UNIVERSITÉ D'ARTOIS

UNIVERSITÉ D'ARTOIS

ECOLE DOCTORALE SPI 072
SCIENCES POUR L'INGÉNIEUR

T H È S E

pour l'obtention du titre de

Docteur en Sciences

de l'Université d'Artois
Mention : INFORMATIQUE

Présentée par

Sébastien RAMON

Méthodes Permettant la Prédominance de Connaissances Subsumées

soutenue publiquement le
2 décembre 2011

Composition du jury :

Rapporteurs :	Odile PAPINI (Université d'Aix-Marseille)	
	Marie-Christine ROUSSET (Université de Grenoble & IUF)	
Examinateurs :	Philippe BESNARD (IRIT CNRS, Toulouse)	*co-directeur de thèse*
	Éric GRÉGOIRE (Université d'Artois)	*co-directeur de thèse*
	Sylvain LAGRUE (Université d'Artois)	
	Pierre MARQUIS (Université d'Artois)	*président du jury*
Invité :	Daniel LE BERRE (Université d'Artois)	

CENTRE DE RECHERCHE EN INFORMATIQUE DE LENS (CRIL CNRS UMR 8188)
Université d'Artois, rue Jean Souvraz, S.P. 18 F-62307, Lens Cedex France
Secrétariat : Tél. : +33 (0)3 21 79 17 23 – Fax : +33 (0)3 21 79 17 70
http://www.cril.univ-artois.fr

Méthodes Permettant la Prédominance de Connaissances Subsumées

—

Sébastien Ramon

Margarita Philosophica (Typus Logice) – Gregor Reisch (1517)
Gravure représentant Dame Logique armée de son épée Syllogismus, chassant le lièvre Problema en compagnie de ses deux chiens Veritas et Falsitas aux abords de la forêt Insolubilia.

Remerciements

THÈSE n.f. (gr. *thesis*, action de poser). Ensemble de travaux présentés sous forme d'ouvrage, en vue de l'obtention du grade de docteur ; exposé public de cet ouvrage.

Au-delà de cette définition sur laquelle la plupart des dictionnaires de la langue française s'accordent, la thèse est tout d'abord une expérience humaine. Je tiens donc à remercier ici celles et ceux qui ont contribué, de près ou de loin, à la réalisation de cette thèse.

Mes premiers remerciements vont à l'ensemble des membres du jury pour l'intérêt qu'ils ont porté à mes travaux. Je tiens ainsi à remercier très chaleureusement Odile Papini et Marie-Christine Rousset de m'avoir fait l'honneur de rapporter ce manuscrit. Leurs remarques pertinentes ont inéluctablement permis de le rendre meilleur. Je tiens aussi à remercier Sylvain Lagrue, Pierre Marquis et Daniel Le Berre d'avoir accepté d'examiner mes travaux, portant sur eux un regard extérieur expert. Je remercie tout particulièrement Pierre Marquis d'avoir accepter de présider le jury de cette thèse.

Bien que quelques lignes ne puissent suffire à exprimer ma gratitude envers ceux qui ont dirigé cette thèse, je tiens à remercier Éric Grégoire et Philippe Besnard sans qui celle-ci n'aurait pu voir le jour. Je tiens à les remercier pour la confiance qu'ils m'ont octroyée et pour la patience dont ils ont fait preuve durant ces trois dernières années. Au-delà de m'avoir transmis un précieux savoir, ils m'ont inculqué leur sens inné de la rigueur et de la persévérance.

Je tiens à remercier aussi l'ensemble de mes collègues pour leur bonne humeur et leur sympathie. Nombre d'entre-eux m'ont donné envie d'exercer ce métier d'enseignant-chercheur lorsque, il y a quelques années déjà, ils m'enseignèrent cette "science des ordinateurs" que je m'efforce d'enseigner aujourd'hui. Je tiens à remercier tout particulièrement ceux avec qui j'ai partagé, plus qu'un bureau, de nombreux moments d'entraide et de convivialité : Badran, Jerry, Karim, Karima, Nicolas, Said, Safa, Tayeb... Je retiendrais les nombreux proverbes camerounais qui y ont été prononcés.

Aussi, je tiens à remercier ici ma famille et mes proches dont la présence et le soutien fût sans faille. Je remercie ainsi mes parents et grand-parents, ma sœur et mes frères qui malgré la complexité de mon travail se sont toujours intéressé à celui-ci. Un grand merci aussi à mon ami Nicolas qui m'a fait le plaisir de quitter le parvis de la Défense pour assister à ma soutenance.

Enfin, mes derniers remerciements vont à celle qui a su donner une saveur exquise à cette fin de thèse. Le regard qu'elle porte sur moi vaut bien plus que ces deux lignes là.

Table des matières

Liste des Figures

Liste des Tables

Liste des Algorithmes

Introduction générale

Dans le domaine de la représentation des connaissances et des raisonnements, la logique a toujours joué un rôle prépondérant. Dans le monde occidental, ses bases ont été formalisées dans l'antiquité par Aristote qui voit en la logique un moyen de construire un discours philosophique cohérent. Il définit pour cela des concepts comme le *syllogisme*, lequel permet de formaliser des raisonnements du type "Tous les hommes sont mortels, or Socrate est un homme, donc Socrate est mortel". Au Moyen Âge, la logique d'Aristote ira jusqu'à être érigée au rang de discipline reine et fera partie des sept *arts libéraux* qui structurent le savoir enseigné (voir en pages liminaires la gravure attribuée à Martin Schongauer extraite de l'ancêtre de l'encyclopédie : la *Margarita Philosophica* ou *Perle de la Connaissance* de Gregor Reisch). Par la suite, bien que la période classique vît naître de profonds changements quant à la manière d'appréhender la logique, il fallut attendre le XIX^{e} siècle et les travaux de Boole et de De Morgan (voir [Boole 1847] et [Boole 1854]) pour que la *logique moderne* (aussi appelée *logique classique*), à la fois symbolique et mathématique, fasse son apparition.

Contexte de la thèse

Le fragment le plus simple et le plus utilisé de la logique classique est certainement la *logique propositionnelle* (voir [Kleene 1967]). Intuitivement, elle permet d'un côté de représenter des connaissances du type "L'interrupteur est enclenché" ou encore "Si l'interrupteur est enclenché alors la pièce est éclairée", appelées *formules propositionnelles*. D'un autre côté, elle permet aussi de raisonner à partir de ces formules et donc par exemple de pouvoir déduire la formule "La pièce est éclairée" à partir des deux formules précédentes. Cependant, la logique propositionnelle ayant vu le jour afin de formaliser des raisonnements mathématiques souffre de la propriété de *monotonie* qui leur est intimement liée, laquelle exprime qu'un raisonnement établi ne peut être rétracté. Elle se limite donc aux raisonnement *absolument corrects*, ne permettant pas de modéliser des *raisonnements révisables*. Ces raisonnements font pourtant partie intégrante des capacités naturelles qu'a l'être humain à tirer des conclusions *simplement plausibles* et à les rétracter à la lueur de nouvelles informations qui les contredisent. Pour reprendre notre exemple, si nous apprenons à la suite de notre raisonnement initial que "L'ampoule est cassée", nous devons alors réviser notre jugement et rétracter notre raisonnement initial à propos de la conclusion "La pièce est éclairée". La logique classique en est totalement incapable et s'effondre alors en présence d'informations contradictoires. Ce comportement est aussi connu sous le nom latin de *ex falso quodlibet sequitur* qui exprime que du faux tout est dérivable.

Ainsi, afin de pallier cette lacune de la logique classique, deux domaines majeurs de l'Intelligence Artificielle (I.A.) virent le jour il y a quelques décennies. D'un côté, les *logiques non monotones* étendent la logique classique afin de permettre un raisonnement révisable (voir [Manor & Rescher 1970], [Gottlob 1992] et [Cayrol *et al.* 1998]). D'un autre côté, la *théorie de la révision de croyances* se propose d'étudier comment doit se comporter un en-

semble de croyances face à une nouvelle croyance pour un agent idéalement *rationnel* (voir [Alchourrón *et al.* 1985], [Konieczny & Pino Pérez 1998] et [Benferhat *et al.* 2000]).

Parmi les logiques non monotones, la *logique des défauts* (voir [Reiter 1980]), certainement la plus connue et la plus étudiée d'entre elles, permet d'exprimer des connaissances sous la forme de règles du type "S'il est cohérent de penser que l'ampoule n'est pas cassée et si l'interrupteur est enclenché, alors la pièce est éclairée", lesquelles permettent de raisonner *par défaut*. En effet, dans notre exemple il n'est pas nécessaire de prouver que "L'ampoule n'est pas cassée" pour conclure que "La pièce est éclairée", conclusion qui peut par la suite être rétractée s'il s'avère que "L'ampoule est cassée".

Problématique de la thèse

Cette thèse s'inscrit dans le domaine de l'*Intelligence Artificielle symbolique*. Elle y traite d'une question fondamentale liée à la représentation des connaissances et des raisonnements en logique. Plus précisément, elle s'intéresse au problème pouvant se produire lors de l'insertion dans un ensemble de connaissances d'une information qui peut déjà en être déduite. Comment faire en sorte que cette nouvelle information vienne *préempter* les informations qui permettent son inférence ? Illustrons l'intérêt de ce problème en reprenant notre exemple canonique. Supposons qu'un ensemble de prémisses contienne l'information "Si l'interrupteur est enclenché alors la pièce est éclairée". Il est naturel d'espérer que l'ajout d'une règle additionnelle, en un sens plus précise que la première, qui exprime que "Si l'interrupteur est enclenché et si l'ampoule n'est pas cassée alors la pièce est éclairée", puisse venir la préempter. En effet, il ne doit plus suffire de savoir que "L'interrupteur soit enclenché" pour en conclure que "La pièce est éclairée" : il faut aussi que "L'ampoule ne soit pas cassée".

La logique classique ne permet pas cette dynamique de raisonnement par le simple ajout de la seconde règle. Par la propriété de monotonie de cette logique, toute conclusion qui peut être tirée d'un ensemble de prémisses continuera à pouvoir l'être quelles que soient les nouvelles informations ajoutées. Ainsi la conclusion qui énonce que "La pièce est éclairée" et qui pouvait être tirée de la présence de la première règle et de l'information établissant que "L'interrupteur est enclenché", restera dérivable même en présence de la seconde règle et d'un fait précisant que "L'ampoule est cassée". Une logique qui résout ce problème devra donc revêtir un caractère non monotone puisque des conclusions peuvent devoir être abandonnées en présence d'informations additionnelles. De manière notable, les *logiques non monotones* ne tiennent pas compte de ce problème car elles apparaissent comme des généralisations de la logique classique et conservent donc leurs propriétés lorsque les prémisses demeurent cohérentes. Dans notre exemple, il est clair que la nouvelle règle est logiquement cohérente avec la première, elle en constitue même une conséquence déductive : son insertion dans un ensemble contenant la première règle ne donne pas lieu à un ensemble incohérent de formules. De même, les approches traditionnelles de *révision de croyances* ne semblent pas directement adaptées car elles préconisent, la plupart du temps, l'union ensembliste des informations quand celles-ci ne sont pas logiquement contradictoires.

Note importante : La question posée ici est bien un problème réel, qui ne résulte pas de choix spécifiques de mode de représentation. En effet, représenter la règle exprimant que "Si l'inter-

rupteur est enclenché alors la pièce est éclairée" par une autre exprimant que "Si la pièce est éclairée alors l'interrupteur est enclenché" n'est pas une solution adaptée. Cette approche de représentation peut sembler résoudre le problème soulevé dans cette thèse, mais empêche la simple déduction de l'information exprimant que "La pièce est éclairée" à partir de la prémisse exprimant que "L'interrupteur est enclenché", implication que nous souhaitions pourtant exprimer en insérant la règle "Si l'interrupteur est enclenché alors la pièce est éclairée".

De plus, l'information devant prédominer n'apporte pas forcément plus de précision comme c'est le cas dans notre exemple canonique. Pour s'en convaincre, considérons qu'un ensemble de prémisses contienne une information exprimant que "Des silos nucléaires ont été observés sur la zone A412 lors d'une opération militaire aérienne". L'ajout d'une information exprimant que "Suite à une nouvelle opération militaire aérienne, des silos ont bien été observés sur la zone A412 mais nul ne sait dire si ce sont des silos nucléaires ou des silos à grains" doit naturellement venir inhiber la première information bien que celle-ci soit plus précise que notre nouvelle information. Pourtant la première information sera toujours déductible de notre ensemble de prémisses suite à l'ajout de notre nouvelle information, laquelle ne peut là non plus prédominer par elle-même.

Notons aussi, que le problème soulevé dans cette thèse se produit aussi bien lorsque l'on souhaite "réviser" des croyances (comme c'est le cas dans les exemples précédents), que lorsque l'on souhaite "mettre à jour" des connaissances (voir Chapitre 3, Section 3.3 pour une explication de la différence entre "révision" et "mise à jour"). Par exemple, considérons qu'un ensemble de prémisses contienne une information exprimant que "Suite aux violents affrontements entre la milice et la population en zone A412, l'ONU a décidé que l'armée Britannique devait intervenir". L'ajout d'une information exprimant que "Suite aux violents affrontements entre la milice et la population en zone A412, l'ONU a décidé que l'armée Britannique ou l'armée Française devait intervenir" doit venir mettre à jour notre connaissance, et doit naturellement venir inhiber la première information.

En respect avec cela, dans la suite nous privilégierons généralement l'utilisation du terme "connaissance" à celui de "croyance".

Contributions de la thèse

Dans cette thèse, notre première contribution à pour but de répondre à cette question fondamentale de la représentation des connaissances et des raisonnement, en nous plaçant dans le cadre de la logique propositionnelle. Dans ce cadre, notre problème peut s'écrire comme suit. Soit une formule propositionnelle g représentant une règle que l'on veut voir préempter lors de son insertion dans un ensemble Γ de formules propositionnelles, au sens où toute capacité à déduire une règle f proche mais "plus forte" que g doit disparaître. Ce problème peut sembler à première vue trivial. Pour le résoudre, nous pourrions simplement retirer f et toute autre règle logiquement "plus forte" que g de l'ensemble de connaissance, et, "casser" les cheminements de raisonnement (impliquant potentiellement plusieurs règles) permettant de déduire g. Seulement, cela n'est pas suffisant puisqu'en insérant ensuite g dans l'ensemble de connaissances ainsi transformé, g pourrait être en mesure de "créer" de nouveaux cheminements de raisonnement de ce type ou d'en "restaurer" certains que nous viendrions à peine de retirer. Dans cette contribution, une solution générale à ce problème, plus fine, se basant sur l'util-

isation d'opérateurs du domaine de la *révision de croyances* (voir [Alchourrón *et al.* 1985]), est présentée. Celle-ci est ensuite mise en œuvre algorithmiquement à l'aide de technologies récentes du domaine de la *satisfiabilité de formules propositionnelles SAT* (voir [Saïs 2008]).

Notre deuxième contribution vise quant à elle à étendre le cadre expressif aux logiques non monotones et plus particulièrement aux logiques permettant une représentation de règles révisables avec exceptions qui peuvent être exprimées par des tests de cohérence, comme la *logique des défauts* (voir [Reiter 1980]). Dans cette contribution, une caractérisation du problème dans un cadre logique très général est formulée. Pour cela nous avons défini de nouveaux objets appelés règles PEC (pour *Prémisses-Exceptions-Conclusion*) qui permettent de représenter et de traiter de manière uniforme aussi bien des formules logiques classiques que des règles d'inférence avec exceptions basées sur des hypothèses de cohérence. Pour pouvoir ensuite exprimer une approche résolvant notre problème dans ce cadre très général, nous avons eu à définir et à étudier différents concepts. Certains de ceux-ci constituent d'intéressants résultats et ouvrent de véritables pistes de recherche en soi. Nous avons défini un concept d'arbre de dérivation manipulant des règles PEC, instanciable aussi bien au cadre de la logique classique qu'à ceux des logiques permettant l'expression de règles d'inférence sous des conditions de cohérence. De manière très puissante, cet outil permet l'inférence aussi bien de formules que de règles PEC dans toute leur généralité. Ce concept a été généralisé de manière à considérer des arbres de raisonnement modulo l'assertion d'une formule additionnelle, ou d'une règle PEC supplémentaire et de ses prémisses, permettant ainsi la modélisation d'une forme de raisonnement hypothétique à double niveau. Ce concept généralisé porte le nom de X-dérivation. Alors que le concept d'impliquant est bien cerné en logique standard, nous avons eu à proposer une forme novatrice d'impliquants portant sur les règles PEC et se basant sur le concept de X-dérivation évoqué plus haut. Ce concept permet notamment la comparaison logique dans un cadre uniforme à la fois de formules et de règles avec exceptions.

Plan du manuscrit

Ce manuscrit est organisé en deux parties. La première introduit les *outils de logique* dont il est fait usage dans la deuxième partie, laquelle regroupe les *contributions* de cette thèse dont nous venons de faire la présentation.

Dans la première partie, le chapitre 0 regroupe quelques *préliminaires formels* introduisant les notions indispensables à la compréhension du manuscrit. Le chapitre 1 a pour objectif la présentation succincte de la *logique propositionnelle*. Son langage, une théorie des modèles ainsi qu'une théorie formelle de la démonstration y seront très brièvement introduites. Certains problèmes pratiques utilisant le formalisme de la logique propositionnelle auxquels nous ferons référence dans la suite du manuscrit y seront aussi présentés. Les *logiques non monotones* et plus particulièrement la logique des défauts et la circonscription, seront présentées au chapitre 2. Une brève discussion concernant la notion de raisonnement révisable et les différentes caractéristiques que doivent revêtir les formalismes permettant un tel raisonnement y feront aussi bonne figure. Pour clôturer cette première partie, nous présenterons brièvement les différentes opérations de *changement de croyances* du domaine de la révision de croyances au chapitre 3.

