasProduct} \sqcap \\ & \forall \texttt{hasUser.User} \sqcap = 1\ \texttt{hasUser}\end{aligned}$$

Un service est un processus qui écoute sur un port :

$$\begin{aligned}\texttt{Service} \quad \sqsubseteq \quad & \forall \texttt{hasProcess.Process} \sqcap = 1\ \texttt{hasProcess} \sqcap \\ & \forall \texttt{port.Integer} \sqcap = 1\ \texttt{port}\end{aligned}$$

11.4.4 Les vulnérabilités en DLs

En général, telle que décrite par exemple dans M4D4, une vulnérabilité est caractérisée par son degré de sévérité, le niveau d'accès nécessaire afin de l'exploiter, ses conséquences et sa date de publication. Nous représentons alors le concept vulnérabilité en DL comme suit :

$$\begin{aligned}\texttt{Vulnerability} \quad \sqsubseteq \quad & \forall \texttt{severity}.\{high, medium, low\} \\ & \forall\ \texttt{requires}.\{remote, local, user\} \\ & \forall\ \texttt{losstype}.\{confidentiality, integrity, availibality, privilege\_escalation\} \\ & \forall \texttt{published.Date}\end{aligned}$$

Dans le modèle M4D4, on considère qu'une vulnérabilité affecte tout simplement une liste de produits. Cette liste est appelée configuration. Par ailleurs, les caractéristiques de vulnérabilité peuvent être extraites de plusieurs sources. Par exemple, la base de données NVD[4] (National Vulnerability Database), la base de données OSVDB[5] Open Source Vulnerability Database ainsi que le projet OVAL[6] (Open Vulnerability and Assessment Language) sont des initiatives indépendantes visant à structurer les informations relatives aux vulnérabilités. En analysant de plus près ces sources, en particulier OVAL qui est un standard, nous avons constaté que le fait qu'un nœud soit affecté d'une vulnérabilité revient généralement à vérifier des conditions logiques plus compliquées impliquant des opérateurs de conjonction, de disjonction et de négation. Par exemple, considérons la description (selon OVAL) de la vulnérabilité CVE-2008-0082 donnée par la figure 11.4. Dans cette description, on voit clairement que la condition de la vulnérabilité est donnée sous forme d'une formule logique composée.

[4] http ://nvd.nist.gov/
[5] http ://www.osvdb.org
[6] http ://oval.mitre.org/

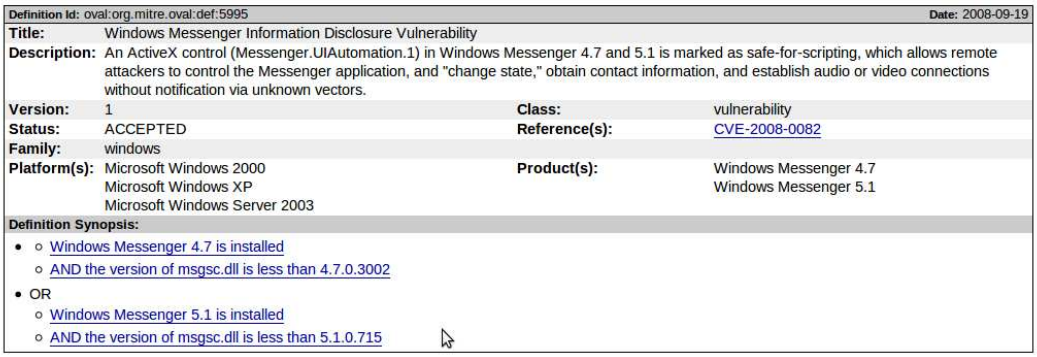

FIG. 11.4 – Vulnérabilité CVE-2008-0082

Par conséquent, la notion de configuration proposée dans M4D4 n'est pas suffisante afin de décrire une vulnérabilité. Nous proposons alors de tirer profit des sources précédentes pour décrire les vulnérabilités.

Ainsi, le fait qu'une machine soit par exemple vulnérable relativement à CVE-2008-0082 est décrit comme suit :

$$\exists vulnerableTo.(\exists hasReference.CVE-2008-0082) \sqsubseteq$$
$$(\exists host.(\exists hasName.WindowsMessenger4.7) \sqcap$$
$$\exists host.((\exists hasName.msgsc.dll) \sqcap (\exists hasVersion. \leq 4.7.0.3002)))\sqcup$$
$$(\exists host.(\exists hasName.WindowsMessenger5.1) \sqcap$$
$$\exists host.((\exists hasName.msgsc.dll) \sqcap (\exists hasVersion. \leq 5.1.0.715)))$$

11.5 Nouvelle approche de corrélation d'alertes basée sur la gestion de l'incohérence

L'approche de corrélation d'alertes que nous proposons se situe dans un cadre d'une détection d'intrusions coopérative. Dans ce cas, plusieurs IDSs et d'autres analyseurs tels que les scanners de vulnérabilités et réseaux sont déployés à travers le réseau, et coopèrent dans le processus de détection. L'idée derrière est d'avoir une vision globale et une meilleure couverture du réseau et aussi plusieurs points de vue qui peuvent être complémentaires.

De plus, nous considérons pour les attaques exploitant des vulnérabilités connues, une base de description de vulnérabilités pour vérifier à quel point une attaque signalée a vraiment eu lieu et ce dans le sens où la présence d'une vulnérabilité renforce la présence d'une attaque alors que son absence vérifie l'inverse.