Dans la seconde partie, le chapitre 4 présentera notre solution au problème de l'inser-

tion dans un ensemble de formules propositionnelles d'une formule qui en est déjà déductible. Notre problème y sera défini de manière formelle et il y sera montré que la solution triviale discutée n'est pas adaptée. Nous y présentons de manière progressive notre solution, certaines de ses propriétés les plus intéressantes et une contre-partie sémantique. Notre solution y sera aussi étudiée d'un point de vue algorithmique et soumise à différentes expérimentations. Le chapitre 5 présentera, quant à lui, notre solution à ce problème dans le cadre général des logiques non monotones permettant l'expression de règles avec exceptions. Nous y présenterons auparavant un formalisme général pour la représentation de règles avec exceptions ainsi que différents outils d'inférence permettant de raisonner à partir de ces règles, en ayant la possibilité de considérer des hypothèses additionnelles. Un concept d'impliquant pour ces règles y sera aussi présenté.

Première partie

Outils de logique

Chapitre 0

Préliminaires formels

Sommaire

Dans ce chapitre, nous introduisons les notions indispensables à la compréhension de la suite du manuscrit. La première, laquelle s'inscrit dans la théorie des ensembles, est la notion de *relation d'ordre* qui permet de comparer les éléments d'un ensemble de manière cohérente. La seconde notion que nous aborderons est celle de *théorie de la complexité* qui permet quant à elle de discriminer différents problèmes informatiques selon leur complexité ou difficulté.

Le chapitre est donc organisé comme suit. Dans la section 0.1, nous présentons brièvement la notion de relation d'ordre. Dans la section 0.2, nous introduisons la théorie de la complexité.

0.1 Relation d'ordre

Dans cette section, nous présentons de manière succincte la notion de *relation d'ordre*. Intuitivement, cette relation est une *relation binaire* dans un *ensemble* qui permet de comparer les éléments de l'ensemble entre eux de manière cohérente. Cette notion s'inscrit donc dans le cadre de la *théorie des ensembles* [1] dont nous ne rappelons que les notations élémentaires. Nous invitons le lecteur à consulter les ouvrages faisant référence (voir [van Heijenoort 2002] par exemple), pour de plus amples détails sur cette théorie.

0.1.1 Notations élémentaires

Définition 1 (taille d'un ensemble)**.** *Soit E un ensemble. Quand un ensemble E est fini, on note $|E|$ son nombre d'éléments (ou cardinal). On appelle singleton tout ensemble E t.q. $|E| = 1$. On appelle ensemble vide, noté $E = \emptyset$, tout ensemble E t.q. $|E| = 0$.*

1. La théorie des ensembles est une branche des mathématiques introduite par Cantor à la fin du XIX^{e} siècle siècle (voir [Cantor 1895] et [Cantor & Jourdain 1911]). Elle se donne comme primitives les notions d'ensemble (éventuellement infinis) et d'appartenance.

Définition 2 (ensemble des parties d'un ensemble)**.** *Soit E un ensemble. On note 2^E l'ensemble des parties de E, également appelé ensemble des sous-ensembles de E.*

Définition 3 (produit cartésien)**.** *Soient E et F deux ensembles. Le produit cartésien de E et F, noté $E \times F = \{(x, y)\ t.q.\ (x \in E)\ et\ (y \in F)\}$, est l'ensemble des couples composés d'un élément de E et d'un autre de F. Pour tout nombre entier n positif, on note E^n le produit cartésien $\underbrace{E \times \ldots \times E}_{n}$.*

0.1.2 Relations binaires

Définition 4 (relation binaire)**.** *Soit E un ensemble. Une relation binaire $\mathcal{R}$ sur E est un sous-ensemble de $E \times E$. On note $x\mathcal{R}y$ lorsqu'un couple d'éléments (x, y) appartient à $\mathcal{R}$.*

Définition 5 (propriétés sur les relations)**.** *Soient E un ensemble et $\mathcal{R}$ une relation binaire sur E. $\mathcal{R}$ est :*

- *réflexive ssi $\forall x \in E,\ x\mathcal{R}x$,*
- *irréflexive ssi $\forall x \in E,\ (x, x) \notin \mathcal{R}$,*
- *transitive ssi $\forall x, y, z \in E$, si $x\mathcal{R}y$ et que $y\mathcal{R}z$ alors $x\mathcal{R}z$,*
- *symétrique ssi $\forall x, y \in E$, si $x\mathcal{R}y$ alors $y\mathcal{R}x$,*
- *anti-symétrique ssi $\forall x, y \in E$, si $x\mathcal{R}y$ et $y\mathcal{R}x$ alors $x = y$,*
- *totale ssi $\forall x, y \in E,\ x\mathcal{R}y$ ou $y\mathcal{R}x$.*

Notons qu'une relation binaire qui n'est pas totale est *partielle*. Les principaux types de relations binaires que nous manipulerons dans la suite ainsi que leur axiomatisation minimale sont présentés dans le tableau 1.

relation	réflexive	irréflexive	transitive	symétrique	anti-symétrique
pré-ordre	✓	✗	✓	✗	✗
ordre	✓	✗	✓	✗	✓
ordre strict	✗	✓	✓	✓	✗
équivalence	✓	✗	✓	✓	✗

TABLE 1 – Principaux types de relations binaires et leur axiomatisation minimale.

Définition 6 (ordre strict induit)**.** *Soient E un ensemble et $\preceq$ un pré-ordre sur E. L'ordre strict $\prec$ induit par $\preceq$ est défini pour tout $x, y \in E$ par $x \prec y$ ssi $x \preceq y$ et $y \npreceq x$.*

Définition 7 (éléments minimaux pour un pré-ordre)**.** *Soient E et F deux ensembles t.q. $E \subseteq F$ et $\preceq$ est un pré-ordre sur F. L'ensemble $min(E, \preceq)$ des éléments de E minimaux pour $\preceq$ est défini comme suit :*

$$min(E, \preceq) = \{e \in E\ t.q.\ \nexists e' \in E\ t.q.\ e' \prec e\}.$$

Exemple 1. *Soient $A = \{a, c, d\}$ et $B = \{a, b, c, d, e\}$ deux ensembles dont la représentation graphique est donnée dans le diagramme de Venn de la figure 1. Soit $\preceq = \{\{a, b, c\} \prec \{d, e\}\}$ un pré-ordre total sur B.*

Notons que $A \subset B$. $min(A, \preceq) = \{a, c\}$.

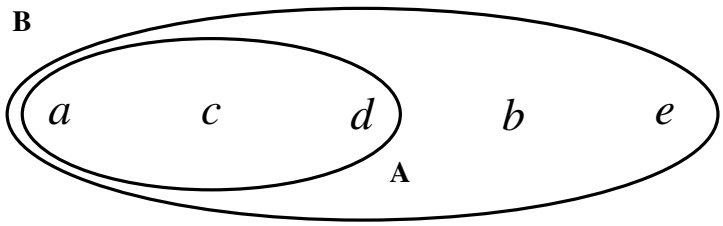

FIGURE 1 – Diagramme de Venn des ensembles de l'exemple 1.

Notons que l'ensemble des parties d'un ensemble, muni de la relation d'inclusion ensembliste et des opérations ensemblistes d'union et d'intersection forment une *algèbre de Boole*.

0.2 Théorie de la complexité

La théorie de la complexité revêt une importance fondamentale en informatique. En effet, elle permet de discriminer différents problèmes informatiques en fonction de leur "difficulté", laquelle se mesure en fonction de la taille de l'instance donnée d'un problème considéré, en terme de temps d'exécution ou d'espace mémoire nécessaire à la résolution du dit problème. De manière générale, on considère le temps d'exécution comme la ressource la plus significative pour la résolution d'un problème, l'espace mémoire lui étant fortement lié. Ainsi, dans cette section, nous présentons succinctement la théorie de la complexité en faisant référence, sauf mention contraire, au temps d'exécution lorsque nous parlerons de la complexité d'un problème. Nous invitons donc le lecteur à consulter les ouvrages de référence pour une présentation complète de la théorie de la complexité (voir [Garey & Johnson 1979] et [Papadimitriou 1994], entre autres).

0.2.1 Notion de problème

En théorie de la complexité, un *problème* est constitué d'un ensemble de données génériques, appelé *entrée*, et, d'une *question* sur cette entrée. Une *instance d'un problème* est une version du problème considéré dont l'entrée a été instanciée à un problème particulier.

Exemple 2. *Soit un problème $\mathcal{P}$ consistant à savoir si un élément x donné appartient à un ensemble E lui aussi donné. Le problème $\mathcal{P}$ est constitué des données génériques x et E et de la question "L'élément x appartient-il à E ?".*

Le problème particulier consistant à savoir si l'élément a appartient à l'ensemble $\{a, b, d\}$ est une instanciation du problème $\mathcal{P}$.

La théorie de la complexité se focalise principalement sur les problèmes de décision, comme le problème de l'exemple 2. Bien que cela puisse sembler très limitatif de prime abord, notons que très souvent, un problème général peut être réduit à un problème de décision qui lui correspond.

Définition 8 (problème de décision)**.** *Un problème de décision est un problème qui ne peut avoir que deux solutions possibles : "oui" ou "non" (ou encore "vrai" ou "faux",* 0 *ou* 1*, . . .).*

Définition 9 (instance positive (resp. négative))**.** *Une instance positive (resp. négative) d'un problème de décision est une instance pour laquelle la solution est "oui" (resp. "faux").*

Définition 10 (problème complémentaire)**.** *Soit $\mathcal{P}$ un problème de décision. Le problème complémentaire à $\mathcal{P}$, noté co$\mathcal{P}$, est le problème dont les instances positives sont celles qui ne le sont pas pour $\mathcal{P}$.*

En théorie de la complexité, ne sont étudiés que les problèmes de décision décidables, dans le sens où il existe un algorithme permettant de les résoudre en un temps fini.

0.2.2 Complexité d'un problème

La complexité d'un problème est donc liée à la complexité des algorithmes permettant de le résoudre en temps fini. La complexité en temps d'un algorithme est quant à elle liée au nombre d'opérations élémentaires nécessaires à son exécution, laquelle dépend fortement de la taille de l'entrée du problème qu'il doit résoudre. Dans la suite, nous considérons la complexité d'un algorithme dans le pire des cas, c'est-à-dire pour une entrée qui demande le plus grand nombre d'opérations élémentaires.

Définition 11 (complexité d'un algorithme)**.** *Considérons un algorithme $\mathcal{A}$. Lorsqu'une instance donnée en entrée de l'algorithme est de taille n, on note $g_{\mathcal{A}}(n)$ le nombre d'opérations élémentaires nécessaires à son exécution. Un algorithme a une complexité en temps $O(f(n))$ ssi :*

$$\exists \alpha \in \mathbb{R}, \exists n_0 \in \mathbb{N} : \forall n \geq n_0, g_{\mathcal{A}}(n) \leq \alpha.f(n).$$

Intuitivement, on dit qu'un algorithme résout un problème est en temps $O(f(n))$, où n est la taille des données du problème, à condition qu'à partir d'un certain rang n_0, le nombre d'opérations élémentaires nécessaires à son déroulement soit majoré par un multiple de $f(n)$. Asymptotiquement, un algorithme en temps $O(n.log(n))$ est plus efficace qu'un algorithme en temps $O(n^3)$, lequel est lui-même plus efficace qu'un algorithme en temps $O(2^n)$.

Exemple 3. *Considérons l'algorithme classique qui effectue un tri interne de données par insertion*[2]*, sur n éléments et prenons comme opérations élémentaires l'affectation et la comparaison. Cet algorithme effectue, dans le pire des cas, $n^2/2$ affectations et $n^2/2$ comparaisons. Par conséquent, cet algorithme a une complexité en temps de $O(n^2)$.*

Définition 12 (algorithme en temps polynomial)**.** *Soit un problème $\mathcal{P}$ dont l'entrée est de taille n. Un algorithme qui résout $\mathcal{P}$ est dit en temps polynomial ssi il est en $O(n^k)$, avec k un entier naturel.*

Définition 13 (algorithme en temps exponentiel)**.** *Soit un problème $\mathcal{P}$ dont l'entrée est de taille n. Un algorithme qui résout $\mathcal{P}$ est dit en temps exponentiel ssi il est en $O(k^n)$, avec $k > 1$ un entier naturel.*

2. De manière conceptuelle, le tri par insertion est le tri que la plupart des personnes utilisent pour trier des cartes : prendre les cartes mélangées une à une sur la table et former une main en insérant chaque carte à sa place.

Définition 14 (traitabilité d'un problème)**.** *Un problème est dit traitable ssi il existe un algorithme en temps polynomial qui le résout. Il est dit intraitable dans le cas contraire.*

Définition 15 (complexité d'un problème)**.** *La complexité d'un problème est la complexité dans le pire des cas du meilleur algorithme qui le résout.*

Exemple 4. *Reprenons l'exemple 3. L'algorithme de tri interne de données par insertion, dont la complexité en temps est en $O(n^2)$, est donc en temps polynomial.*

Pourtant, la complexité du problème de tri interne de données est en temps "quasi-linéaire" $O(n.log(n))$ puisque l'algorithme de tri de données par fusion est en temps $O(n.log(n))$.

Notons que le problème de tri interne de données est donc traitable.

0.2.3 Classification des problèmes

Il est important de noter à ce stade que pour beaucoup de problèmes, aucun algorithme en temps 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 entre "problèmes traitables" et "problèmes intraitables" n'est pas assez fine pour pouvoir les classer de manière pertinente en terme de leur complexité, d'où la nécessité d'associer des classes de complexité plus fines à de tels problèmes.

Machine de Turing

La définition des différentes classes de complexité se base sur un modèle de calcul appelé machine de Turing (voir [Turing 1936]). Une machine de Turing est un modèle abstrait du fonctionnement des appareils mécaniques de calcul, tel un ordinateur et sa mémoire, qui permet de donner une définition précise au concept d'algorithme.

Définition 16 (machine de Turing [Turing 1936])**.** *Une machine de Turing est un quadruplet constitué des éléments suivants :*

1. *Un "ruban" divisé en cases consécutives et de longueur infinie à gauche et à droite. Chacune des cases contient un symbole issu d'un alphabet fini contenant le caractère spécial "blanc",*
2. *Une tête de lecture/écriture qui peut lire et écrire des symboles sur le ruban et qui peut se déplacer vers la gauche et vers la droite d'une case au plus à la fois,*
3. *Un registre d'état mémorisant l'état courant du dispositif, c'est-à-dire l'état du ruban et la position de la tête sur le ruban. L'ensemble des états possibles est fini et contient des états particuliers : l'état initial et un ensemble d'états d'arrêts,*
4. *Une table de transition qui, en fonction de l'état courant du dispositif et du symbole lu par la tête, indique quel symbole écrire sur le ruban, s'il faut déplacer la tête et dans quelle direction et le nouvel état du dispositif.*

Le ruban, la tête de lecture/écriture et le registre constituent le dispositif de calcul commun à tous les appareils mécaniques de calcul (comme un ordinateur et sa mémoire). La table de transitions, quant à elle, représente un algorithme. Le calcul d'une machine de Turing,

en fonction d'une entrée particulière, s'effectue donc de la manière suivante. On place l'entrée sur le ruban, la tête pointant sur la première case contenant la donnée. Puis le calcul se déroule en suivant à chaque étape la table de transitions (et donc l'algorithme) et ne se termine (éventuellement) que lorsque la machine aboutit à un état d'arrêt. Le résultat de l'exécution est alors le mot figurant sur le ruban.

La thèse de Church-Turing affirme que tout procédé de calcul algorithmique peut être modélisé par une machine de Turing [3]. Les machines de Turing sont donc des outils mathématiques simples permettant la simulation de toutes les fonctions calculables par un algorithme, indépendamment de la taille limitée de la mémoire des ordinateurs actuels. Dans la suite, afin de se limiter aux problèmes de décision, nous considérons des machines de Turing ayant exactement deux états d'arrêts : "oui" et "non".

Définition 17 (machine de Turing déterministe/non-déterministe)**.** *Une machine de Turing est dite déterministe si et seulement si sa table de transitions est une application, c'est-à-dire si toute transition ne permet d'atteindre qu'un seul nouvel état du dispositif. À l'inverse, si sa table de transitions possède au moins une transition permettant d'atteindre plusieurs nouveaux états différents du dispositif, alors la machine de Turing est dite non déterministe.*

Les ordinateurs actuels ne peuvent être modélisés que par des machines de Turing déterministes. En effet, quand une machine de Turing est non déterministe, en fonction de l'état du système plusieurs "choix" peuvent être proposés : un ordinateur ne peut pourtant pas "deviner" quelle est la prochaine instruction à exécuter. Néanmoins, les machines de Turing non-déterministes constituent un outils essentiel en théorie de la complexité pour l'établissement des différentes classes de complexité.

Classes de complexité

Définition 18 (résolution d'un problème)**.** *On dit qu'une machine de Turing résout un problème de décision $\mathcal{P}$ si pour toute instance p de $\mathcal{P}$, la machine s'arrête en renvoyant la réponse "oui" (resp. "non") quand p est une instance positive (resp. négative).*

Ici, nous présentons différentes classes de complexité regroupant des problèmes de décision selon la capacité qu'une machine de Turing a pour en permettre la résolution en consommant plus ou moins de ressources (en temps).

Définition 19 (classe P/NP)**.** *La classe P (resp. NP) est l'ensemble des problèmes de décision qui peuvent être résolus par une machine de Turing déterministe (resp. non-déterministe) en temps polynomial par rapport à la taille de l'entrée.*

Définition 20 (classe $coNP$)**.** *La classe $coNP$ est l'ensemble des problèmes de décision dont les problèmes complémentaires appartiennent à la classe NP.*

Définition 21 (classe DP)**.** *Un problème $\mathcal{P}$ appartient à la classe DP s'il peut être écrit comme $\mathcal{P} = \mathcal{P}_1 \cap \mathcal{P}_2$ avec $\mathcal{P}_1 \in NP$ et $\mathcal{P}_2 \in coNP$.*

3. Alonzon Church (voir [Church 1936]), fut le premier à postuler que les fonctions calculables (au sens informel) sont les fonctions récursives (de manière équivalente, celles qui sont caractérisées par une machine de Turing).

Définition 22 (classe $EXPTIME$)**.** *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 de l'entrée.*

Les problèmes appartenant à la classe P sont considérés comme des problèmes relativement faciles, c'est-à-dire ceux pour lesquels on connaît au moins un algorithme efficace. Par le non-déterminisme de la machine de Turing pouvant les résoudre, les problèmes de la classe NP sont quant à eux considérés comme difficiles. En effet, simuler le fonctionnement d'une machine de Turing non-déterministe sur une machine de Turing déterministe a été prouvé de complexité exponentielle.

De manière triviale $P \subseteq NP$. En effet, une machine de Turing déterministe est aussi une machine de Turing non déterministe (il suffit de considérer le déterminisme comme un non déterminisme particulier où il n'y a à chaque étape qu'un seul choix possible). Par contre, la théorie de la complexité ne permet pas pour l'heure de déterminer s'il y a égalité entre les deux classes ou une inclusion stricte entre P et NP[4]. Cependant, même si la théorie ne possède pas encore de réponse définitive à cette question, en pratique, il n'existe encore aucun problème de la classe NP pour lequel on connaît un algorithme polynomial permettant de le résoudre dans le pire des cas. Ainsi, on conjecture que $P \neq NP$.

De la même manière, $P \subseteq coNP$ et on conjecture que $P \neq coNP$. Les classes DP et $EXPTIME$, quant à elles, regroupent les problèmes très difficiles pour lesquels il est nécessaire d'affiner encore la classification.

Réduction polynomiale

Une notion fondamentale en théorie de la complexité et de la calculabilité est celle de réduction. Elle permet de dire qu'un problème $\mathcal{Q}$ est aussi difficile qu'un problème $\mathcal{P}$.

Définition 23 (réduction)**.** *On dit qu'un problème $\mathcal{P}$ est réductible en un problème $\mathcal{Q}$ ssi il existe une fonction f, appelée fonction polynomiale, calculable en temps polynomial, telle que pour toute instance p de $\mathcal{P}$, p est une instance positive de $\mathcal{P}$ ssi $f(p)$ en est une de $\mathcal{Q}$.*

Définition 24 (problème difficile/complet)**.** *Un problème de décision $\mathcal{P}$ est dit X-difficile (ou X-dur) si tout problème de la classe X lui est réductible. $\mathcal{P}$ est dit X-complet s'il est X-difficile et qu'il appartient à 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 (voir [Cook 1971]). Ceci revient au fait que de nombreux problèmes intéressants sont NP-complets et qu'il est très difficile de résoudre un problème NP-complet de manière efficace de manière générale.

4. Ce problème, depuis son établissement en 1970, est certainement l'un des problèmes ouverts les plus fondamentaux de l'informatique théorique. À ce titre, l'Institut de mathématiques Clay, qui se consacre au développement et la diffusion des connaissances mathématiques, a inclus ce problème dans sa liste des problèmes du prix du millénaire (voir http://www.claymath.org/millennium/).

Hiérarchie polynomiale

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

Définition 25 (machine de Turing à oracle)**.** *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.*

À l'aide de ces machines de Turing, on peut définir les classes de complexité suivantes.

Définition 26 (classe P^X/NP^X)**.** *La classe P^X (resp. NP^X) est l'ensemble des problèmes de décision qui peuvent être résolus par une machine de Turing déterministe (resp. non-déterministe) à oracle X en temps polynomial par rapport à la taille de l'entrée.*

Définition 27 (classes Δ_k^P, Σ_k^P et Π_k^P)**.** *On définit récursivement les classes Δ_k^P, Σ_k^P et Π_k^P (k étant un entier positif) par :*

- $\Delta_0^P = \Sigma_0^P = \Pi_0^P = P$,
- $\Delta_{k+1}^P = P^{\Sigma_k^P}$,
- $\Sigma_{k+1}^P = NP^{\Sigma_k^P}$,
- $\Pi_{k+1}^P = co\Sigma_{k+1}^P$.

En particulier, $\Delta_1^P = P$, $\Sigma_1^P = NP$ et $\Pi_1^P = coNP$.

Définition 28 (hiérarchie polynomiale)**.** *La hiérarchie polynomiale PH est l'union des Σ_k^P (k étant un entier positif) :*

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

La définition de PH implique les inclusions suivantes :

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

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_2^P-difficile ou Π_2^P-difficile.

La figure 2 est une représentation graphique des premiers niveaux de la hiérarchie polynomiale PH.

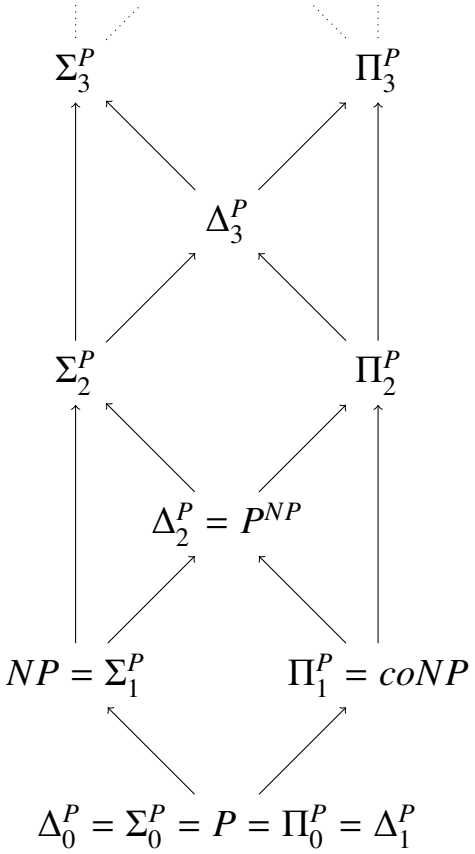

FIGURE 2 – Représentation graphique des premiers niveaux de PH.

Chapitre 1

Logique propositionnelle

Sommaire

C'est le désir d'exprimer de manière formelle certains raisonnements mathématiques qui a conduit à la naissance des premiers outils logiques mathématiques. Aux prémices de l'algébrisation de la logique, les travaux de Boole et de Morgan aboutissent à l'apparition de la logique propositionnelle, fragment le plus simple et le plus utilisé de la logique mathématique classique (voir [Boole 1847] et [Boole 1854]). Cette logique devint le socle de nombreuses autres logiques développées en I.A. par la suite, lesquelles permettent des raisonnements plus élaborés comme le raisonnement révisable ou le raisonnement par défaut, entre autres-choses [1].

Intuitivement, la logique propositionnelle permet d'un côté de représenter des connaissances du type "L'interrupteur est enclenché" ou "Si l'interrupteur est enclenché alors la pièce est éclairée", appelées "formules propositionnelles". D'un autre côté, elle permet aussi de raisonner à partir de ces formules (par exemple déduire la formule "La pièce est éclairée" à partir des deux formules précédentes).

Notons que de nombreux problèmes techniques s'inscrivent dans le cadre de la logique propositionnelle. Les problèmes de satisfiabilité de formules (voir [Saïs 2008]), de compilation de formules (voir [Darwiche & Marquis 2002]) ou encore d'extraction de sources minimales d'incohérence (voir [Piette 2007]), en sont de parfaits exemples.

1. Une présentation succincte de certaines logiques non classiques permettant ce genre de raisonnements est donnée au chapitre 2.

Dans ce chapitre nous présentons une brève introduction à la logique propositionnelle. Des ouvrages de référence permettront au lecteur d'obtenir une présentation plus complète (voir [Kleene 1967] et [Alliot & Schiex 1993], entre autres).

Le chapitre est organisé comme suit. Dans la section 1.1, nous présentons le langage des formules propositionnelles. Dans la section 1.2, une théorie des modèles, qui permettra de donner un sens (ou sémantique) aux formules propositionnelles, sera introduite. Nous étudierons, dans la section 1.3, une théorie formelle de la démonstration sur le langage des formules propositionnelles qui, sans s'intéresser au sens des formules, permettra aussi de les manipuler. Dans cette section, nous prendrons soin de rappeler que la théorie des modèles et la théorie de la démonstration sont équivalentes et qu'il est donc possible d'utiliser indifféremment l'une ou l'autre pour raisonner en logique propositionnelle. Dans la section 1.4, nous présenterons les problèmes pratiques utilisant le formalisme de la logique propositionnelle auxquels nous ferons référence dans la suite du manuscrit.

1.1 Langage

1.1.1 Formules propositionnelles

Les atomes du langage de la logique propositionnelle sont les *variables propositionnelles*. Ces variables ont pour objectif de représenter un énoncé simple (par exemple "L'interrupteur est enclenché") et ne peuvent revêtir que deux valeurs possibles, à savoir *vrai* ou *faux*.

Définition 29 (variable propositionnelle)**.** *Une variable propositionnelle est une variable booléenne représentant une proposition binaire qui ne peut prendre que deux valeurs possibles : vrai ou faux (appelées valeurs de vérité, respectivement notées par* 1 *et* 0*).*

À l'aide d'autres symboles du langage, appelés *connecteurs logiques*, les variables propositionnelles peuvent être "combinées" entre elles.

Définition 30 (connecteurs logiques)**.** *Un connecteur logique est un symbole réservé du langage de la logique propositionnelle qui permet de combiner une ou plusieurs variables propositionnelles. Le nombre (positif) de variables combinées est son "arité". Dans la suite, nous considérerons les connecteurs logiques suivants :*

- *la négation :* $\neg$ *(non), d'arité 1,*
- *la conjonction :* $\wedge$ *(et), d'arité 2,*
- *la disjonction :* $\vee$ *(ou), d'arité 2,*
- *l'implication matérielle :* $\Rightarrow$*, d'arité 2,*
- *et l'équivalence :* $\Leftrightarrow$*, d'arité 2.*

Ainsi, des énoncés plus complexes (par exemple "Si l'interrupteur est enclenché alors la pièce est éclairée"), appelés *formules propositionnelles*, qui forment le *langage propositionnel*, peuvent être construits en "combinant" plusieurs variables propositionnelles entre elles à l'aide des connecteurs logiques, des symboles de parenthèses (et) et des symboles de constantes booléennes $\top$ et $\bot$, représentant respectivement le *vrai* et le *faux*.

Définition 31 (langage des formules propositionnelles)**.** *Soient $\mathcal{V}$ un ensemble fini de variables propositionnelles. Le langage $\mathcal{PROP}_{\mathcal{V}}$ des formules propositionnelles est défini de manière inductive comme le plus petit ensemble tel que :*

- $\forall x \in \mathcal{V}, x \in \mathcal{PROP}_{\mathcal{V}}$,
- $\top, \bot \in \mathcal{PROP}_{\mathcal{V}}$,
- *si* $\alpha \in \mathcal{PROP}_{\mathcal{V}}$, *alors :*
  - $(\alpha) \in \mathcal{PROP}_{\mathcal{V}}$,
  - $(\neg\alpha) \in \mathcal{PROP}_{\mathcal{V}}$,
- *si* $\alpha, \beta \in \mathcal{PROP}_{\mathcal{V}}$, *alors :*
  - $(\alpha \wedge \beta) \in \mathcal{PROP}_{\mathcal{V}}$,
  - $(\alpha \vee \beta) \in \mathcal{PROP}_{\mathcal{V}}$,
  - $(\alpha \Rightarrow \beta) \in \mathcal{PROP}_{\mathcal{V}}$,
  - *et* $(\alpha \Leftrightarrow \beta) \in \mathcal{PROP}_{\mathcal{V}}$.

Notons que souvent, pour alléger l'écriture et quand cela ne crée aucune ambiguïté, les parenthèses sont volontairement omises.

Définition 32 (littéral)**.** *Un littéral l est soit une variable propositionnelle x (on parle de littéral positif) ou sa négation $\neg x$ (on parle alors de littéral négatif). $\neg l$ représente le littéral complémentaire de l.*

Définition 33 (taille d'une formule)**.** *Soit une formule $\alpha \in \mathcal{PROP}_{\mathcal{V}}$. La taille de α, notée $|\alpha|$, est le nombre d'occurrences de variables propositionnelles utilisées pour écrire α.*

Définition 34 (ensemble des variables d'une formule)**.** *Soit une formule $\alpha \in \mathcal{PROP}_{\mathcal{V}}$. On note $Var(\alpha)$ l'ensemble des variables propositionnelles qui apparaissent dans α.*

Exemple 5. *Soit $\mathcal{V} = \{interrupteur\_on, ampoule\_ok, piece\_eclairee\}$ un ensemble de variables propositionnelles représentant respectivement les énoncés de notre exemple canonique "L'interrupteur est enclenché", "L'ampoule n'est pas cassée" et "La pièce est éclairée".*

$\alpha = (interrupteur\_on \wedge ampoule\_ok) \Rightarrow piece\_eclairee$ est une formule propositionnelle de $\mathcal{PROP}_{\mathcal{V}}$ exprimant que "Si l'interrupteur est enclenché et si l'ampoule n'est pas cassée alors la pièce est éclairée".

Notons que $|\alpha| = 3$ et $Var(\alpha) = \mathcal{V}$.

1.1.2 Formes normales

Afin d'optimiser la résolution de certains problèmes techniques s'inscrivant dans le cadre de la logique propositionnelle [2], il est courant de "normaliser" les formules du problème. La "forme normale conjonctive" (ou CNF) et la "forme normale disjonctive" (ou DNF) sont les deux normalisations les plus utilisées.

Définition 35 (monôme)**.** *Un monôme est une conjonction de littéraux.*

Définition 36 (clause)**.** *Une clause est une disjonction de littéraux.*

2. Une présentation succincte de certains d'entre eux est donnée sans la section 1.4.

Une clause (resp. un monôme) peut être représentée par l'ensemble de ces littéraux. Une sous-clause stricte (resp. un sous-monôme) est donc un sous-ensemble strict de l'ensemble de ces littéraux. Notons que la clause vide, symbole du *faux*, est notée $\bot$. On désigne par clause tautologique toute clause qui contient un littéral a et son complémentaire $\neg a$. La clause $\top$, symbole du *vrai* est, par convention, une clause tautologique.

Définition 37 (forme normale disjonctive)**.** *Une formule propositionnelle $\alpha \in \mathcal{PROP}_{\mathcal{V}}$ est sous forme normale disjonctive (abrégée en DNF pour Disjunctive Normal Form) ssi α est une disjonction de monômes.*

Définition 38 (forme normale conjonctive)**.** *Une formule propositionnelle $\alpha \in \mathcal{PROP}_{\mathcal{V}}$ est sous forme normale conjonctive (abrégée en CNF pour Conjunctive Normal Form) ssi α est une conjonction de clauses.*

Notons que toute formule propositionnelle peut être mise sous forme DNF ou sous forme CNF équivalentes pour le problème de la satisfiabilité (dont nous donnerons une description dans la section 1.4). Cependant, la transformation peut nécessiter une croissance exponentielle de la taille en fonction de la taille de la formule initiale.

1.2 Théorie des modèles

Intuitivement, la théorie des modèles est une formalisation de la notion intuitive que nous avons de la vérité ou de la fausseté (ou plus généralement de la correction) d'un énoncé en fonction des propositions qui le composent.

La théorie des modèles a donc pour but d'établir un mécanisme *sémantique* d'évaluation des formules propositionnelles. Pour ce faire, nous allons tout d'abord donner un *sens* aux variables propositionnelles, en leur donnant une valeur de vérité (*vrai* ou *faux*). Ensuite, à partir des de ces valeurs et d'un interprétation sémantique des connecteurs, nous donnerons un sens aux formules elles-mêmes.

1.2.1 Évaluation des formules

Définition 39 (interprétation)**.** *Soit $\mathcal{V}$ un ensemble de variables propositionnelles. Une interprétation ω sur $\mathcal{V}$ est une application de l'ensemble $\mathcal{V}$ des variables propositionnelles vers l'ensemble des valeurs de vérité $\{1, 0\}$:*

$$\omega : \mathcal{V} \to \{1, 0\}.$$

Par abus de notation, dans la suite nous représenterons une interprétation ω sur $\mathcal{V}$ par le sous-ensemble des variables propositionnelles de $\mathcal{V}$ vraies dans ω (les variables de $\mathcal{V}$ n'apparaissant pas dans ω étant donc fausses dans ω).

Notons aussi que l'ensemble des interprétation sur $\mathcal{V}$ est noté $\mathcal{W}_{\mathcal{V}}$.

Définition 40 (sémantique d'un formule)**.** *La sémantique d'une formule propositionnelle dans une interprétation ω est définie inductivement de la manière suivante. Pour toutes formules propositionnelles $\alpha, \beta \in \mathcal{PROP}_{\mathcal{V}}$:*

- $\omega(\top) = 1$,
- $\omega(\bot) = 0$,
- $\omega(\neg\alpha) = 1$ *ssi* $\omega(\alpha) = 0$,
- $\omega(\alpha \wedge \beta) = 1$ *ssi* $\omega(\alpha) = 1$ *et* $\omega(\beta) = 1$,
- $\omega(\alpha \vee \beta) = 0$ *ssi* $\omega(\alpha) = 0$ *et* $\omega(\beta) = 0$,
- $\omega(\alpha \Rightarrow \beta) = 0$ *ssi* $\omega(\alpha) = 1$ *et* $\omega(\beta) = 0$,
- $\omega(\alpha \Leftrightarrow \beta) = 1$ *ssi* $\omega(\alpha) = \omega(\beta)$.