Par ailleurs, il est bien connu que les dispositifs utilisés en détection d'intrusions ne sont pas totalement fiables. En effet, pour les IDSs, les faux positifs (fausse alertes) et les faux négatifs (attaques non détectée) en témoignent. En ce qui concerne les analyseurs réseaux, la récolte des informations se fait en général hors ligne. De ce fait, à un moment donné, la configuration du réseau peut changer sans pour autant mettre à jour les informations de l'analyseur réseau. De plus, en général, on a affaire à des informations incomplètes : l'analyseur est incapable de déterminer la version de tel ou tel logiciel.

Par conséquent, la coopération dans ce cas peut facilement générer des incohérences. Par exemple, un IDS alarme quant à l'existence d'une attaque envers un système exploitant une certaine vulnérabilité alors que le scanner réseau dit que la cible n'est pas vulnérable. Un autre exemple est qu'un IDS signale une attaque alors qu'un autre IDS, qui se trouve capable de détecter une telle attaque selon sa visibilité topologique et fonctionnelle, ne le fait pas.

Nous proposons alors de résoudre les conflits issus de cette coopération par le biais d'une approche logique de raisonnement en présence d'incohérence. Naturellement, nous adoptons une approche basée sur la restauration de la cohérence pour tous les avantages avancés dans le chapitre 4. En particulier, nous considérons le cadre de bases de croyances partiellement préordonnées. En effet, en détection d'intrusions, certains dispositifs sont comparables entre eux. A titre d'exemple, citons le travail de thèse [50] qui consiste à évaluer, et donc à comparer des IDSs. Néanmoins, d'autres dispositifs ne le sont pas : comparer un IDS et les informations fournies par un scanner réseau, surtout si on tient compte de la dynamique du réseau n'a pas de sens. Par conséquent, on voit clairement la nécessité de considérer des bases de croyances partiellement préordonnées.

La question qui se pose maintenant est la suivante : quelle relation d'inférence parmi celles dédiées au raisonnement en présence d'incohérence à partir de bases de croyances partiellement préordonnées convient-il de choisir ? Ces relations d'inférence diffèrent essentiellement relativement à leur degré de prudence, notre réponse à cette question est de déléguer un tel choix à l'administrateur réseau. En effet, tout dépend de la nature du système sous surveillance et du degré de prudence qu'il souhaite assigner à son système. Autrement dit, il va falloir trouver un compromis entre le taux de réduction d'alertes souhaité et le risque de passer à côté d'une vraie attaque (ce risque n'est pas nul car les informations déduites sont plutôt plausibles). Nous pensons par exemple que la surveillance d'un système dans un centre de recherche en énergie atomique nécessite l'approche la plus sceptique possible, par exemple les inférences possibilistes.

Pour résumer, les entrées de notre approche sont :

– les alertes générées par les différents IDSs,
– la topologique ainsi que la cartographie du réseau,
– les caractéristiques des vulnérabilités.

Le résultat est une base de croyances partiellement préordonnées. Plus précisément, une base de croyances exprimées en logiques de description. Ensuite, en appliquant une inférence à partir de bases de croyances partiellement préordonnées selon le degré de prudence requis par l'administrateur, on détermine la plausibilité qu'une certaine machine soit la cible ou non d'une éventuelle attaque signalée. Dans le cas où l'on détermine que cette attaque n'est pas plausible, l'alerte correspondante est éliminée (ou est affectée un degré faible) ce qui réduit le nombre d'alertes à traiter.

11.6 Cas d'utilisation

Illustrons l'approche de corrélation proposée avec un cas d'utilisation simple. Pour ce faire, nous considérons un système de détection d'intrusions coopérative où sont impliqués deux IDSs de type Snort I_A et I_B en plus d'un scanner de réseaux S. Supposons que l'IDS I_A a le même degré de fiabilité que l'IDS I_B. En outre, ces deux IDSs sont incomparables par rapport au scanner de réseaux S.

Supposons maintenant ce qui suit :

– L'IDS I_A génère une alerte signalant une attaque A ayant pour nom "NETBIOS DCERPC ISystemActivator bind attempt" contre la machine M. La traduction de cette information en DL donne l'assertion suivante :

$$A_1 : hasName(A,'' NETBIOS\ DCERPC\ ISystemActivator\ bind\ attempt'')$$

$$\sqcap\ attackedBy(M, A).$$

– L'IDS I_B n'a pas généré d'alerte relative à l'attaque A par rapport à la machine M après un certain temps t ce qui signifie que cette alerte ne sera jamais émise. En DL, on obtient :

$$A_2 : \neg attackedBy(M, A).$$

– D'après le scnan effectué par l'analyseur réseaux S, la machine M héberge Microsoft Windows NT ainsi que le Patch Q823980. La description correspondante en DL est donnée par l'assertion A_3 :

$$A_3 :\ hasName(S_1,'' MicrosoftWindowsNT'')\ \sqcap\ host(M, S_1)$$

$$\sqcap\ hasName(S_2,'' PatchQ823980'')\ \sqcap\ host(M, S_2).$$

– Par ailleurs, on sait d'une manière certaine, selon la documentation de Snort, plus précisément selon la signature suivante :