La définition précédente permet donc de donner une interprétation sémantique à une formule propositionnelle. Notons que pour toute formule α, il existe $2^{|Var(\alpha)|}$ interprétations différentes. Cette définition donne aussi une interprétation sémantique aux différents connecteurs utilisés. Notons qu'à partir de cette dernière, il est possible d'associer à chaque connecteur une fonction de valeur dans $\{0, 1\}$, appelée table de vérité (voir Figure 1.1).

$\omega(\alpha)$	$\omega(\beta)$	$\omega(\neg\alpha)$	$\omega(\alpha \wedge \beta)$	$\omega(\alpha \vee \beta)$	$\omega(\alpha \Rightarrow \beta)$	$\omega(\alpha \Leftrightarrow \beta)$
0	0	1	0	0	1	1
0	1	1	0	1	1	0
1	0	0	0	1	0	0
1	1	0	1	1	1	1

TABLE 1.1 – Tables de vérité des différents connecteurs logiques utilisés.

Définition 41 (modèle/contre-modèle)**.** *Soient $\alpha \in \mathcal{PROP}_\mathcal{V}$ une formule et $\omega \in \mathcal{W}_\mathcal{V}$ une interprétation. On dit que ω est un modèle (resp. contre-modèle) de α, noté $\omega \models \alpha$ (resp. $\omega \not\models \alpha$) si $\omega(\alpha) = 1$ (resp. $\omega(\alpha) = 0$).*

On dit aussi que ω satisfait (resp. ne satisfait pas) α ssi $\omega \models \alpha$ (resp. $\omega \not\models \alpha$).

L'ensemble des modèles de α est noté $Mod(\alpha)$.

Définition 42 (formule satisfiable/insatisfiable)**.** *Soit $\alpha \in \mathcal{PROP}_\mathcal{V}$ une formule propositionnelle. α est une formule satisfiable ou "cohérente" (resp. insatisfiable ou "incohérente") ssi il existe (resp. il n'existe pas) $\omega \in \mathcal{W}_\mathcal{V}$ une interprétation telle que $\omega \models \alpha$.*

Définition 43 (formule valide)**.** *Soit $\alpha \in \mathcal{PROP}_\mathcal{V}$ une formule propositionnelle. α est une formule valide, notée $\models \alpha$ ssi $\forall \omega \in \mathcal{W}_\mathcal{V}$, $\omega \models \alpha$.*

Quand α est une formule valide, α (resp. $\neg\alpha$) est une "tautologie" (resp. "contradiction").

Exemple 6. *Reprenons l'exemple 5.*

Soient $\mathcal{V} = \{interrupteur\_on, ampoule\_ok, piece\_eclairee\}$ un ensemble de variables propositionnelles et $\alpha = (interrupteur\_on \wedge ampoule\_ok) \Rightarrow piece\_eclairee$ est une formule propositionnelle de $\mathcal{PROP}_\mathcal{V}$.

Considérons les interprétations $\omega_1, \omega_2 \in \mathcal{W}_\mathcal{V}$ t.q. :

$$\omega_1 = \{interrupteur\_on, piece\_eclairee\},$$
$$\omega_2 = \{interrupteur\_on, ampoule\_ok\}.$$

En accord avec la table de vérité de α (voir Table 1.2), ω_1 est un modèle de α ($\omega_1 \models \alpha$), et, ω_2 un contre-modèle de α ($\omega_2 \not\models \alpha$).

Ainsi, α est une formule satisfiable mais n'est pas une formule valide.

interrupteur_on	*ampoule_ok*	*piece_eclairee*	α	β	γ
0	0	0	1	1	0
0	0	1	1	1	1
0	1	0	1	1	0
0	1	1	1	1	1
1	0	0	1	0	0
1	0	1	1	1	1
1	1	0	0	0	0
1	1	1	1	1	1

TABLE 1.2 – Tables de vérité des formules des exemples 6 et 7.

1.2.2 Conséquence logique

À l'instar de la relation de satisfiabilité qui relie interprétations et formules, il est possible de relier des formules entre elles via une relation de "conséquence logique". À partir de cette relation, nous introduisons différentes notions d'impliquant lesquelles nous serons très utiles par la suite.

Définition 44 (conséquence logique)**.** *Soient $\alpha, \beta \in \mathcal{PROP}_\mathcal{V}$ deux formules propositionnelles. α est une conséquence logique de β, noté $\beta \models \alpha$ ssi $Mod(\beta) \subseteq Mod(\alpha)$.*

Définition 45 (fermeture déductive)**.** *Soit $\alpha \in \mathcal{PROP}_\mathcal{V}$ une formule propositionnelle. La fermeture déductive de α est l'ensemble de formules $Cn(\alpha) = \{\phi \text{ t.q. } \alpha \models \phi\}$ des conséquences logiques de α.*

Définition 46 (base de connaissances propositionnelle)**.** *Soit K un ensemble de formules propositionnelles. La fermeture déductive de K est l'ensemble $Cn(K)$ de formules vraies dans chaque interprétation qui est un modèle de chacune des formules de K.*

Un ensemble de formules propositionnelles clos déductivement est appelé base de connaissances (ou théorie). $K_\bot$ est la base de connaissances triviale et $K_\top$ la base de connaissances tautologique.

Propriété 1. *Soient $\alpha, \beta, \gamma \in \mathcal{PROP}_\mathcal{V}$ trois formules propositionnelles. La table 1.3 dresse quelques propriétés intéressantes de la relation de conséquence logique.*

Définition 47 (impliquant (impliqué))**.** *Soient $\alpha, \beta \in \mathcal{PROP}_\mathcal{V}$ deux formules propositionnelles. α est un impliquant de β et β un impliqué de α ssi β est une conséquence logique de α ($\alpha \models \beta$).*

réflexivité	$\alpha \models \alpha$
transitivité	si $\alpha \models \beta$ et $\beta \models \gamma$, alors $\alpha \models \gamma$
coupure	si $(\alpha \wedge \beta) \models \gamma$ et $\alpha \models \beta$ alors $\alpha \models \gamma$
monotonie	si $\alpha \models \beta$ alors $(\alpha \wedge \gamma) \models \beta$

TABLE 1.3 – Quelques propriétés intéressantes de la relation de conséquence logique.

Définition 48 (impliquant (impliqué) strict)**.** *Soient $\alpha, \beta \in \mathcal{PROP}_{\mathcal{V}}$ deux formules propositionnelles. α est un impliquant (resp. impliqué) strict de β si α est un impliquant (resp. impliqué) de β et que β n'est pas un impliquant (resp. impliqué) de α.*

Définition 49 (impliquant (impliqué) premier)**.** *Soient $\alpha, \beta \in \mathcal{PROP}_{\mathcal{V}}$ deux formules propositionnelles. α est un impliquant (resp. impliqué) premier de β si α est un impliquant (resp. impliqué) strict de β et qu'il n'existe aucune formule propositionnelle $\gamma \in \mathcal{PROP}_{\mathcal{V}}$ t.q. α soit un impliquant (resp. impliqué) strict de γ et que γ soit un impliquant (resp. impliqué) strict de β.*

Exemple 7. *Soient $\alpha, \beta, \gamma \in \mathcal{PROP}_{\mathcal{V}}$ trois formules propositionnelles t.q. :*

$$\alpha = (interrupteur\_on \wedge ampoule\_ok) \Rightarrow piece\_eclairee,$$
$$\beta = interrupteur\_on \Rightarrow piece\_eclairee,$$
$$\gamma = piece\_eclairee.$$

Comme le montrent les tables de vérité de ces formules (voir Table 1.2), $Mod(\gamma) \subset Mod(\beta) \subset Mod(\alpha)$. Ainsi α et β sont des conséquences logiques de γ et α est une conséquence logique de β. On peut en déduire les liens d'implication suivants :

- *γ est un impliquant (strict) de α et de β, et, β est un impliquant (strict) de α.*
- *β est ainsi un impliquant premier de α, γ un impliquant premier de β, mais γ n'est par contre pas un impliquant premier de α.*
- *α et β sont des impliqués (stricts) de γ, et, α est un impliqué (strict) de β.*
- *β est ainsi un impliqué premier de γ, α un impliqué premier de β, mais α n'est par contre pas un impliqué premier de γ.*

Équivalence de formules

À partir de la notion de conséquence logique il est possible de définir une relation d'équivalence entre formules propositionnelles.

Définition 50 (équivalence logique)**.** *Soient $\alpha, \beta \in \mathcal{PROP}_{\mathcal{V}}$ deux formules propositionnelles. α et β sont logiquement équivalentes, noté $\alpha \equiv \beta$, ssi $\alpha \models \beta$ et $\beta \models \alpha$.*

L'équivalence et le principe de remplacement des équivalents permettent la manipulation syntaxique des formules propositionnelles tout en préservant la sémantique. Il existe ainsi de nombreuses règles de réécriture de formules propositionnelles comme le montre la table 1.4 où $\alpha, \beta, \gamma \in \mathcal{PROP}_{\mathcal{V}}$ sont trois formules propositionnelles. Aussi, on peut facilement déduire les équivalences suivantes des tables de vérité de la table 1.1 :

- $\alpha \vee \beta \equiv \neg\alpha \Rightarrow \beta$,
- $\alpha \wedge \beta \equiv \neg(\alpha \Rightarrow \neg\beta)$,
- $\alpha \Leftrightarrow \beta \equiv (\alpha \Rightarrow \beta) \wedge (\beta \Rightarrow \alpha) \equiv (\neg\alpha \vee \beta) \wedge (\neg\beta \vee \alpha)$,
- $\neg(\alpha \vee \beta) \equiv \neg\alpha \wedge \neg\beta$,
- $\neg(\alpha \wedge \beta) \equiv \neg\alpha \vee \neg\beta$.

double négation	$\neg(\neg\alpha) \equiv \alpha$
associativité	$(\alpha \vee \beta) \vee \gamma \equiv \alpha \vee (\beta \vee \gamma)$ $(\alpha \wedge \beta) \wedge \gamma \equiv \alpha \wedge (\beta \wedge \gamma)$
commutativité	$\alpha \vee \beta \equiv \beta \vee \alpha$ $\alpha \wedge \beta \equiv \beta \wedge \alpha$
idempotence	$\alpha \vee \alpha \equiv \alpha$ $\alpha \wedge \alpha \equiv \alpha$
distributivité	$\alpha \vee (\beta \wedge \gamma) \equiv (\alpha \vee \beta) \wedge (\alpha \vee \gamma)$ $\alpha \wedge (\beta \vee \gamma) \equiv (\alpha \wedge \beta) \vee (\alpha \wedge \gamma)$

TABLE 1.4 – Réécritures de formules propositionnelles.

1.2.3 Conséquence logique modulo une théorie

Par la suite, nous ferons référence à la notion de conséquence logique modulo une théorie laquelle étend la notion de conséquence logique précédente. À partir de celle-ci nous introduisons les notions d'impliquant et d'impliqué modulo une théorie auxquelles nous ferons référence par la suite.

Définition 51 (conséquence logique modulo une théorie)**.** *Soient K une théorie et $\alpha, \beta \in \mathcal{PROP}_\mathcal{V}$ deux formules propositionnelles. α est une conséquence logique de β modulo K, noté $\beta \models_K \alpha$, ssi $K \cup \{\beta\} \models \alpha$.*

Définition 52 (impliquant (impliqué) modulo une théorie)**.** *Soient K une théorie et $\alpha, \beta \in \mathcal{PROP}_\mathcal{V}$ deux formules propositionnelles. α est un impliquant de β modulo K et β un impliqué de α modulo K ssi β est une conséquence logique de α modulo K ($\alpha \models_K \beta$).*

Définition 53 (impliquant (impliqué) strict modulo une théorie)**.** *Soient K une théorie et $\alpha, \beta \in \mathcal{PROP}_\mathcal{V}$ deux formules propositionnelles. α est un impliquant (resp. impliqué) strict de β modulo K si α est un impliquant (resp. impliqué) de β modulo K et que β n'est pas un impliquant (resp. impliqué) de α modulo K.*

Définition 54 (impliquant (impliqué) premier modulo une théorie)**.** *Soient K une théorie et $\alpha, \beta \in \mathcal{PROP}_\mathcal{V}$ deux formules propositionnelles. α est un impliquant (resp. impliqué) premier de β modulo K si α est un impliquant (resp. impliqué) strict de β modulo K et qu'il n'existe aucune formule propositionnelle $\gamma \in \mathcal{PROP}_\mathcal{V}$ t.q. α soit un impliquant (resp. impliqué) strict de γ modulo K et que γ soit un impliquant (resp. impliqué) strict de β modulo K.*

Exemple 8. *Soient $K = \{dis\,joncteur\_ok\}$ une théorie et $\alpha, \beta \in \mathcal{PROP}_\mathcal{V}$ deux formules propositionnelles t.q. :*

$$\alpha = (interrupteur\_on \wedge ampoule\_ok) \Rightarrow piece\_eclairee,$$
$$\beta = (interrupteur\_on \wedge disjoncteur\_ok) \Rightarrow piece\_eclairee.$$

Il est clair que $K \cup \{\beta\} \models \alpha$, α est donc une conséquence logique de β modulo K. Ainsi, β est un impliquant (strict) de α modulo K et α est un impliqué (strict) de β modulo K.

Équivalence de formules modulo une théorie

À l'instar de la notion de conséquence logique classique, à partir de la notion de conséquence logique modulo une théorie il est possible de définir une relation d'équivalence entre formules propositionnelles modulo une théorie.

Définition 55 (équivalence logique modulo une théorie)**.** *Soient K une théorie et $\alpha, \beta \in \mathcal{PROP}_{\mathcal{V}}$ deux formules propositionnelles. α et β sont logiquement équivalentes modulo K, noté $\alpha \equiv_K \beta$, ssi $\alpha \models_K \beta$ et $\beta \models_K \alpha$.*

1.3 Théorie de la démonstration

Contrairement à la théorie des modèles (voir Section 1.2), la théorie de la démonstration est une théorie purement syntaxique. En effet, le mode de déduction utilisé est purement formel et se définit uniquement en termes d'application de *règles d'inférence* sur des formules propositionnelles, sans s'intéresser à leur valeur de vérité.

Ainsi, à partir de règles d'inférences et d'*axiomes*, il sera possible de produire de nouvelles formules que nous appellerons *théorèmes* qui pourront servir à leur tour à produire de nouveaux théorèmes, dans le but, comme pour la théorie des modèles, de prouver la validité d'une formule propositionnelle [3].

Bien que nous nous limitons ici à ne présenter que le "système à la Hilbert", d'autres *systèmes déductifs* pour le calcul des propositions existent. Citons parmi eux la "déduction naturelle" de Gentzen (voir [Gentzen 1935a]) ou son "calcul des séquents" (voir [Gentzen 1935b] et [Gentzen *et al.* 1955]), ou encore le célèbre "principe de résolution" [4] de Robinson (voir [Robinson 1965]).

1.3.1 Axiomatique du calcul propositionnel

Définition 56 (règle d'inférence/axiome)**.** *Soient $\alpha_1, \ldots, \alpha_n, \gamma \in \mathcal{PROP}_{\mathcal{V}}$ des formules propositionnelles. Une règle d'inférence se présente sous la forme :*

$$\frac{\alpha_1 \quad \ldots \quad \alpha_n}{\gamma}$$

3. Plus exactement, le but est de démontrer des théorèmes, lesquels sont, selon le *théorème d'adéquation*, des formules valides (voir Sous-section 1.3.3).

4. Intuitivement, la résolution consiste en une unique règle d'inférence s'appliquant a des formules transformées en clauses. Elle est principalement utilisée dans les *systèmes de preuves automatiques* et est à la base du langage de programmation logique *Prolog*. Notons que la résolution s'utilise le plus souvent pour effectuer des *preuves par réfutation*. En effet, pour prouver par exemple que la formule γ est une conséquence logique de l'ensemble $\{\alpha_1, \ldots, \alpha_n\}$, il suffit de démontrer que l'ensemble $\{\alpha_1, \ldots, \alpha_n, \neg\gamma\}$ est incohérent.

où $\alpha_1, \ldots, \alpha_n$ sont les prémisses de la règle et γ sa conclusion, et, exprime que l'on peut inférer γ à condition que $\alpha_1, \ldots, \alpha_n$ soient déjà inférés.

Quand $n = 0$, une règle d'inférence est aussi appelée axiome et exprime que γ peut être produit sans condition [5].

Définition 57 (théorème)**.** *Soit $\alpha \in \mathcal{PROP}_{\mathcal{V}}$ une formule propositionnelle. α est un théorème, noté $\vdash \alpha$, si α est un axiome ou si α est obtenue à partir de l'application d'une règle d'inférence sur d'autres théorèmes.*

La table 1.5 [6], où $\alpha, \beta, \gamma \in \mathcal{PROP}_{\mathcal{V}}$ sont trois formules propositionnelles, rassemble un ensemble de schémas d'axiomes du calcul propositionnel. Cet ensemble est celui de plus petit cardinal connu (dit *minimal*) pour le calcul propositionnel muni des connecteurs logiques $\neg$, $\vee$, $\wedge$, $\Rightarrow$ et $\Leftrightarrow$ [7]. Un certain nombre de schémas d'axiomes très naturels peuvent être déduits de cet ensemble de schémas d'axiomes minimal pour le calcul propositionnel (voir Table 1.6, où $\alpha, \beta, \gamma \in \mathcal{PROP}_{\mathcal{V}}$ sont trois formules propositionnelles).

axiome K	$\alpha \Rightarrow (\beta \Rightarrow \alpha)$
axiome S	$(\alpha \Rightarrow (\beta \Rightarrow \gamma)) \Rightarrow ((\alpha \Rightarrow \beta) \Rightarrow (\alpha \Rightarrow \gamma))$
contraposition	$(\neg\alpha \Rightarrow \neg\beta) \Rightarrow (\beta \Rightarrow \alpha)$

TABLE 1.5 – Ensemble de schémas d'axiomes minimal.

raisonnement par l'absurde	$(\neg\alpha \Rightarrow \beta) \Rightarrow ((\neg\alpha \Rightarrow \neg\beta) \Rightarrow \alpha)$
conjonction	$\alpha \Rightarrow (\beta \Rightarrow (\alpha \wedge \beta))$ $(\alpha \wedge \beta) \Rightarrow \alpha$ $(\alpha \wedge \beta) \Rightarrow \beta$
disjonction	$\alpha \Rightarrow (\alpha \vee \beta)$ $\beta \Rightarrow (\alpha \vee \beta)$
raisonnement par cas	$(\alpha \vee \beta) \Rightarrow ((\alpha \Rightarrow \gamma) \Rightarrow ((\beta \Rightarrow \gamma) \Rightarrow \gamma))$

TABLE 1.6 – Schémas d'axiomes supplémentaires.

Intéressons nous maintenant à l'aspect déductif du langage. Pour ce faire, nous introduisons une règle d'inférence, appelée *modus ponens* [8], qui permet de construire de nouveaux théorèmes (voir Figure 1.1). Cette règle d'inférence exprime que "Si α est un théorème et que $\alpha \Rightarrow \beta$ est un théorème, alors β est un théorème aussi".

5. Dans ce cas et afin d'alléger la notation, la barre horizontale de la règle d'inférence séparant prémisses et conclusion sera omise.

6. Les noms des axiomes K et S proviennent de ceux des axiomes (ou "combinateurs") de la logique combinatoire qui leur sont "proches". Rappelons que la logique combinatoire est une théorie générale des fonctions, alternative directe au λ-calcul, principalement due à Curry (voir [Curry & Feys 1958] et [Curry *et al.* 1972]).

7. Notons qu'il existe une axiomatique complète de la logique propositionnelle à l'aide d'un seul axiome et d'un connecteur appelé "barre de Sheffer" dont la table de vérité correspond à celle du *non et*.

8. Le terme *modus ponens*, ou plus exactement *modus ponendo ponens*, vient du latin "ce que l'on pose afin d'en tirer conclusion" (*ponens* est le participe présent de verbe latin *ponere* qui signifie poser).

$$\frac{\vdash \alpha \quad \vdash \alpha \Rightarrow \beta}{\vdash \beta}$$

FIGURE 1.1 – Règle d'inférence du *modus ponens*.

1.3.2 Démonstration & déduction

Définition 58 (démonstration)**.** *Une démonstration d'un théorème α est un arbre $\mathcal{T}$ dont les nœuds sont des formules propositionnelles t.q. :*

1. *pour toute feuille de la forme $\vdash \alpha_i$, α_i est un axiome,*
2. *tout nœud $\vdash \alpha_i$ qui n'est pas une feuille a comme parents un couple $(\vdash \alpha_j, \vdash \alpha_k)$ t.q. $\dfrac{\vdash \alpha_j \quad \vdash \alpha_k}{\vdash \alpha_i}$ est une règle d'inférence,*
3. *$\vdash \alpha$ est la racine de l'arbre.*

Définition 59 (déduction)**.** *On dit qu'une formule α se déduit d'un ensemble Σ de formules propositionnelles appelées hypothèses, noté $\Sigma \vdash \alpha$ ssi il existe $\mathcal{T}$ un arbre dont les nœuds sont des formules propositionnelles t.q. :*

1. *pour toute feuille de la forme $\vdash \alpha_i$, α_i est soit un axiome soit une formule de Σ,*
2. *tout nœud $\vdash \alpha_i$ qui n'est pas une feuille a comme parents un couple $(\vdash \alpha_j, \vdash \alpha_k)$ t.q. $\dfrac{\vdash \alpha_j \quad \vdash \alpha_k}{\vdash \alpha_i}$ est une règle d'inférence,*
3. *$\vdash \alpha$ est la racine de l'arbre.*

Une *démonstration* d'un théorème est définie comme une "suite finie" d'applications de règles d'inférences et d'axiomes. La notion de *déduction*, quant à elle, résulte d'un prolongement de la notion de démonstration. La différence étant qu'une déduction permet de tenir compte de formules supplémentaires, les hypothèses.

Ainsi, nous pouvons donc maintenant, via les schémas d'axiomes des tables 1.5 et 1.6 et de la règle d'inférence du *modus ponens* (voir Figure 1.1), construire des démonstrations et des déductions de théorèmes, comme dans l'exemple suivant.

Exemple 9. *Soit Σ un ensemble composé des formules propositionnelles suivantes :*

1. *interrupteur_on $\wedge$ ampoule_ok,*
2. *interrupteur_on $\Rightarrow$ piece_eclairee .*

$\Sigma \vdash$ piece_eclairee. En effet, la figure 1.2 montre qu'il existe un arbre dont la racine est piece_eclairee t.q. :

- *(interrupteur_on $\wedge$ ampoule_ok) $\Rightarrow$ interrupteur_on est un axiome (voir Table 1.6),*
- *interrupteur_on $\wedge$ ampoule_ok $\in \Sigma$ est une hypothèse,*
- *interrupteur_on est obtenue par l'application du modus ponens sur les deux formules précédentes,*
- *interrupteur_on $\Rightarrow$ piece_eclairee $\in \Sigma$ est une hypothèse,*
- *et piece_eclairee est obtenue par l'application du modus ponens sur les deux formules précédentes.*

$$\dfrac{\dfrac{\vdash (interrupteur\_on \wedge ampoule\_ok) \Rightarrow interrupteur\_on \quad \vdash interrupteur\_on \wedge ampoule\_ok}{\vdash interrupteur\_on} \qquad \vdash interrupteur\_on \Rightarrow piece\_eclairee}{\vdash piece\_eclairee}$$

FIGURE 1.2 – Arbre de déduction de l'exemple 9.

1.3.3 Correspondance avec la théorie des modèles

La théorie des modèles et la théorie de la démonstration sont "équivalentes" pour la logique propositionnelle. Ainsi, l'on peut indifféremment utiliser l'une ou l'autre pour prouver la "vérité" d'une formule propositionnelle. Ce résultat repose sur un certain nombre de propriétés fondamentales que possède la logique propositionnelle, que sont l'adéquation, la cohérence et la complétude forte (voir Propriété 2).

Propriété 2. *Soient $\alpha \in \mathcal{PROP}_{\mathcal{V}}$ une formule propositionnelle et $\Sigma = \{\beta \text{ t.q. } \beta \in \mathcal{PROP}_{\mathcal{V}}\}$ un ensemble de formules propositionnelles. La table 1.7 dresse quelques propriétés fondamentales que possède la logique propositionnelle.*

Adéquation	si $\vdash \alpha$ alors $\models \alpha$
Cohérence	si $\vdash \alpha$ alors $\nvdash \neg\alpha$
Complétude forte	si $\Sigma \models \beta$ alors $\Sigma \vdash \beta$

TABLE 1.7 – Propriétés fondamentales de la logique propositionnelle.

Les propriétés d'adéquation et de complétude permettent de relier l'interprétation sémantique (théorie des modèles) et l'interprétation syntaxique (théorie de la démonstration) de la logique. Il est donc équivalent de démontrer une formule du calcul propositionnel ou de calculer sa table de vérité. Les notions de véracité et de démontrabilité sont ainsi équivalentes : si une proposition est valide, on doit pouvoir la démontrer et réciproquement, toute proposition que l'on parvient à démontrer doit être valide.

1.4 Problèmes techniques

De nombreux problèmes techniques s'inscrivent dans le cadre de la logique propositionnelle. Dans cette section, nous allons introduire deux d'entre eux auxquels nous ferons référence dans la suite du manuscrit, à savoir les problèmes intimement liés de *satisfiabilité d'une formule propositionnelle* et d'*extraction de sources minimales d'incohérence*. Pour une présentation complète de ces deux problèmes, le lecteur pourra se référer à [Saïs 2008] par exemple.

1.4.1 Problème de satisfiabilité

Le problème de satisfiabilité, abrégé en problème SAT, lequel consiste à déterminer si une formule booléenne mise sous forme *CNF* possède ou non un modèle (autrement dit est satisfiable ou non), est en I.A. un axe de recherche majeur. D'un côté, ce problème étant

un problème *NP*-complet (voir [Cook 1971]), une éventuelle résolution de celui-ci en temps polynomial permettrait de répondre à l'une des plus grandes questions de la théorie de la complexité, à savoir l'affirmation que $P = NP$. D'un autre côté, d'un point de vue purement pratique, le problème SAT est la pierre angulaire de nombreuses applications en I.A. : démonstration automatique de théorème, inférence non monotone, vérification formelle de matériel et de logiciel, ou encore recherche opérationnelle pour ne citer que quelques exemples.

Définition 60 (problème SAT)**.** *Soit α une formule propositionnelle au format CNF. Le problème de décision SAT consiste à déterminer si α est satisfiable et à fournir un modèle de α dans le cas positif.*

Le problème complémentaire au problème SAT, appartenant donc à la classe de complexité *coNP*, est appelé problème d'insatisfiabilité, abrégé en problème UNSAT.

Définition 61 (problème UNSAT)**.** *Soit α une formule propositionnelle au format CNF. Le problème de décision UNSAT consiste à déterminer si α n'est pas satisfiable.*

Lorsque une formule α n'est pas satisfiable, on s'intéresse au problème de trouver une interprétation qui permette de satisfaire le plus grand nombre de clauses de α. Ce problème d'optimisation, intimement lié au problème de décision SAT, est connu sous le nom de problème Max-SAT. Ce problème est *NP*-dur puisque sa solution mène facilement à la solution du problème SAT.

Définition 62 (problème Max-SAT)**.** *Soit α une formule propositionnelle au format CNF. Le problème d'optimisation Max-SAT consiste à déterminer le nombre maximal de clauses de α qui peuvent être satisfaites par une même interprétation.*

Exemple 10. *Soit $\Sigma = \{a, \neg a, b, \neg b \vee c, \neg c, c \vee \neg d, d, d \vee e\}$ une formule au format CNF mise sous la forme d'un ensemble de clauses.*

Σ ne possède aucun modèle et est donc insatisfiable. Une même interprétation peut satisfaire au plus six clauses de Σ. En effet, les deux sous-ensembles de Σ suivants sont satisfiables :

1. $\Sigma' = \{a, b, \neg b \vee c, c \vee \neg d, d, d \vee e\}$,
2. $\Sigma'' = \{\neg a, b, \neg b \vee c, c \vee \neg d, d, d \vee e\}$.

1.4.2 Extraction des sources minimales d'incohérence

Comme nous venons de le voir, lorsque le résultat du test de satisfiabilité d'une formule booléenne s'avère positif, un modèle de la formule est fourni. En revanche, lorsque le résultat du test de satisfiabilité est négatif, aucune explication quant aux raisons de ce résultat n'est fournie. Le problème Max-SAT ne s'intéresse pas non plus aux raisons qui l'amènent à conclure que le nombre maximal de clauses d'une formule pouvant être satisfaites par une même interprétation diffère du nombre total de clauses de cette même formule.

Pourtant, en règle générale, seules quelques parties minimes de la formule la rendent incohérente : pouvoir les réparer permet donc d'en rétablir la cohérence. Ainsi, dans cette perspective, il est très utile de pouvoir extraire les sources minimales d'incohérence d'une formule propositionnelle au format *CNF*. Ici, nous introduisons brièvement deux concepts différentes permettant d'extraire les sources minimales d'incohérence d'une formule insatisfiable. Pour de plus amples détails sur le sujet, le lecteur peut aussi consulter [Piette 2007] par exemple.

Noyaux minimaux incohérents

Les noyaux minimaux incohérents (ou MUSes pour "Minimally Unsatisfiable Subformulae") d'une formule au format CNF constituent l'explication la plus fine, en terme d'ensembles minimaux de clauses, de l'incohérence d'une formule et désignent quelles parties de la formule sont sources de l'incohérence.

Définition 63 (noyau minimal incohérent)**.** *Considérons une formule α au format CNF mise sous la forme d'un ensemble Σ de clauses. Un MUS Γ de Σ est un ensemble de clauses tel que :*

- $\Gamma \subseteq \Sigma$,
- Γ *est incohérent,*
- *tout sous-ensemble propre de* Γ *est cohérent.*

Il est important de noter que la recherche d'un des MUSes d'une formule incohérente est un problème de lourde complexité algorithmique. Décider si une formule au format CNF est un MUS est un problème DP-complet (voir [Papadimitriou & Wolfe 1988]). En effet, il faut d'abord vérifier l'insatisfiabilité de la formule puis la satisfiabilité de chacune des ses plus grandes sous-formules. On comprend alors son appartenance à la classe DP (laquelle, rappelons le, est composée de l'intersection d'un langage de NP et d'un langage de $coNP$).

De plus, comme le montre l'exemple suivant, une formule au format CNF peut posséder plusieurs MUSes. Pour être plus précis, elle peut même en contenir un nombre qui est exponentiel par rapport à la taille de la formule. Dans le pire des cas, une formule au format CNF composée de n clauses peut posséder $C_n^{n/2}$ MUSes. Ce nombre potentiellement élevé de MUSes au sein d'une même formule rend les problèmes qui y sont corrélés d'une complexité algorithmique très élevée. Pour preuve, tester si une formule appartient à l'ensemble des MUSes d'une formule incohérente a été prouvé Σ_2^P-difficile (voir [Eiter & Gottlob 1992]).

Notons que pour restaurer la cohérence d'une formule au format CNF, il est nécessaire de retirer au moins une clause de chacun des MUSes. Notons aussi que certains MUSes peuvent partager des clauses. Ainsi pour ces MUSes, retirer une de ces clauses suffit.

Exemple 11. *Reprenons l'exemple 10. Soit $\Sigma = \{a, \neg a, b, \neg b \vee c, \neg c, c \vee \neg d, d, d \vee e\}$ une formule au format CNF mise sous la forme d'un ensemble de clauses.*

Σ est incohérente. En effet, Σ possède trois MUSes :

- $MUS_1 = \{a, \neg a\}$,
- $MUS_2 = \{b, \neg b \vee c, \neg c\}$,
- $MUS_3 = \{\neg c, c \vee \neg d, d\}$.

Le diagramme de Venn de la figure 1.3 met en évidence que les sources d'incohérence de Σ peuvent partager des clauses ($\neg c$ appartient à deux MUSes de Σ).

Notons que la clauses $d \vee e$ n'appartient à aucun MUS de Σ et qu'elle ne participe donc pas à l'incohérence de Σ.

Couvertures incohérentes strictes

Comme nous venons de le voir, calculer de manière exhaustive tous les MUSes d'une formule peut s'avérer intraitable d'un point de vue calculatoire, leur nombre pouvant être

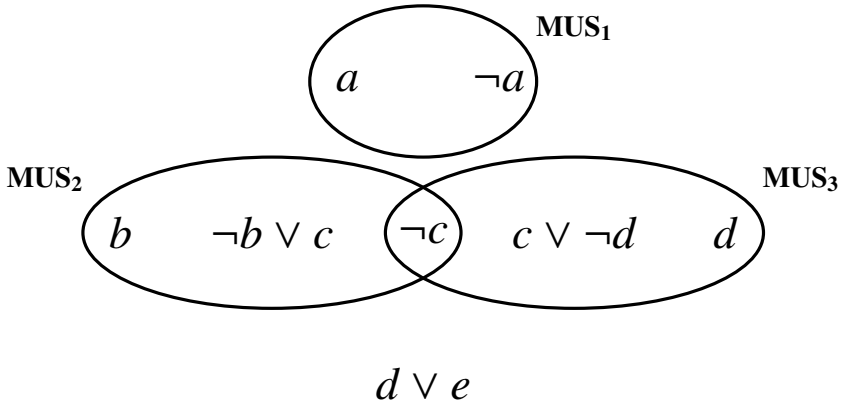

FIGURE 1.3 – Diagramme de Venn des MUSes de la formule CNF de l'exemple 11.

exponentiel en la taille de la formule dans le pire des cas. Si pour de nombreux problèmes, le nombre de MUSes est souvent limité, rien ne peut cependant le garantir dans le cas général. Il semble donc naturel de chercher à *approximer* l'ensemble des MUSes d'une formule le plus précisément possible, dans le but, là aussi, de rétablir sa cohérence.

Grégoire, Mazure et Piette (voir [Grégoire *et al.* 2006b]) proposent de rétablir la cohérence d'une formule en ne retirant que les MUSes *indépendants* (c.-à-d. ne partageant aucune clause) de cette dernière. Cette idée est principalement motivée par le fait que des MUSes indépendants expriment des causes indépendantes, ou non corrélées, d'incohérence au sein d'une même formule. Une telle approximation doit fournir une *couverture* suffisante de cause d'incohérence tout en garantissant un nombre polynomial de MUSes.

Définition 64 (ensembles de clauses indépendants [Grégoire *et al.* 2006b])**.** *Deux ensembles de clauses sont dits indépendants ssi leur intersection est vide.*

Définition 65 (couverture incohérente stricte [Grégoire *et al.* 2006b])**.** *Une couverture incohérente stricte* Γ' *d'une formule* α *au format CNF mise sous la forme d'une ensemble* Σ *de clauses est une union ensembliste de MUSes indépendants de* Σ *t.q.* $\Sigma \setminus \Gamma'$ *est satisfiable,*

En reprenant l'exemple précédent, nous pouvons constater qu'une même formule incohérente peut posséder plusieurs couvertures incohérentes strictes.

Exemple 12. *Reprenons l'exemple 11.* Σ *possède trois MUSes identifiables à l'aide du diagramme de Venn de la figure 1.3.*

Σ *possède deux couvertures incohérentes strictes :*

- $IC_1 = MUS_1 \cup MUS_2$,
- $IC_2 = MUS_1 \cup MUS_3$.