alert tcp EXTERNAL_NET any -> HOME_NET 135 ("**msg :NETBIOS DCERPC ISystemActivator bind attempt**"; flow :to_server,established; content :"|05|"; distance :0; within :1; content :"|0b|"; distance :1; within :1; byte_test :1,&,1,0,relative; content :"|A0 01 00 00 00 00 00 C0 00 00 00 00 00 00 46|"; distance :29; within :16; **reference :cve,CAN-2003-0352**; classtype :attempted-admin; sid :2192; rev :1;)

que l'attaque A exploite une vulnérabilité ayant pour référence CVE 2003-0352. Nous exprimons cette information en DL via l'axiome d'inclusion suivant :

$$T_1 : \exists attackedBy.(\exists hasName.NETBIOSDCERPCISystemActivatorbindattempt)$$

$$\sqsubseteq \exists vulnerableTo.(\exists hasReference.CVE2003-0352)$$

– Enfin, selon OVAL, une machine est vulnérable par rapport à CVE 2003-0352 si et seulement si elle vérifie les conditions suivantes :
  – La machine héberge Microsoft Windows NT.
  – La machine n'héberge pas le Patch Q823980.
  – La version de rpcss.dll sur cette machine doit être inférieure à 4.0.1381.7224.

  En DL, ces conditions sont résumées via l'axiome d'équivalence suivant :

$$T_2 : \exists vulnerableTo.(\exists hasReference.CVE2003-0352) \sqsubseteq$$
$$\exists host.(\exists hasName.MicrosoftWindowsNT) \sqcap$$
$$\neg\exists host.(\exists hasName.Patch\ Q823980) \sqcap$$
$$\exists host.((\exists hasName.rpcss.dl) \sqcap (\exists hasVersion.\leq 4.0.1381.7224))$$

En résumé, on obtient une base de croyances DL partiellement préordonnée $((T, A), \preceq)$ telle que :
– $T = \{T_1, T_2\}$,
– $A = \{A_1, A_2, A_3\}$.

Le préordre partiel $\preceq$ sur cette base est décrit par la figure 11.5.

Voyons maintenant lesquelles des croyances $attackedBy(M, A)$ et $\neg attackedBy(M, A)$ est par exemple une conséquence p-lexicographique de (T, A).

Tout d'abord, les sous-bases maximales cohérentes de $attackedBy(M, A)$ contenant les informations certaines T_1 et T_2 sont A et B telles que :
– $A = \{T_1, T_2, A_2, A_3\}$,
– $B = \{T_1, T_2, A_1\}$.

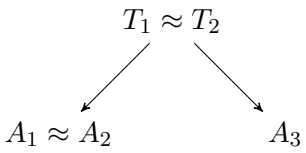

FIG. 11.5 – Le préordre partiel $\preceq$ sur (T, A)

Il est claire que $A \prec_{plex} B$ ce qui implique que l'inférence p-lexicographique à partir (T, A) équivaut l'inférence classique à partir de la sous-base A. De ce fait, on infère $\neg attackedBy(M, A) : (T, A) \vdash_{plex} attackedBy(M, A)$ ce qui est intuitivement attendu. Par conséquent, l'alerte liée à cette attaque peut être supprimée, ce qui réduit le nombre d'alertes qui doivent être analysées et traitées par l'administrateur. Elle peut également se voir assigner un degré de plausibilité faible, ce qui permet de la traiter quand même par prudence, une fois toutes les attaques les plus plausibles sont traitées.

Néanmoins, ce résultat peut être considéré comme étant aventureux pour un administrateur réseau dans un contexte critique. Dans ce dernier cas, il peut opter par exemple pour une approche plus prudente telle que les les extensions possibilistes qui ne permettent de rien déduire dans ce cas particulier, mais qui peuvent bien évidemment inférer des conséquences plus prudentes dans d'autres situations, en modifiant par exemple la relation de fiabilité entre les IDSs.

11.7 Conclusion

Dans ce chapitre, nous avons tout d'abord montré la pertinence du raisonnement en présence d'incohérence, en particulier, à partir de bases de croyances partiellement préordonnées, en détection d'intrusions coopérative. Nous avons alors proposé une nouvelle approche logique de corrélation d'alertes où les informations sont exprimées en logiques de description. Cette approche est paramétrée par une des relations d'inférence à partir de bases de croyances partiellement préordonnées considérées tout au long de ce mémoire. Comme ces relations d'inférence diffèrent essentiellement par rapport à leur prudence, nous déléguons le choix de la relation d'inférence à l'administrateur réseau pour l'effectuer d'une manière subjective et contextuelle. L'étape suivante dans ce cadre consiste à implémenter et à expérimenter cette approche de corrélation d'alertes.

Chapitre 12

Conclusion générale

Conclusion

Dans cette thèse, nous nous sommes intéressés au problème du raisonnement en présence d'incohérence qui est un problème très pertinent en Intelligence Artificielle, et en Informatique d'une manière générale. En particulier, nous nous sommes focalisés sur les approches basées sur la restauration de la cohérence, aussi bien à partir de bases de croyances totalement ordonnées ou stratifiées qu'à partir de bases de croyances partiellement préordonnées.

Dans le cadre des relations d'inférence à partir de bases de croyances partiellement préordonnées, nous avons proposé trois nouvelles approches de compilation afin d'appréhender l'écueil lié à la complexité calculatoire de ces relations d'inférence qui varie entre les classes des problèmes $\Delta_2^p[O(log\ (n))]$-complets et Π_2^p-complets.

La première approche de compilation se rapporte à l'inférence en logique possibiliste : inférence simple, pondérée et conditionnement possibiliste. En plus, elle s'adapte facilement pour traiter l'inférence linéaire. Cette approche s'inspire de celle proposée par Benferhat et Prade [15]. Cependant, la nôtre se trouve paramétrée par n'importe quel langage cible de compilation, entre autres DNNF et les impliqués premiers alors que l'approche de Benferhat et Prade est définie relativement au langage DNF qui est moins intéressant d'un point de vue efficacité spatiale. Aussi, comparée à l'approche de compilation proposée par Marquis et Coste-Marquis [31], notre approche est plus efficace, car par exemple d'un point de vue codage, elle introduit une nouvelle variable par strate, et non pas par formule comme le font Coste-Marquis et Marquis. De plus, à l'inverse de leur approche, la nôtre permet de traiter l'inférence possibiliste pondérée et le conditionnement possibiliste.

La deuxième approche de compilation que nous avons introduite concerne l'inférence lexicographique. Elle se base sur la notion de contraintes de cardinalité Booléennes. Cette approche se trouve complémentaire par rapport aux approches existantes, du fait qu'elle soit paramétrée par n'importe quel langage cible de compilation, y compris celui des impliqués premiers.

En passant au cadre le plus général, à savoir celui des bases de croyances partiellement préordonnées, nous avons proposé une nouvelle extension de l'inférence lexicographique. Plus précisément, nous avons défini deux relations d'inférence basées sur la cardinalité à partir de bases de croyances partiellement préordonnées. Ces deux relations sont assez naturelles et présentent une extension de l'inférence lexicographique classique. La première, consiste à appliquer l'inférence lexicographique classique sur toutes les bases totalement préordonnées compatibles. En ce qui concerne la seconde, qui est moins productive que la première, elle revient à appliquer l'inférence classique à partir de toutes les sous-bases cohérentes qui sont préférées par rapport à une nouvelle relation de préférence lexicographique entre les sous-bases cohérentes d'une base de croyances partiellement préordonnées.

Notre deuxième contribution dans ce même cadre se résume en l'étude comparative des différentes relations d'inférence à partir de bases de croyances partiellement préordonnées relativement à trois facteurs clés : complexité, propriétés logiques et prudence, et ce dans un cadre propositionnel. Cette étude comparative traite des relations d'inférence lexicographiques que nous introduisons, ainsi que les extensions de l'inférence possibiliste à savoir l'inférence possibiliste forte et l'inférence possibiliste faible, et les extensions de l'inférence de l'inclusion telles que l'inférence démocratique et l'inférence relative à la préférence pour l'inclusion basée-compatibles.

Selon cette étude, toutes les relations d'inférence à partir de bases de croyances partiellement préordonnées satisfont les propriétés du système P, tandis qu'aucune d'entre elles ne satisfait la monotonie rationnelle, même celles qui la vérifient par rapport à bases de croyances stratifiées. Notons que nous ne qualifions par le dernier résultat lié à la monotonie rationnelle de négatif car la propriété de monotonie rationnelle reste discutable. Par exemple, elle a été proposée dans l'optique de résoudre le problème de non-pertinence alors que finalement, elle ne le résout que partiellement.

D'un point de vue complexité, notre étude a révélé que ces relations d'inférence sont typiquement au deuxième niveau de la hiérarchie polynomiale PH. Ce passage d'un niveau dans la hiérarchie polynomiale par rapport à l'inférence à partir de bases de croyances stratifiées n'est pas surprenant étant donné la flexibilité offerte par les bases de croyances partiellement préordonnées : c'est le revers de la médaille.

Finalement, ces relations d'inférence diffèrent essentiellement relativement au critère de la prudence versus productivité. En général, ces résultats de prudence généralisent ceux connus dans le cadre totalement préordonné, c'est-à-dire que l'inférence possibiliste est plus prudente que l'inférence relative à la préférence pour l'inclusion qui, à son tour, est plus prudente que l'inférence lexicographique. Notons que la relation entre l'inférence lexicographique partielle et l'inférence de l'inclusion basée-compatibles n'est pas fidèle à ce cas particulier. En effet, ces deux relations d'inférence sont incomparables par rapport à des bases de croyances partiellement préordonnées, alors que la deuxième est plus prudente que la première quant à des bases de croyances totalement préordonnées.

Enfin, nous nous sommes penchés sur un domaine pratique, à savoir celui de la détection d'intrusions coopérative, pour une application du raisonnement en présence d'incohérence, en particulier à partir de bases de croyances partiellement préordonnées. En effet, les dispositifs utilisés en détection d'intrusion ne sont pas totalement fiables, ce qui veut dire que leur coopération peut facilement mener à des situations incohérentes. Par ailleurs, la relation de fiabilité entre ces dispositifs est donnée de manière partielle.

Nous avons alors introduit dans ce cadre une nouvelle approche de corrélation d'alertes, dans l'optique d'améliorer le taux de détection d'alertes. Cette approche se base sur le raisonnement à partir de bases de croyances partiellement préordonnées. Par ailleurs, comme en détection d'intrusion les informations que l'on manipule sont de nature struc-

turée, la logique propositionnelle n'est pas vraiment adéquate pour exprimer de telles informations. Nous avons proposé en revanche d'utiliser les logiques de description [3], qui sont bien adaptées à la représentation de telles connaissances, tout en garantissant la décidabilité du raisonnement, à la différence de la logique du premier ordre dans son intégralité.