Bien qu'une couverture incohérente ne nous fournit pas l'ensemble de tous les MUSes présents dans une formule, elle encode une série d'explications minimales pour l'incohérence, ce qui est suffisant pour expliquer assez de sources d'incohérence dans le but de retrouver la cohérence si ces dernières sont réparées.

Notons toutefois que l'unicité n'est pas garantie avec cette méthode. En effet, dans le cas où plusieurs couvertures incohérentes strictes existent, la formule au format CNF résultante diffère selon la couverture incohérente stricte considérée lors de l'opération de retrait.

Chapitre 2

Logiques non monotones

Sommaire

Bien que la logique classique ait pour objet l'étude rigoureuse des formes de raisonnement, elle se limite pourtant aux seuls raisonnements *absolument corrects*, ne permettant donc pas de modéliser des *raisonnements révisables*. Ces raisonnements font pourtant partie intégrante des capacités naturelles qu'a l'être humain à tirer des conclusions *simplement plausibles* et à les rétracter à la lueur de nouvelles informations qui les contredisent. Pour reprendre notre exemple canonique, en apprenant que l'interrupteur d'une pièce est enclenché, nous concluons logiquement que la pièce est éclairée. Mais si par la suite nous apprenons que l'ampoule est cassée, nous révisons alors notre jugement et rétractons tout naturellement notre raisonnement précédent et ceux qui en découlent. La logique classique, quant à elle, en est totalement incapable.

Cette lacune s'explique par le fait que, comme nous avons pu le voir dans le chapitre précédent au travers de la logique propositionnelle, la logique classique a vu le jour afin de formaliser des raisonnements "mathématiques". Elle souffre ainsi de la propriété de monotonie qui leur est intimement liée et qui exprime qu'un raisonnement établi ne peut être rétracté (voir Table 1.3).

Dans ce chapitre, nous introduisons la famille des "logiques non monotones", lesquelles se proposent d'étendre la logique classique afin de permettre un raisonnement révisable et pallier ainsi ce problème de la logique classique. Parmi elles, nous étudions plus particulièrement la *logique des défauts* (voir [Reiter 1980]), certainement la plus connue et la plus étudiée des logiques non monotones qui permettent un *raisonnement par défaut*. Elle utilise pour cela des *règles d'inférence avec exceptions* qui permettent d'exprimer des connaissances de la forme "S'il est cohérent de penser que l'ampoule n'est pas cassée et si l'interrupteur est enclenché, alors la pièce est éclairée", qui ne nécessitent pas de prouver que "L'ampoule n'est pas cassée" pour conclure que "La pièce est éclairée", conclusion qui peut par la suite être

rétractée s'il s'avère que "L'ampoule est cassée". Dans ce chapitre, nous introduisons aussi la *circonscription* (voir [McCarthy 1980] et [McCarthy 1986]), autre approche permettant un raisonnement révisable.

Ce chapitre fournit une présentation succincte de la notion de raisonnement révisable au travers de certaines logiques non monotones auxquelles nous ferons référence dans la suite de ce manuscrit. Le lecteur trouvera une présentation plus complète dans la littérature (voir [Gabbay & Robinson 1994] et [Grégoire 1990] par exemple).

Le chapitre est organisé comme suit. Dans la section 2.1, nous introduisons la notion de raisonnement révisable et les différentes caractéristiques que doivent revêtir les formalismes permettant un tel raisonnement. Dans la section 2.2, nous présentons le raisonnement par défaut au travers de la logique des défauts. La section 2.3 est quant à elle vouée à la présentation de la circonscription.

2.1 Raisonnement révisable

Une grande part de l'intelligence de l'être humain réside dans sa capacité à raisonner à partir d'informations bien souvent *incomplètes* et *évolutives*. En effet, dans l'absolu la plupart de nos raisonnements sont *simplement plausibles* et peuvent devoir être révisés à la lueur de nouvelles informations qui les contredisent. Seulement, la logique classique ayant été élaborée afin de formaliser des raisonnements mathématiques "absolument corrects" n'est pas en capacité de formaliser ces *raisonnements révisables* : sa propriété de monotonie l'en empêche.

Pourtant, de nombreux domaines de l'I.A. comme le *diagnostic*, le *traitement du langage naturel*, la *spécification* et la *vérification* de programmes et de systèmes, ou encore l'*ordonnancement*, nécessitent de pouvoir réviser des raisonnements.

Ainsi, différentes logiques qui étendent la logique classique afin de permettre un raisonnement révisable ont été développées durant ces trente dernières années. Elles sont qualifiées de *non monotones* puisqu'elles ne revêtent pas la propriété de monotonie de la logique classique : le nombre de conclusions qu'elles permettent d'obtenir peut décroître lorsque le nombre de leurs prémisses augmente. Mentionnons parmi elles les logiques modales non monotones (voir [McDermott 1982]), la logique autoépistémique (voir [Moore 1985]), l'hypothèse de monde clos (voir [Reiter 1977]), la complétion de prédicat (voir [Clark 1978]), la logique des défauts (voir [Reiter 1980]) ou encore la circonscription (voir [McCarthy 1980] et [McCarthy 1986]).

2.1.1 Caractéristiques des logiques non monotones

Les logiques monotones ont pour but la mise au point de systèmes d'inférence permettant la modélisation de raisonnements révisables et donc non valides au sens classique du terme. Intuitivement, ces systèmes d'inférence doivent permettre l'obtention de formules "plausibles" dans le sens où elles ne sont pas valides classiquement avec un ensemble de prémisses mais uniquement "cohérentes" avec celui-ci. Pour reprendre notre exemple canonique, l'information exprimant que "La pièce est éclairée" n'est pas une conséquence valide des prémisses exprimant d'un côté que "Si l'interrupteur est enclenché et qu'il est cohérent de penser que

l'ampoule n'est pas cassée, alors la pièce est éclairée" et d'un autre côté que "L'interrupteur est enclenché" ; elle est simplement cohérente avec ces dernières.

D'autre part, il semble raisonnable d'exiger que des conclusions mutuellement ne puissent être inférées conjointement. Même si le raisonnement de ces formalismes non monotones est simplement plausible, on peut exiger une cohérence entre ses différentes conclusions. Conclure à la fois que la pièce est éclairée et qu'elle ne l'est pas ne sera naturellement pas accepté. Cette exigence de *cohérence* est liée à la révisabilité du raisonnement. Lorsque de nouvelles informations entrent en jeu, des conjectures peuvent ne plus être cohérentes avec le nouvel ensemble de prémisses et doivent dès lors être rétractées. Pour reprendre notre exemple canonique, si l'on apprend que "L'ampoule est cassée" alors notre raisonnement à propos du fait que "La pièce est éclairée" n'est plus cohérent et est donc rétracté. Une logique non monotone permettra donc d'inférer des informations cohérentes entre elles et cohérentes avec un ensemble donné de prémisses.

Cela a pour conséquence qu'à partir d'un même ensemble de prémisses, plusieurs ensembles incompatibles de conclusions possibles peuvent éventuellement être obtenus. Cette caractéristique est appelée propriété de *pluri-extensionalité*, les ensembles de conclusions possibles en question sont souvent appelés *extensions*. Notons que la caractérisation et la construction des extensions sont souvent difficiles car certains raisonnements révisables sont *circulaires* : on ne peut inférer certaines conclusions qu'après avoir vérifié que d'autres résultats ne peuvent pas être obtenus. Ces extensions sont souvent caractérisées à partir du système opérant sur un ensemble de prémisses par l'évaluation des "points fixes" de cette opération. Un point fixe est un ensemble stable de croyances pour lequel aucune nouvelle formule ne peut plus être inférée de façon cohérente. Cette technique est directement appliquée dans les logiques non monotones de McDermott (voir [McDermott 1982]), la logique des défauts (voir [Reiter 1980]) et la logique autoépistémique (voir [Moore 1985]).

Dans la suite, nous présentons uniquement la logique des défauts (voir [Reiter 1980]) et la circonscription (voir [McCarthy 1980] et [McCarthy 1986]), la plupart des logiques non monotones pouvant être retranscrites de l'une à l'autre pour des fragments importants.

2.2 Logique des défauts

Dans la communauté de la recherche en I.A., l'un des outils les plus populaires pour manier des formes de raisonnement révisable est certainement la logique des défauts de Reiter (voir [Reiter 1980]) et ses principales variantes (voir [Lukaszewicz 1988] , [Brewka 1991], [Schaub 1992] et [Mikitiuk & Truszczynski 1993]). La logique des défauts a été introduite et développée pour formaliser le raisonnement simplement cohérent. En présence d'informations incomplètes, nous sommes amenés à tirer des conclusions conjecturales simplement plausibles. En particulier, nous interprétons parfois comme absolument générales des lois qui apparaissent correctes dans la plupart des cas, mais qui admettent certaines exceptions. Pour reprendre notre exemple canonique, si l'interrupteur d'une pièce est enclenché, alors nous inférons naturellement que le pièce est éclairée. Il existe pourtant des situations où cette affirmation n'est pas correcte (panne de courant, ampoule cassée...) mais de manière générale, nous nous sentons autorisés à conclure que la pièce est éclairée si il n'existe rien dans nos

croyances qui nous l'interdise. Ce genre de raisonnement est appelé *raisonnement par défaut*.

La logique des défauts permet de formaliser le raisonnement par défaut sous la forme de règles d'inférence particulières appelées *défauts* se représentant comme suit :

$$\frac{\alpha : \beta}{\gamma}$$

exprimant que si α est cru et si β est cohérent avec tout ce qui est cru, alors γ peut être cru également. Ainsi, le raisonnement de notre exemple canonique exprimant qu'une pièce est *généralement* considérée comme éclairée lorsque l'interrupteur est enclenché, peut se représenter sous la forme du défaut suivant :

$$\frac{interrupteur\_on : piece\_eclairee}{piece\_eclairee}$$

lequel exprime que "Si l'interrupteur est enclenché et s'il est cohérent de penser que la pièce est éclairée, alors on peut conclure que la pièce est éclairée". On représente ainsi la loi générale, munie d'exceptions. Une règle de ce genre permet de traiter les cas d'exceptions sans nécessiter l'identification préalable de ces cas.

Un système de logique des défauts se présentera sous la forme d'une *théorie avec défauts*, constituée d'un ensemble particulier de formules et de règles d'inférence. L'ensemble en question contient des formules de la logique propositionnelle [1] représentant les informations de base données au système et traitées en tant qu'*axiomes* ; il contient en outre des défauts qui traduisent les différentes *lois comprenant des exceptions*.

Pour un tel système il existe un certain nombre d'ensembles cohérents de croyances appelés *extensions* (zéro, un ou plusieurs ensembles) qui peuvent en être inférées. Ces ensembles représentent les différentes images du monde qui peuvent être imaginées à partir de la théorie avec défauts.

Dans la suite nous introduisons les notions de *théorie avec défauts* et d'*extensions*. Nous discuterons aussi des quelques faiblesses de la définition de Reiter et des solutions existantes mises en place pour les inhiber. Pour une présentation plus complète le lecteur peut se référer aux ouvrages faisant référence (voir [Besnard 1989] par exemple).

2.2.1 Théorie avec défauts

Définition 66 (règle de défaut [Reiter 1980])**.** *Soient $\alpha, \beta_1, \ldots, \beta_m$ et γ des formules de $\mathcal{PROP}_{\mathcal{V}}$. Une règle de défaut (en abrégé défaut) $\mathcal{D}$ est une expression de la forme :*

$$\frac{\alpha : \beta_1, \ldots, \beta_m}{\gamma}$$

où $\alpha, \beta_1, \ldots, \beta_m$ et γ sont appelés respectivement pré-requis, justificatifs et conséquent du défaut $\mathcal{D}$, et, sont respectivement notés $pre(\mathcal{D})$, $just(\mathcal{D})$ et $cons(\mathcal{D})$.

Un défaut dont l'unique justificatif et le conséquent sont équivalents est appelé défaut *normal*. Quand un défaut normal ne possède pas de pré-requis, il est appelé défaut *supernormal*.

1. Dans ce chapitre le langage de représentation est propositionnel. Pour une présentation dans le cadre prédicatif, le lecteur peur se référer aux ouvrages faisant référence (voir [Besnard 1989] entre autres).

Définition 67 (théorie avec défauts [Reiter 1980])**.** *Une théorie avec règles de défauts (en abrégé théorie avec défauts)* Γ *est une paire* (Δ, Σ) *où :*

- Δ *est un ensemble de défauts,*
- Σ *est un ensemble de formules propositionnelles de* $\mathcal{PROP}_{\mathcal{V}}$.

Exemple 13. *Reprenons notre exemple canonique exprimant d'un côté que "Si l'interrupteur est enclenché et qu'il est cohérent de penser que la pièce est éclairée alors la pièce est éclairée" et que "S'il fait nuit et qu'il est cohérent de penser que la pièce n'est pas éclairée alors la pièce n'est pas éclairée", et, d'un autre côté que "L'interrupteur est enclenché" et que "Il fait nuit". Cette exemple peut être représenté sous la forme de la théorie avec défauts* $\Gamma = (\Delta, \Sigma)$ *où :*

- $\Delta = \{\frac{interrupteur\_on : piece\_eclairee}{piece\_eclairee}, \frac{nuit : \neg piece\_eclairee}{\neg piece\_eclairee}\}$,
- $\Sigma = \{interrupteur\_on, nuit\}$.

Dans cet exemple, les deux défauts induisent des conclusions *mutuellement incohérentes*. Si l'on applique d'abord le premier, il sera alors impossible d'appliquer le deuxième et inversement. Ces deux conclusions appartiennent vraisemblablement à deux *extensions* différentes.

2.2.2 Extensions de théorie avec défauts

Une théorie avec défauts $\Gamma = (\Delta, \Sigma)$ sous-tend un certain nombre d'ensembles de croyances (zéro, un ou plusieurs ensembles) que l'on peut inférer de façon cohérente avec l'ensemble Σ de formules. Ces ensembles de croyances sont appelés *extensions* de la théorie avec défauts. Intuitivement, une extension est un sur-ensemble des informations de base du système qui comprend tout ce qui peut-être inféré, que ce soit par les règles logiques classiques ou par les défauts. Définir et calculer les extensions d'une théorie avec défauts n'est pas trivial. Premièrement, comme nous avons pu le voir, il y a une forme de *circularité* dans la définition et dans le calcul de ce qui peut être inféré. Pour décider si le conséquent d'un défaut doit être inféré, nous avons besoin de vérifier la cohérence de ses justificatifs. Cependant, ce test de cohérence revient à prouver que les opposés des justificatifs ne peuvent être inférés à leur tour. Dans le cas général, les *méthodes à point fixe* sont utilisées pour caractériser ce qui peut être inféré à partir d'une théorie avec défauts.

Définition 68 (extension d'une théorie avec défauts [Reiter 1980])**.** *Soit* $\Gamma = (\Delta, \Sigma)$ *une théorie avec défauts. Étant donné un sous-ensemble* S *de* $\mathcal{PROP}_{\mathcal{V}}$*, désignons par* $\Pi(S)$ *le plus petit sous-ensemble de* $\mathcal{PROP}_{\mathcal{V}}$ *satisfaisant aux trois conditions suivantes :*

- $\Sigma \subseteq \Pi(S)$,
- $Cn(\Pi(S)) = \Pi(S)$,
- *si* $\frac{\alpha : \beta_1, \ldots, \beta_m}{\gamma} \in \Delta$, *si* $\alpha \in \Pi(S)$ *et si* $\neg\beta_1, \ldots, \neg\beta_m \notin S$, *alors* $\gamma \in \Pi(S)$.

Un ensemble de formules $E \subseteq \mathcal{PROP}_{\mathcal{V}}$ est une extension pour Γ si et seulement si $\Pi(E) = E$, c'est-à-dire si et seulement si E est un *point-fixe* de l'opérateur Π. Une telle extension E peut être caractérisée de la manière suivante.

Définition 69 (construction d'une extension par point fixe [Reiter 1980])**.** *Construisons une suite de formules* E_i *en posant* $E_0 = \Sigma$ *et*

$$E_{i+1} = Cn(E_i) \cup \{\gamma | \frac{\alpha : \beta_1, \ldots \beta_m}{\gamma} \in \Delta\} \text{ où } \alpha \in E_i \text{ et } \neg\beta_1, \ldots, \neg\beta_m \notin E,$$

pour $i = 0, 1, 2$, etc. Alors l'ensemble donné E est une extension pour Γ si et seulement si $E = \bigcup_{i=0}^{\infty} E_i$.

Définition 70 (défaut générateur [Reiter 1980])**.** *Soient $\Gamma = (\Delta, \Sigma)$ une théorie avec défauts et E une extension de Γ. $d \in \Delta$ est un défaut générateur de Δ dans E si :*

$$pred(d) \in E \text{ et } \{\neg a \text{ t.q. } a \in just(d)\} \cap E = \emptyset.$$

L'ensemble de tous les défauts de Δ qui sont générateurs dans E est noté $GD(\Delta, E)$.

Il est ainsi possible de caractériser une extension par l'ensemble de ses défauts générateurs.

Théorème 1 ([Reiter 1980])**.** *Soient une théorie avec défauts $\Gamma = (\Delta, \Sigma)$, une extension E de Γ et son ensemble de défauts générateurs $GD(\Delta, E)$.*

$$E = Cn(\Sigma \cup cons(GD(\Delta, E))) \text{ où } cons(\Delta') = \{cons(d) \text{ t.q. } d \in \Delta'\}.$$

Dans la suite, nous ferons une distinction entre *raisonnement sceptique* et *raisonnement crédule* à partir d'une théorie Γ. Une formule (de la logique standard) f peut-être inférée de manière *sceptique* (resp. *crédule*) à partir d'une théorie Γ si et seulement si f appartient à toutes (resp. au moins une) extension(s) de Γ.