Perspectives

Le travail effectué dans cette thèse ouvre de nouvelles pistes à explorer. La première perspective est relative aux bases de croyances stratifiées, et concerne le développement d'une approche de compilation de l'inférence relative à la préférence pour l'inclusion, qui soit paramétrée par n'importe quel langage cible de compilation, comme les approches que nous avons introduites dans le cas des inférences possibiliste, linéaire et lexicographique.

Nous envisageons également d'adapter la compilation de l'inférence lexicographique que nous avons introduite au cas de la fermeture lexicographique qui s'applique à des théories des défauts.

Une autre perspective, consiste à expérimenter davantage ces approches de compilation, notamment celles liées à l'inférence lexicographique et l'inférence MSP, même si théoriquement, elles jouissent de caractéristiques attrayantes, notamment en permettant d'exploiter tous les langages cibles de compilation de connaissances développés dans le cadre de la logique classique, entre autres les plus concis. De plus, nous souhaitons considérer la compilation basée sur les impliqués premiers modulo une théorie proposée par Marquis [74].

Pour le raisonnement à partir de bases de croyances partiellement préordonnées, la première perspective consiste à étendre l'inférence linéaire. Nous aurons ainsi une nouvelle relation d'inférence qui diffèrent des extensions lexicographiques et de l'inclusion en termes de productivité, car déjà dans le cadre stratifié, elle leur est incomparable.

Une autre perspective, d'une importance considérable, part du constat que les relations d'inférence à partir de bases de croyances partiellement préordonnées sont coûteuses d'un point de vue calculatoire, en se situant typiquement au second niveau de la hiérarchie polynomiale. De ce fait, il importe d'appréhender ce problème afin que ces relations soient efficacement exploitables en pratique. Nous ambitionnons alors de proposer de nouvelles approches de compilation pour de telles relations d'inférence. En particulier, nous nous intéressons dans un premier temps à l'adaptation de celles que nous avons introduites dans le cadre stratifié.

En ce qui concerne l'approche de corrélation d'alertes que nous avons proposée, il est naturel de l'implémenter en vue de l'évaluer et de la comparer avec d'autres approches de corrélation. Par ailleurs, nous comptons étendre les résultats de notre étude comparative

au cadre des logiques de description [3], qui conviennent à la représentation d'informations structurées, tout en garantissant le critère de décidabilité. De plus, ces logiques revêtent une grande importance dans le cadre du Web Sémantique. Le Web Sémantique désigne un ensemble de technologies dont le but est de rendre le contenu des ressources du World Wide Web accessible et utilisable de manière automatique, grâce à un système de méta-données formelles. Dans ce contexte, les logiques de description constituent la pierre angulaire du langage OWL [1] (pour Ontology Web Language) : le langage d'ontologies pour le web qui est recommandé par le W3C (World Wide Web Consortium)[2].

Enfin, nous pensons qu'il est intéressant d'aller au delà des bases de croyances partiellement préordonnées, et ce en considérant par exemple des bases de croyances symboliquement pondérées. Dans ce cas, les poids assignés aux croyances, peuvent être donnés sous forme d'expressions composées à partir des opérateurs maximum et minimum [14].

[1] http ://www.w3.org/TR/owl2-profiles/
[2] http ://www.w3.org/

Bibliographie

[1] J. P. Anderson. Computer security threat monitoring and surveillance. Technical Report 98–17, James P Anderson Co., FortWashington, Pennsylvania,USA, April 1980.

[2] K. J. Arrow. *Social choice and individual values*. Yale University Press, 1963.

[3] Franz Baader, Diego Calvanese, Deborah L. McGuinness, Daniele Nardi, and Peter F. Patel-Schneider, editors. *The Description Logic Handbook : Theory, Implementation, and Applications*. Cambridge University Press, 2003.

[4] Olivier Bailleux and Yacine Boufkhad. Efficient cnf encoding of boolean cardinality constraints. In *CP*, pages 108–122, 2003.

[5] S. Benferhat, S. Lagrue, and O. Papini. Reasoning with partially ordered information in a possibilistic framework. *Fuzzy Sets and Systems*, 144 :25–41, 2004.

[6] Salem Benferhat and Rania El Baida. A stratified first order logic approach for access control. *Int. J. Intell. Syst.*, 19(9) :817–836, 2004.

[7] Salem Benferhat, Rania El Baida, and Frédéric Cuppens. A possibilistic logic encoding of access control. In *FLAIRS Conference*, pages 481–485, 2003.

[8] Salem Benferhat, Jean-François Bonnefon, and Rui Da Silva Neves. An experimental analysis of possibilistic default reasoning. In *KR*, pages 130–140, 2004.

[9] Salem Benferhat, Claudette Cayrol, Didier Dubois, Jérôme Lang, and Henri Prade. Inconsistency management and prioritized syntax-based entailment. In *IJCAI*, pages 640–647, 1993.

[10] Salem Benferhat, Didier Dubois, and Henri Prade. Argumentative inference in uncertain and inconsistent knowledge bases. In *UAI*, pages 411–419, 1993.

[11] Salem Benferhat, Didier Dubois, and Henri Prade. How to infer from inconsisent beliefs without revising ? In *IJCAI*, pages 1449–1457, 1995.

[12] Salem Benferhat, Didier Dubois, and Henri Prade. Practical handling of exception-tainted rules and independence information in possibilistic logic. *Appl. Intell.*, 9(2) :101–127, 1998.

[13] Salem Benferhat, Souhila Kaci, Daniel Le Berre, and Mary-Anne Williams. Weakening conflicting information for iterated revision and knowledge integration. *Artif. Intell.*, 153(1-2) :339–371, 2004.

[14] Salem Benferhat and Henri Prade. Encoding formulas with partially constrained weights in a possibilistic-like many-sorted propositional logic. In *IJCAI*, pages 1281–1286, 2005.

[15] Salem Benferhat and Henri Prade. Compiling possibilistic knowledge bases. In *ECAI*, pages 337–341, 2006.

[16] Salem Benferhat and Safa Yahi. Complexity and cautiousness results for reasoning from partially preordered belief bases. In *ECSQARU*, pages 817–828, 2009.

[17] Salem Benferhat, Safa Yahi, and Habiba Drias. On the compilation of stratified belief bases under linear and possibilistic logic policies. In *IJCAI (Poster)*, pages 2425–2430, 2007.

[18] Salem Benferhat, Safa Yahi, and Habiba Drias. On the compilation of possibilistic default theories. In *FLAIRS Conference*, pages 613–618, 2008.

[19] Salem Benferhat, Safa Yahi, and Habiba Drias. A new default theories compilation for msp-entailment. *Journal of Automated Reasoning*, (à paraître), 2009.

[20] R. J. Brachman and J. G. Schmolze. An overview of the kl-one knowledge representation system. *Cognitive Science*, 9(2) :171–216, 1985.

[21] Gerhard Brewka. Preferred subtheories : An extended logical framework for default reasoning. In *IJCAI*, pages 1043–1048, 1989.

[22] Randal E. Bryant. Graph-based algorithms for boolean function manipulation. *IEEE Trans. Computers*, 35(8) :677–691, 1986.

[23] Marco Cadoli and Francesco M. Donini. A survey on knowledge compilation. *AI Commun.*, 10(3-4) :137–150, 1997.

[24] Marco Cadoli, Francesco M. Donini, and Marco Schaerf. Is intractability of nonmonotonic reasoning a real drawback ? *Artif. Intell.*, 88(1-2) :215–251, 1996.

[25] Claudette Cayrol and Marie-Christine Lagasquie-Schiex. On the complexity of nonmonotonic entailment in syntax-based approaches. In *ECAI'94 Workshop on Algorithm, Complexity and Commonsense Reasoning*, 1994.

[26] Claudette Cayrol and Marie-Christine Lagasquie-Schiex. Non-monotonic syntax-based entailment : A classification of consequence relations. In *ECSQARU*, pages 107–114, 1995.

[27] Claudette Cayrol, Marie-Christine Lagasquie-Schiex, and Thomas Schiex. Nonmonotonic reasoning : From complexity to algorithms. *Ann. Math. Artif. Intell.*, 22(3-4) :207–236, 1998.

[28] Claudette Cayrol, Véronique Royer, and Claire Saurel. Management of preferences in assumption-based reasoning. In *IPMU*, pages 13–22, 1992.

[29] Ashok K Chandra and Markowsky G. On the number of prime implicants. *Discrete Mathematics*, pages 7–11, 1978.

[30] Stephen A. Cook. The complexity of theorem-proving procedures. In *Proceedings of the 3rd annual ACM symposium on Theory of computing*, pages 151–158, New York, NY, USA, 1971.

[31] Sylvie Coste-Marquis and Pierre Marquis. On stratified belief base compilation. *Ann. Math. Artif. Intell.*, 42(4) :399–442, 2004.

[32] N. C. A. da Costa. Theory of inconsistent formal systems. *Notre Dame Journal of Formal Logic*, 15 :497–510, 1974.

[33] Adnan Darwiche. Model-based diagnosis using structured system descriptions. *J. Artif. Intell. Res. (JAIR)*, 8 :165–222, 1998.

[34] Adnan Darwiche. Decomposable negation normal form. *J. ACM*, 48(4) :608–647, 2001.

[35] Adnan Darwiche. On the tractable counting of theory models and its application to truth maintenance and belief revision. *Journal of Applied Non-Classical Logics*, 11(1-2) :11–34, 2001.

[36] Adnan Darwiche and Mark Hopkins. Using recursive decomposition to construct elimination orders, jointrees, and dtrees. In *ECSQARU*, pages 180–191, 2001.

[37] Adnan Darwiche and Pierre Marquis. A knowledge compilation map. *J. Artif. Intell. Res. (JAIR)*, 17 :229–264, 2002.

[38] Adnan Darwiche and Pierre Marquis. Compiling propositional weighted bases. *Artif. Intell.*, 157(1-2) :81–113, 2004.

[39] Martin Davis, George Logemann, and Donald Loveland. A machine program for theorem-proving. *Commun. ACM*, 5(7) :394–397, 1962.

[40] Johan de Kleer. An improved incremental algorithm for generating prime implicates. In *AAAI*, pages 780–785, 1992.