Exemple 14. *Reprenons notre exemple 13. $\Gamma = (\Delta, \Sigma)$ où :*

- $\Delta = \{\frac{interrupteur\_on : piece\_eclairee}{piece\_eclairee}, \frac{nuit : \neg piece\_eclairee}{\neg piece\_eclairee}\}$,
- $\Sigma = \{interrupteur\_on, nuit\}$.

Γ possède deux extensions à savoir :

- $E_1 = Cn(\{interrupteur\_on, nuit, piece\_eclairee\})$,
- $E_2 = Cn(\{interrupteur\_on, nuit, \neg piece\_eclairee\})$.

où :

- $GD(\Delta, E_1) = \{\frac{interrupteur\_on : piece\_eclairee}{piece\_eclairee}\}$,
- $GD(\Delta, E_2) = \{\frac{nuit : \neg piece\_eclairee}{\neg piece\_eclairee}\}$.

Notons que les formules interrupteur_on et nuit appartenant à Σ peuvent être inférées de manière sceptique puisqu'elles appartiennent logiquement aux deux extensions. Par contre, les conclusions piece_eclairee et ¬piece_eclairee des deux défauts de Δ ne peuvent être inférées que de manière crédule puisque étant mutuellement incohérentes, elles ne peuvent appartenir à la même extension.

2.2.3 Faiblesses de la définition de Reiter

Dans cette section, nous discutons des quelques faiblesses de la logique des défauts telle qu'elle a été définie par Reiter et des solutions existantes mises en place pour les inhiber.

Problèmes de la trivialisation

Lorsqu'une théorie avec défauts $\Gamma = (\Delta, \Sigma)$ contient un ensemble de formules classiques Σ incohérent, elle ne possède qu'une unique extension, laquelle contient tout le langage (voir [Reiter 1980]). Il est clair que dans une telle situation, la théorie est totalement inexploitable, tout et son contraire peut en être déduit. Pourtant, en accord avec les efforts fournis par la communauté en I.A. à l'étude du raisonnement en présence de connaissances incohérentes et au développement de techniques de tolérance à l'incohérence (voir [Bertossi *et al.* 2005], [Konieczny & Grégoire 2006], [Besnard & Hunter 1995] ou [Gabbay & Hunter 1991] par exemple), il semble tout à fait justifiable qu'une théorie avec défauts puisse contenir un ensemble de formules classiques incohérent. Pourtant, ce *problème de la trivialisation de la logique des défauts* n'a reçu que très peu d'intérêt dans la littérature.

Dans un travail tout à fait annexe au problème traité dans cette thèse, nous avons proposé une solution au problème de la trivialisation qui consiste intuitivement à remplacer chaque formule de Σ participant à l'incohérence par un défaut super-normal lui correspondant (voir [Besnard *et al.* 2009b] et [Besnard *et al.* 2009a]).

Problème de la non existence d'extension

De même qu'une théorie avec défauts $\Gamma = (\Delta, \Sigma)$ peut ne posséder qu'une seule extension contenant tout le langage, elle peut aussi n'en posséder aucune. En effet, comme le montrent les exemples 15, 16 et 17, cela se produit lorsque les conséquents de défauts applicables de Δ (c.-à-d. dont le pré-requis est déductible et les négations des justificatifs ne le sont pas) sont incohérents, contradictoires entre eux ou avec Σ.

Exemple 15. *Soit* $\Gamma = (\Delta, \Sigma)$ *une théorie avec défauts où :*

- $\Sigma = \{interrupteur\_on, \neg piece\_eclairee\}$,
- $\Delta = \{\dfrac{interrupteur\_on : ampoule\_ok}{piece\_eclairee}\}$.

Γ *ne possède aucune extension. En effet, l'unique défaut de* Δ *est applicable et son conséquent est contradictoire avec une formule de* Σ.

Exemple 16. *Soit* $\Gamma = (\Delta, \Sigma)$ *une théorie avec défauts où :*

- $\Sigma = \{interrupteur\_on\}$,
- $\Delta = \{\dfrac{interrupteur\_on : ampoule\_ok}{piece\_eclairee}, \dfrac{interrupteur\_on : \neg ampoule\_ok}{\neg piece\_eclairee}\}$.

Γ *ne possède aucune extension. En effet, les deux défauts de* Δ *sont applicables et leurs conséquents sont contradictoires.*

Exemple 17. *Soit* $\Gamma = (\Delta, \Sigma)$ *une théorie avec défauts où :*

- $\Sigma = \emptyset$,
- $\Delta = \{\dfrac{interrupteur\_on : ampoule\_ok}{piece\_eclairee \wedge \neg piece\_eclairee}\}$.

Γ *ne possède aucune extension. En effet, l'unique défaut de* Δ *est applicable et son conséquent est incohérent.*

Pour pallier ce problème, certaines variantes de la logique des défauts de Reiter ont été définies afin d'assurer l'existence d'au moins une extension (voir la logique des défauts justifiée de Lukaszewicz par exemple [Lukaszewicz 1988]).

Notons qu'il existe cependant une classe de théories avec défauts pour lesquelles l'existence d'extensions est assurée. Il s'agit des *théories normales* lesquelles ne contiennent que des défauts normaux (rappelons que ces défauts peuvent être utilisés pour exprimer des assertions du type "Normalement ou typiquement, les α sont des β"). Outre la propriété intéressante d'admettre au moins une extension, les théories normales possèdent une propriété de *semi-monotonie* : si on accroît l'ensemble des défauts d'une théorie normale, alors la nouvelle théorie normale ainsi obtenue admet une extension qui inclut une extension de la première théorie. Une conséquence pratique importante de cette propriété est la possibilité de construire une théorie de la preuve qui reste locale par rapport aux défauts mis en jeu (voir [Reiter 1980]).

Problème du choix des extensions

En logique des défauts, certaines situations mènent à établir des préférences entre extensions. Si l'on reprend l'exemple 13, il semble naturel de préférer l'extension indiquant que "La pièce est éclairée". En effet, bien que de manière générale une pièce n'est pas éclairée la nuit (c'est ce qu'exprime le deuxième défaut de la théorie), si l'interrupteur de la pièce est enclenché alors la pièce est éclairée, qu'il fasse nuit ou non.

Afin de tenir compte de ce problème, un certains nombre de variantes de la logique des défauts de Reiter, prenant en compte des priorités entre défauts, ont été proposées dans la littérature (voir [Brewka & Eiter 2000] par exemple).

2.3 Circonscription

Nous introduisons maintenant la *circonscription* de McCarthy (voir [McCarthy 1980] et [McCarthy 1986]), une autre approche de la littérature permettant un raisonnement révisable, laquelle est une généralisation de l'*hypothèse du monde clos* (voir [Reiter 1977]) [2]. Pour plus de détails sur la circonscription, nous invitons le lecteur à se référer à la littérature faisant référence (voir [Besnard 1989] par exemple).

Reprenons pour cela notre exemple canonique dans lequel l'on souhaite exprimer que "Généralement lorsque l'interrupteur d'une pièce est enclenché, alors la pièce est éclairée". Dans la section précédente, nous avons exprimé cela en logique des défauts par le biais d'un défaut du type :

$$\frac{interrupteur\_on : piece\_eclairee}{piece\_eclairee},$$

exprimant que "Si l'interrupteur est enclenché et s'il est cohérent de penser que la pièce est éclairée, alors la pièce est éclairée".

Dans le cas de la circonscription, McCarthy traduit cela par une règle du type :

$$(interrupteur\_on \wedge \neg anormal) \Rightarrow piece\_eclairee,$$

exprimant que "Si l'interrupteur est enclenché et que l'on ne se trouve pas dans une situation anormale, alors la pièce est éclairée".

2. Rappelons que, intuitivement, l'hypothèse du monde clos permet de supposer comme *faux* les faits positifs que nous ne connaissons pas comme étant *vrais* (voir [Reiter 1977] pour plus de détails).

Il est introduit ici un *prédicat d'anormalité*. Cela signifie que dans une situation normale, si l'interrupteur est enclenché, alors la pièce est éclairée. Par contre, si l'on apprend que l'on se trouve dans une situation exceptionnelle (panne de courant, ampoule cassée...) alors le prédicat d'anormalité bloquera la possibilité de déduire que la pièce est éclairée. Par exemple, on peut traduire le cas anormal de l'ampoule cassée comme suit :

$$\neg ampoule\_ok \Rightarrow anormal,$$

Ainsi, si l'on apprend que l'ampoule est cassée, alors il sera impossible de conclure que la pièce est éclairée.

Seulement, dans une situation normale, pour laquelle il doit être possible de conclure que la pièce est éclairée, il est nécessaire de montrer la négation du prédicat d'anormalité (à savoir $\neg anormal$ dans notre exemple). Intuitivement, *circonscrire* le prédicat d'anormalité reviendra à minimiser le critère d'anormalité. En d'autres termes, les modèles dans lesquels l'anormalité est fausse seront préférés aux autres.

Dans le cas le plus simple, le principe de la circonscription est de minimiser une relation (représentée par un prédicat P) au prix d'un effet sur une autre relation (représentée par un prédicat Q). On parle donc de circonscription du prédicat P en faisant varier le prédicat Q. Voyons cela plus dans le détail.

2.3.1 Théorie de la démonstration

La circonscription étend la théorie de la preuve de la logique classique par un schéma d'axiomes.

Définition 71. *Pour la circonscription, dans la théorie $\mathcal{T}[P, Q]$ du prédicat P en faisant varier le prédicat Q, le schéma, noté $CIRC(\mathcal{T}[P, Q])$, est :*

$$(\mathcal{T}[\Phi_P, \Phi_Q] \wedge (\forall x_1, \ldots, x_n(\Phi_P(x_1, \ldots, x_n) \Rightarrow P(x_1, \ldots, x_n))) \Rightarrow (\forall x_1, \ldots, x_n(P(x_1, \ldots, x_n) \Rightarrow \Phi_P(x_1, \ldots, x_n)))$$

où Φ_P et Φ_Q peuvent être remplacées par n'importe quelles formules (qui peuvent contenir des variables libres, notamment $x_1, \ldots, x_n$).

Exemple 18. *Soit la théorie $\mathcal{T}$ constituée des formules suivantes :*

1. $On(interrupteur) \wedge \neg Anormal(ampoule) \Rightarrow Eclairee(piece)$,
2. $On(interrupteur)$.

Le schéma pour la circonscription, dans $\mathcal{T}$, de Anormal en faisant varier Eclairee est :

$$((On(interrupteur) \wedge \neg\Phi_{Anormal}(ampoule) \Rightarrow \Phi_{Eclairee}(piece)) \wedge On(interrupteur) \wedge (\forall x\ (\Phi_{Anormal}(x) \Rightarrow Anormal(x))) \Rightarrow \forall x\ (Anormal(x) \Rightarrow \Phi_{Anormal}(x)))$$

Donc, une instance de ce schéma, est par exemple obtenue en posant que :

- $\Phi_{Anormal}(x)$ *vaut* $\neg(x = x)$,
- $\Phi_{Eclairee}(x)$ *vaut* $x = piece$.

C'est-à-dire, dans cette instance du schéma, $\mathcal{T}[\Phi_{Anormal}, \Phi_{Eclairee}]$ s'écrit :

$$(On(interrupteur) \wedge \neg(\neg ampoule = ampoule) \Rightarrow piece = piece) \wedge On(interrupteur).$$

Clairement, $\mathcal{T}[\Phi_{Anormal}, \Phi_{Eclairee}]$ est prouvable (logique classique avec égalité). Passons à la partie suivante, $\forall x_1, \ldots, x_n(\Phi_P(x_1, \ldots, x_n) \Rightarrow P(x_1, \ldots, x_n))$. Dans cette instance du schéma, cette partie-ci s'écrit :

$$\forall x\ (\neg(x = x) \Rightarrow Anormal(x)),$$

ce qui est facilement prouvable en logique classique avec égalité. Nous pouvons donc déduire la partie finale de cette instance du schéma de circonscription, c'est-à-dire :

$$\forall x\ (Anormal(x) \Rightarrow \neg(x = x)).$$

Ainsi, nous avons déduit :

$$\forall x\ \neg Anormal(x).$$

Il suffit alors d'employer cette formule avec $On(interrupteur) \wedge \neg Anormal(ampoule) \Rightarrow Eclairee(piece)$ et $On(interrupteur)$ pour inférer :

$$Eclairee(piece).$$

La méthode employée dans l'exemple correspond exactement à l'inférence par circonscription, ce qui donne la définition suivante.

Définition 72. *Une formule φ est dérivable par circonscription de P dans $\mathcal{T}[P, Q]$ en faisant varier Q ssi $\mathcal{T}[P, Q] \cup CIRC(\mathcal{T}[P, Q]) \models \varphi$.*

Autrement dit, φ est dérivable par circonscription de P dans $\mathcal{T}$ en faisant varier Q ssi il existe une preuve en logique classique (éventuellement avec égalité) de φ à partir de $\mathcal{T}$ et des instances du schéma de circonscription (de P dans $\mathcal{T}$ en faisant varier Q).

2.3.2 Théorie des modèles

Le principe de la circonscription étant de minimiser une relation (représentée par un prédicat P), la sémantique de la circonscription est une théorie de modèles minimaux. Pour définir ceux-ci, un ordre entre modèles est introduit. Il a pour but de caractériser les modèles qui minimisent P, ce seront donc les modèles minimaux.

Définition 73. *Soit P et Q deux prédicats. L'ordre $\leq_{P,Q}$ entre interprétations de la logique classique est défini comme suit. Pour toutes interprétations I et I' de la logique classique, $I \leq_{P,Q} I'$ ssi toutes les conditions ci-dessous sont satisfaites :*

1. *I et I' ont le même domaine D*
2. *Pour tout symbole de fonction f, $I(f) = I'(f)$ (c.-à.d., f est interprété de la même façon dans I et I')*
3. *Pour tout symbole de prédicat X autre que P et Q, $I(X) = I'(X)$*
4. *$I(P) \subseteq I'(P)$*

Exemple 19. *Soit un langage se limitant à une constante c et aux trois prédicats unaires P, Q et R. On considère I défini par :*

$$\begin{aligned} D_I &= \{e_1, e_2\} \\ I(c) &= e_1 \\ I(P) &= \{e_1\} \\ I(Q) &= \{e_1, e_2\} \\ I(R) &= \{e_2\} \end{aligned}$$

Autrement dit, le domaine de I comporte deux éléments, e_1 et e_2, et, I interprète la constante c par e_1. D'autre part, I satisfait $P(c)$ et $\forall x Q(x)$ mais ne satisfait pas $R(c)$.

On considère I' défini par :

$$\begin{aligned} D_{I'} &= \{e_1, e_2\} \\ I'(c) &= e_1 \\ I'(P) &= \{e_1, e_2\} \\ I'(Q) &= \{e_1\} \\ I'(R) &= \{e_2\} \end{aligned}$$

Alors, I et I' ont même domaine, interprètent c pareillement, interprètent R pareillement, tandis que l'interprétation de P dans I est un sous-ensemble de l'interprétation de P dans I'. Donc, $I \leq_{P,Q} I'$.

Attention, on ne compare que des interprétations qui sont en tout point identiques sauf en ce qui concerne P et Q (éventuellement).

Exemple 20. *On considère I'' défini par :*

$$\begin{aligned} D_{I''} &= \{e_1, e_2\} \\ I''(c) &= e_1 \\ I''(P) &= \{e_1\} \\ I''(Q) &= \{e_1, e_2\} \\ I''(R) &= \{e_1\} \end{aligned}$$

Alors, on n'a pas $I \leq_{P,Q} I''$ car I et I'' n'interprètent pas R pareillement.

Définition 74. *Soit P et Q deux prédicats. Soit $\mathcal{T}$ une théorie. Une interprétation I est un modèle minimal de $\mathcal{T}$ pour P en faisant varier Q ssi I est un modèle de $\mathcal{T}$ et il n'existe aucun modèle M de $\mathcal{T}$ tel que $M \leq_{P,Q} I$ et $I \not\leq_{P,Q} M$.*

Exemple 21. *Soit la théorie $\mathcal{T}$ composée des formules suivantes :*

1. $P(c)$,
2. $Q(c) \wedge \exists x R(x)$.

Alors I défini plus haut est un modèle minimal de $\mathcal{T}$ pour P en faisant varier Q. Car il n'est pas possible de trouver un modèle M de $\mathcal{T}$ tel que $M \leq_{P,Q} I$ et $I \not\leq_{P,Q} M$. En effet, I interprète c par e_1, donc pour avoir $M \leq_{P,Q} I$, il faudrait que M interprète aussi c par e_1, et, pour que M soit un modèle de $\mathcal{T}$, il faudrait $e_1 \in M(P)$, mais alors $I(P) \subseteq M(P)$ et donc $I \leq_{P,Q} M$ (en particulier, I et M sont en tout point identiques, sauf pour P et Q, du fait de $M \leq_{P,Q} I$).

Exemple 22. *Soit la théorie $\mathcal{T}$ composée des formules suivantes :*

1. $On(interrupteur) \wedge \neg Anormal(ampoule) \Rightarrow Eclairee(piece)$,
2. $On(interrupteur)$.

On considère J défini par :

$$\begin{aligned} D_J &= \{e_1, e_2, e_3\} \\ J(ampoule) &= e_2 \\ J(piece) &= e_3 \\ J(interrupteur) &= e_1 \\ J(Anormal) &= \emptyset \\ J(Eclairee) &= \{e_3\} \\ I(On) &= \{e_1\} \end{aligned}$$

On considère J' défini par :

$$\begin{aligned} D_{J'} &= \{e_1, e_2, e_3\} \\ J'(ampoule) &= e_2 \\ J'(piece) &= e_3 \\ J'(interrupteur) &= e_1 \\ J'(Anormal) &= \{e_2\} \\ J'(Eclairee) &= \emptyset \\ I'(On) &= \{e_1\} \end{aligned}$$

J et J' sont clairement des modèles de $\mathcal{T}$. De plus, $J \leq_{Anormal,Eclairee} J'$ et $J' \not\leq_{Anormal,Eclairee} J$. Donc, J' n'est pas un modèle minimal de $\mathcal{T}$ pour Anormal en faisant varier Eclairee. En revanche, J est un tel modèle minimal (on peut voir qu'il n'est pas possible de construire un modèle de $\mathcal{T}$ identique à J sauf pour Eclairee et pour $M(Anormal)$ non vide). On peut montrer que tout modèle minimal de $\mathcal{T}$ pour Anormal en faisant varier Eclairee satisfait $\forall x \neg Anormal(x)$ et $Eclairee(piece)$.