[41] H. Debar. *Application des réseaux de neurones à la détection d'intrusions sur les systèmes informatiques*. PhD thesis, Université de Paris 6, 1993.

[42] H Debar, M. Dacier, and Andreas Wespi. A revised taxonomy for intrusion-detection systems, 2000.

[43] Alvaro del Val. Tractable databases : How to make propositional unit resolution complete through compilation. In *KR*, pages 551–561, 1994.

[44] Dorothy E. Denning. An intrusion-detection model. *IEEE Trans. Softw. Eng.*, 13(2) :222–232, 1987.

[45] D. Dubois, J. Lang, and H. Prade. Possibilistic logic. *Handbook of Logic in Articial Intelligence and Logic Programming*, 3 :439–513, 1994.

[46] D. Dubois and Henri Prade. *Possibility Theory : An Approach to Computerized Processing of Uncertainty*. Plenum Press, New York, 1988.

[47] Didier Dubois, Jérôme Lang, and Henri Prade. Inconsistency in possibilistic knowledge bases : to live with it or not live with it. pages 335–351, 1992.

[48] Hélène Fargier and Pierre Marquis. Extending the knowledge compilation map : Krom, horn, affine and beyond. In *AAAI*, pages 442–447, 2008.

[49] D. Gabbay. Theoretical foundations for non-monotonic reasoning in expert systems. pages 439–457, 1985.

[50] Mohammed El-Sayed Gadelrab. *Évaluation des système de détection d'intrusion*. Thèse de doctorat, Université Paul Sabatier, Toulouse, France, décembre 2008.

[51] Peter Gärdenfors. Nonmonotonic inferences based on expectations : A preliminary report. In *Proceedings of the 2nd Principles of Knowledge Representation and Reasoning*, pages 585–590, 1991.

[52] Peter Gärdenfors and David Makinson. Nonmonotonic inference based on expectations. *Artificial Intelligence*, 65(2) :197–245, 1994.

[53] Michael R. Garey and David S. Johnson. *Computers and Intractability ; A Guide to the Theory of NP-Completeness*. W. H. Freeman and Company, New York, NY, USA, 1979.

[54] Hector Geffner. Conditional entailment : closing the gap between defaults and conditionals. In *Third International Workshop on Nonmonotonic Reasoning*, 199O.

[55] Jordan Gergov and Christoph Meinel. Efficient boolean manipulation with obdd's can be extended to fbdd's. *IEEE Trans. Computers*, 43(10) :1197–1209, 1994.

[56] Goran Gogic, Henry A. Kautz, Christos H. Papadimitriou, and Bart Selman. The comparative linguistics of knowledge representation. In *IJCAI (1)*, pages 862–869, 1995.

[57] Volker Haarslev and Ralf Möller. Description of the racer system and its applications. In *Description Logics*, 2001.

[58] Teresa F. Lunt Ann Tamaru Mabry Tyson Harold S. Javitz, Alfonso Valdez and John Lowrance. Next generation intrusion detection expert system (nides) - 1. statistical algorithms rationale - 2. rationale for proposed resolver. technical report a016–rationales. In *SRI International*, 333 Ravenswood Avenue, Menlo Park, CA, 1993.

[59] Steven A. Hofmeyr, Stephanie Forrest, and Anil Somayaji. Intrusion detection using sequences of system calls. *J. Comput. Secur.*, 6(3) :151–180, 1998.

[60] Ian Horrocks. The fact system. In *TABLEAUX*, pages 307–312, 1998.

[61] Jinbo Huang and Adnan Darwiche. Dpll with a trace : From sat to knowledge compilation. In *IJCAI*, pages 156–162, 2005.

[62] U. Junker and G. Brewka. Handling partially ordered defaults in TMS. In *ecsqaru'91*, pages 211–218, 1991.

[63] Hirofumi Katsuno and Alberto O. Mendelzon. Propositional knowledge base revision and minimal change. *Artif. Intell.*, 52(3) :263–294, 1992.

[64] Sarit Kraus, Daniel J. Lehmann, and Menachem Magidor. Nonmonotonic reasoning, preferential models and cumulative logics. *Artificial Intelligence*, 44(1-2) :167–207, 1990.

[65] Marie-Christine Lagasquie-Schiex. *Contribution à l'étude des relations d'inférence non-monotone combinant inférence classique et préférences.* Thèse de doctorat, Université Paul Sabatier, Toulouse, France, décembre 1995.

[66] Jérôme Lang, Paolo Liberatore, and Pierre Marquis. Propositional independence : Formula-variable independence and forgetting. *J. Artif. Intell. Res. (JAIR)*, 18 :391–443, 2003.

[67] Daniel J. Lehmann. Another perspective on default reasoning. *Ann. Math. Artif. Intell.*, 15(1) :61–82, 1995.

[68] Daniel J. Lehmann and Menachem Magidor. Preferential logics : the predicate calculus case. In *TARK*, pages 57–72, 1990.

[69] Daniel J. Lehmann and Menachem Magidor. What does a conditional knowledge base entail ? *Artificial Intelligence*, 55(1) :1–60, 1992.

[70] H. J. Levesque and R. J. Brachman. Expressiveness and tractability in knowledge representation and reasoning. *Computational Intelligence*, 3 :78–93, 1987.

[71] Fangzhen Lin and Ray Reiter. Forget it ! In *In Proceedings of the AAAI Fall Symposium on Relevance*, pages 154–159, 1994.

[72] Teresa F. Lunt and R. Jagannathan. A prototype real-time intrusion-detection expert system. *Security and Privacy, IEEE Symposium on*, 0 :59, 1988.

[73] David Makinson. General theory of cumulative inference. In *Proceedings of the 2nd international workshop on Non-monotonic reasoning*, pages 1–18, New York, NY, USA, 1988. Springer-Verlag New York, Inc.

[74] Pierre Marquis. Knowledge compilation using theory prime implicates. In *IJCAI (1)*, pages 837–845, 1995.

[75] L. Mé. *Audit de sécurité par algorithmes génétiques.* PhD thesis, Université de Rennes 1, 1994.

[76] Benjamin Morin, Ludovic Mé, Hervé Debar, and Mireille Ducassé. A logic-based model to support alert correlation in intrusion detection. *Information Fusion*, 10(4) :285–299, 2009.

[77] Bernhard Nebel. Belief revision and default reasoning : Syntax-based approaches. In *KR*, pages 417–428, 1991.

[78] Bernhard Nebel. Base revision operations and schemes : Semantics, representation and complexity. In *ECAI*, pages 341–345, 1994.

[79] Bernhard Nebel. How hard is it to revise a belief base ? 1998.

[80] Christos H. Papadimitriou. *Computational Complexity*. Addison- Wesley, 1994.

[81] Judea Pearl. System z : A natural ordering of defaults with tractable applications to nonmonotonic reasoning. In *TARK*, pages 121–135, 1990.

[82] Gadi Pinkas and Ronald Prescott Loui. Reasoning from inconsistency : A taxonomy of principles for resolving conflict. In *KR*, pages 709–719, 1992.

[83] David Poole. A logical framework for default reasoning. *Artif. Intell.*, 36(1) :27–47, 1988.

[84] Teodor C. Przymusinski. On the declarative semantics of deductive databases and logic programs. In *Foundations of Deductive Databases and Logic Programming.* Morgan Kaufmann, 1988.

[85] Nicholas Rescher. *Hypothetical Reasoning*. North-Holland, Amsterdam, 1964.

[86] N. Resher and R. Manor. On inference from inconsistent premises. *Theory and Decision*, 1 :179–219, 1970.

[87] M. Schmidt-Schauß and G. Smolka. Attributive concept descriptions with complements. *Artif. Intell.*, 48(1) :1–26, 1991.

[88] Robert Schrag. Compilation for critically constrained knowledge bases. In *AAAI/IAAI, Vol. 1*, pages 510–515, 1996.

[89] Bart Selman and Henry A. Kautz. Knowledge compilation using horn approximations. In *AAAI*, pages 904–909, 1991.

[90] Yoav Shoham. A semantical approach to nonmonotic logics. In *Proceedings of Logic in Computer Science*, pages 275–279, 1987.

[91] Carsten Sinz. Towards an optimal cnf encoding of boolean cardinality constraints. In *CP'05*, pages 827–831, 2005.

[92] Evren Sirin and Bijan Parsia. Pellet : An owl dl reasoner. In *Description Logics*, 2004.

[93] J. R. Slagle, Chin-Liang Chang, and R. C. T. Lee. A new algorithm for generating prime implicants. *IEEE Trans. Comput.*, 19(4) :304–310, 1970.

[94] Pierre Tison. Generalization of consensus theory and application to the minimization of boolean functions. *IEEE Transactions on Electronic Computers EC-16*, pages 446–456, 1967.

[95] D. Tsarkov and I. Horrocks. Efficient reasoning with range and domain constraints. In *Description Logics*, 2004.

[96] Alan M. Turing. On computable numbers, with an application to the entscheidungsproblem. *Proceedings of the London Mathematical Society*, 2(42) :230–265, 1936.

[97] Joost P. Warners. A linear-time transformation of linear inequalities into conjunctive normal form. *Inf. Process. Lett.*, 68(2) :63–69, 1998.

[98] R.R. Yager. On the specificity of a possibility distribution. *Fuzzy Sets and Systems*, 50 :279–292, 1992.

[99] Safa Yahi and Salem Benferhat. Compiling the lexicographic inference using boolean cardinality constraints. In *Canadian Conference on AI*, pages 171–182, 2009.

[100] Safa Yahi, Salem Benferhat, Sylvain Lagrue, Mariette Sérayet, and Odile Papini. A lexicographic inference for partially preordered belief bases. In *KR*, pages 507–517, 2008.

[101] Safa Yahi, Mariette Serayet, Sylvain Lagrue, and Odile Papini. Une inférence lexicographique à partir de bases de croyances partiellement préordonnées. *Information Interaction Intelligence (revue I3)*, à paraître, 2009.

[102] Lotfi A. Zadeh. Fuzzy sets. *Information and Control*, 8(3) :338–353, 1965.

[103] Lotfi A. Zadeh. Fuzzy sets as a basis for a theory of possibility. *Fuzzy Sets and Systems,*, 1 :3–28, 1978.

[104] Jacob Zimmermann, Ludovic Mé, and Christophe Bidan. An improved reference flow control model for policy-based intrusion detection. In *ESORICS*, pages 291–308, 2003.

[105] Y. Zou, T. W. Finin, and H. Chen. F-owl : An inference engine for semantic web. In *FAABS*, pages 238–248, 2004.