Tout modèle minimal de $\mathcal{T}$ pour P en faisant varier Q satisfait toutes les instances du schéma de circonscription (de P dans $\mathcal{T}$ en faisant varier Q). Ou, de manière équivalente, toute formule dérivable par circonscription (de P dans $\mathcal{T}[P, Q]$ en faisant varier Q) est satisfaite par tout modèle minimal (de $\mathcal{T}$ pour P en faisant varier Q). La réciproque est fausse (voir [Besnard *et al.* 1988]).

Chapitre 3

Changements de croyances

Sommaire

En raison de son caractère monotone (voir Table 1.3), nous avons pu voir que la logique propositionnelle est incapable de rétracter une conclusion précédemment établie lorsqu'une nouvelle information à intégrer la contredit. Au chapitre précédent, nous avons présenté des méthodes, appartenant à la famille des *logiques non monotones*, qui étendent la logique propositionnelle de manière à permettre un *raisonnement révisable*.

Dans cette optique, Gabbay et Makinson, plutôt que de proposer eux-aussi une autre de ces méthodes, ont proposé de considérer les méthodes existantes comme des relations d'inférence et d'étudier les propriétés logiques que l'on pouvait attendre des relations de conséquences logiques qui leur sont associées (voir [Gabbay 1985], [Makinson 1989] et [Makinson 1994] par exemple [1]). Cela a permis de comparer plus finement les différentes méthodes proposées et de définir des familles de méthodes ayant des propriétés particulières.

La *révision de croyances*, notion duale de celle de logique non monotone [2], pose ainsi le problème suivant. Étant donné une base de connaissances [3] représentant les *croyances* d'un agent à propos du monde et une nouvelle information apprise sur ce monde, potentiellement incohérente avec la base, quelles modifications doivent être opérées dans celle-ci pour incorporer la nouvelle information ? Le cadre AGM, défini par Alchourrón, Gärdenfors et Makinson (voir [Alchourrón *et al.* 1985], [Gärdenfors 1988]), propose une caractérisation logique

1. Notons que d'autres études des propriétés des systèmes d'inférence que sous tendent les logiques non monotones apparaissent dans la littérature (voir [Besnard 1988] et [Kraus *et al.* 1990] entre autres).

2. Selon Gärdenfors (voir [Gärdenfors 1990]), la révision de croyances et la logique non monotone sont deux faces d'une même pièce de monnaie ("Belief revision and nonmonotonic logic are two sides of the same coin").

3. Rappelons qu'une *base de connaissances*, est un ensemble de formules clos déductivement ($K = Cn(K)$).

des opérateurs de révision, c'est-à-dire un ensemble de propriétés logiques, appelés *postulats*, qu'un opérateur de révision "rationnel" doit satisfaire. Plus tard, Katsuno et Mendelzon (voir [Katsuno & Mendelzon 1991b]) ont proposé une formulation équivalente des postulats AGM lorsque les bases de connaissances sont exprimées dans un langage propositionnel fini.

Le cadre AGM ne traite pas uniquement du problème de la révision de croyances, d'autres changements de croyances comme le problème du retrait d'une information d'une base de connaissances, opération appelée *contraction*, sont abordés. De la même manière, Katsuno et Mendelzon se sont aussi intéressés à une autre opération de changement de croyances, appelée *mise à jour* (voir [Katsuno & Mendelzon 1991a]). Dans celle-ci, le monde représenté par les croyances de l'agent est considéré comme dynamique : la base de connaissances modélise une situation dans le passé et la nouvelle information concerne une situation dans le présent ; alors qu'en révision de croyances, le monde est considéré comme statique : la base de connaissances et la nouvelle information modélisent une même situation.

Dans ce chapitre, nous présentons brièvement ces différentes opérations de changement de croyances. Pour de plus amples détails sur la *théorie de la révision*, nous invitons le lecteur à se référer aux ouvrages suivants (voir [Konieczny 1999] et [Konieczny 2010]). Notons que bien que ces opérations de changement de croyances soient difficilement implémentable en pratique (voir [Nebel 1992] et [Nebel 1998]), différentes mises en œuvre ont été proposées dans la littérature (voir [Benferhat *et al.* 1999], [Delgrande *et al.* 2007] et [Benferhat *et al.* 2010] pour ne citer que certaines d'entre-elles).

Le chapitre est organisé comme suit. Dans la section 3.1, nous présentons le problème de la révision de croyances dans le cadre AGM puis dans le cadre propositionnel fini de Katsuno et Mendelzon. Le problème de la contraction, puis celui de la mise à jour, sont ensuite introduits respectivement dans la section 3.2 et 3.3.

3.1 Révision

La révision de croyance pose le problème de savoir quelles modifications doivent être opérées dans une base de connaissances pour y incorporer une nouvelle information. Alchourrón, Gärdenfors et Makinson proposent une caractérisation logique des opérateurs de révision de croyances, comme un ensemble de postulats de rationalité se basant sur les trois principes fondamentaux suivants :

1. *Principe de cohérence :* la base de connaissance résultant de l'opération de révision doit être cohérente.
2. *Principe de primauté de la nouvelle information :* la nouvelle information doit primer par rapport à la base de connaissance.
3. *Principe de changement minimal :* le changement effectué dans la base de connaissance afin d'incorporer la nouvelle information doit être minimal.

Les postulats AGM (voir [Alchourrón *et al.* 1985] et [Gärdenfors 1988]) font abstraction de la logique utilisée et ne restreignent pas à la logique propositionnelle. Dans cette section, nous ne présentons donc pas uniquement les postulats de révision du cadre AGM, mais nous présentons aussi ceux établis par Katsuno et Mendelzon dans le cadre propositionnel fini (voir [Katsuno & Mendelzon 1991b]).

3.1.1 Expansion

Avant toute chose, nous présentons l'opération d'*expansion* du cadre AGM. L'opération d'expansion + consiste à ajouter une information A à une base de connaissances K quand K et A sont compatibles. Aucune information ne doit donc être retirée de K lors de cette opération.

Définition 75 (opérateur d'expansion AGM [Alchourrón *et al.* 1985])**.** *Soient une base de connaissances K et une information A. L'expansion de K par A est notée $K + A$ et doit vérifier les postulats suivants :*

(K + 1)	*$K + A$ est une théorie*	*(clôture)*
(K + 2)	$A \in K + A$	*(succès)*
(K + 3)	$K \subseteq K + A$	*(inclusion)*
(K + 4)	*Si* $A \in K$, *alors* $K + A = K$	*(vacuité)*
(K + 5)	*Si* $K \subseteq H$, *alors* $K + A \subseteq H + A$	*(monotonie)*
(K + 6)	*$K + A$ est la plus petite base de connaissance satisfaisant* $(K + 1) - (K + 5)$	*(minimalité)*

$K + 1$ assure que le résultat de l'expansion est bien une théorie. $K + 2$ exprime que la nouvelle information doit appartenir à la théorie résultante. $K + 3$ certifie que l'on garde toutes les informations de l'ancienne base. $K+4$ dit que si l'information appartient déjà à la base alors la nouvelle base est identique à l'ancienne. $K + 5$ exprime le fait que la nouvelle information ne peut remettre en question les informations de la théorie. Enfin $K + 6$ exprime la minimalité du changement, il assure donc que la théorie résultante ne contient pas de connaissance non justifiée par l'ajout de la nouvelle information.

Le théorème suivant montre que seule l'union ensembliste est un opérateur satisfaisant ces postulats de rationalité. La base de connaissances résultant d'une opération d'expansion est donc l'ensemble des conséquences logiques de la conjonction de la base de connaissances et de la nouvelle information.

Théorème 2 ([Alchourrón *et al.* 1985])**.** *Une opération d'expansion + satisfait les postulats* $(K + 1) - (K + 6)$ *ssi* $K + A = Cn(K \cup A)$.

3.1.2 Révision AGM

L'opération d'expansion n'est pas adaptée quand il s'agit d'incorporer à une base de connaissances une information qui la contredit. L'opération de révision $*$ consiste donc à ajouter une information A à une base de connaissances K quand K et A ne sont pas compatibles. Il faut alors abandonner certaines connaissances pour maintenir la cohérence de la base ($*$ ne peux donc pas être monotone au sens de $K + 5$)

Définition 76 (opérateur de révision AGM [Alchourrón *et al.* 1985])**.** *Soient une base de connaissances K et une information A. La révision de K par A est notée $K * A$ et doit vérifier les postulats suivants :*

(**K** $*$ **1**)	$K * A$ *est une théorie*	*(clôture)*
(**K** $*$ **2**)	$A \in K * A$	*(succès)*
(**K** $*$ **3**)	$K * A \subseteq K + A$	*(inclusion)*
(**K** $*$ **4**)	*Si* $\neg A \notin K$, *alors* $K + A \subseteq K * A$	*(vacuité)*
(**K** $*$ **5**)	$K * A = K_{\perp}$ *ssi* $\vdash \neg A$	*(cohérence)*
(**K** $*$ **6**)	*Si* $A \equiv B$ *alors* $K * A = K * B$	*(extensionalité)*
(**K** $*$ **7**)	$K * (A \wedge B) \subseteq (K * A) + B$	*(inclusion conjonctive)*
(**K** $*$ **8**)	*Si* $\neg B \notin K * A$ *alors* $(K * A) + B \subseteq K * (A \wedge B)$	*(vacuité conjonctive)*

$K*1$ assure que le résultat de la révision est bien une théorie. $K*2$ exprime que la nouvelle information doit appartenir à la théorie résultante. $K + 3$ implique que la révision ne peut pas ajouter de connaissance qui ne soit pas une conséquence de la nouvelle information et de la base de connaissances. $K*3$ et $K*4$ expriment ensemble que lorsque la base de connaissances est cohérente avec la nouvelle information, alors la révision revient à une expansion [4]. $K * 5$ dit que le seul moyen de rendre une base contradictoire après la révision est de réviser par une information elle-même contradictoire. $K * 6$ dit que le résultat de la révision ne dépend pas de la syntaxe de la nouvelle information.

Ces six premiers postulats sont les postulats de base pour les opérateurs de révision. $K * 7$ et $K * 8$ sont des postulats supplémentaires qui expriment le bon comportement des opérateurs de révision en terme de minimalité du changement. Ils assurent que la révision par une conjonction de deux informations revient à une révision par la première suive d'une extension par la deuxième, dès que cela est possible (c.-à-d. quand la deuxième ne contredit aucune connaissance de la première révision).

3.1.3 Révision propositionnelle

Katsuno et Mendelzon ont proposé une formulation équivalente des postulats AGM s'adaptant au cadre propositionnel standard. Notons qu'une base de connaissances K est équivalente à la formule φ qui est la conjonction des formules de K.

Notons que dans ce cadre, la révision de φ par μ revient à rechercher les modèles de μ les plus proches de ceux de φ.

Définition 77 (opérateur de révision AGM [Katsuno & Mendelzon 1991b])**.** *Soient φ et μ deux formules propositionnelles où φ joue le rôle de la base de connaissances et μ celui de la nouvelle information. La révision de φ par μ est notée $\varphi \circ \mu$ et doit vérifier les postulats suivants :*

4. En accord avec le théorème 2, quand A est cohérent avec K, la révision se résume donc à une conjonction ou une union ensembliste entre K et A : c'est ce postulat que nous remettons en cause quand il s'agit de faire prédominer une information dans une théorie qui la subsume.

(R1)	$\varphi \circ \mu \vdash \mu$	*(succès)*
(R2)	*Si* $\varphi \wedge \mu$ *est cohérent alors* $\varphi \circ \mu \equiv \varphi \wedge \mu$	*(vacuité)*
(R3)	*Si* μ *est cohérent alors* $\varphi \circ \mu$ *est cohérent*	*(cohérence)*
(R4)	*Si* $\varphi_1 \equiv \varphi_2$ *et* $\mu_1 \equiv \mu_2$ *alors* $\varphi_1 \circ \mu_1 \equiv \varphi_2 \circ \mu_2$	*(extensionalité)*
(R5)	$(\varphi \circ \mu) \wedge \phi \vdash \varphi \circ (\mu \wedge \phi)$	*(inclusion conjonctive)*
(R6)	*Si* $(\varphi \circ \mu) \wedge \phi$ *est cohérent, alors* $\varphi \circ (\mu \wedge \phi) \vdash (\varphi \circ \mu) \wedge \phi$	*(vacuité conjonctive)*

Théorème 3 ([Katsuno & Mendelzon 1991b]). *Soit un opérateur de révision* $*$ *sur des théories et* $\circ$ *son opérateur correspondant*[5]. *Alors* $*$ *satisfait les postulats* $(K*1)-(K*8)$ *ssi* $\circ$ *vérifie les postulats* $(R1)-(R6)$.

Katsuno et Mendelzon ont aussi proposé un *théorème de représentation* qui permet de donner une définition constructive de leur opérateur $\circ$ de révision AGM du cadre propositionnel standard. Ce théorème correspond à une méthode assez intuitive de révision basée sur un préordre sur les mondes possibles, appelé *assignement fidèle*, qui exprime la révision comme une sélection de modèles minimaux de la nouvelle information suivant cette mesure de confiance sur les mondes.

Définition 78 (assignement fidèle [Katsuno & Mendelzon 1991a]). *Un assignement fidèle est une fonction qui associe à chaque base de connaissances* φ *un pré-ordre* $\leq_\varphi$ *sur les interprétations tel que :*

1. *Si* $I \models \varphi$ *et* $J \models \varphi$, *alors* $I \simeq_\varphi J$,
2. *Si* $I \models \varphi$ *et* $J \not\models \varphi$, *alors* $I <_\varphi J$,
3. *Si* $\varphi_1 \equiv \varphi_2$, *alors* $\leq_{\varphi_1} = \leq_{\varphi_2}$.

Cela revient à dire que les modèles de φ sont tous équivalents pour l'ordre associé et sont strictement préférés aux contre-modèles de φ. Plus une interprétation est petite pour l'ordre, plus elle est préférée. Lorsque l'on effectue une révision, ce sont alors les interprétation de la nouvelle information les plus crédibles pour la base courante qui forment les modèles de la nouvelle base de connaissances. Ceci est décrit formellement dans le théorème de représentation suivant.

Théorème 4 ([Katsuno & Mendelzon 1991b]). *Un opérateur de révision* $\circ$ *satisfait les postulats* $(R1)-(R6)$ *si et seulement si il existe un assignement fidèle qui associe à chaque base de connaissances* φ *un pré-ordre total* $\leq_\varphi$ *tel que* $Mod(\varphi \circ \mu) = min(Mod(\mu), \leq_\varphi)$.

Au delà des postulats précédents, plusieurs familles d'opérateurs ont été proposées dans la littérature, tant d'un point de vue syntaxique, donnant ainsi plus d'importance aux formules (voir [Nebel 1991], [Lehmann 1995] et [Wurbel *et al.* 2000] pour ne citer que quelques références), que d'un point de vue sémantique, reposant ainsi sur les interprétations (voir en particulier [Grove 1988], [Dalal 1988] et [Borgida 1985]).

5. Quand $Cn(\varphi) = K$, on dit que un opérateur $*$ correspond à un opérateur $\circ$ ssi $K * A = Cn(\varphi \circ A)$.

3.2 Contraction

À l'opposé des opérations d'extension et de révision qui permettent d'incorporer une nouvelle information dans une base de connaissances, l'opération de contraction, quant à elle, a pour objectif de rétracter une information d'une base de connaissances. Néanmoins, pour cette opération aussi la minimalité du changement est exigée : on ne veut supprimer de la base que ce qui est nécessaire pour ne plus impliquer l'information.

Dans cette section, au-delà de présenter les postulats de contraction du cadre AGM et ceux établis par Katsuno et Mendelzon dans le cadre propositionnel fini, nous présenterons une extension de la contraction de croyances du cadre AGM à la *contraction multiple*. La contraction multiple, définie par Fuhrmann et Hansson (voir [Fuhrmann & Hansson 1994]), permet de contracter une base de connaissances par un ensemble d'informations, cela de manière simultanée.

3.2.1 Contraction AGM

L'opération de contraction $-$ du cadre AGM consiste donc à retirer une information A d'une base de connaissances K.

Définition 79 (opérateur de contraction AGM [Alchourrón *et al.* 1985])**.** *Soient une base de connaissances K et une information A. La contraction de K par A est notée $K \div A$ et doit vérifier les postulats suivants :*

(K ÷ 1)	$K \div A$ *est une théorie*	*(clôture)*
(K ÷ 2)	$K \div A \subseteq K$	*(inclusion)*
(K ÷ 3)	*Si* $A \notin K$, *alors* $K \div A = K$	*(vacuité)*
(K ÷ 4)	*Si* $\nvdash A$, *alors* $A \notin K \div A$	*(succès)*
(K ÷ 5)	*Si* $A \in K$, *alors* $K \subseteq (K \div A) + A$	*(restauration)*
(K ÷ 6)	*Si* $A \equiv B$, *alors* $K \div A = K \div B$	*(préservation)*
(K ÷ 7)	$(K \div A) \cap (K \div B) \subseteq K \div (A \wedge B)$	*(intersection)*
(K ÷ 8)	*Si* $A \notin K \div (A \wedge B)$, *alors* $K \div (A \wedge B) \subseteq K \div A$	*(conjonction)*

$K \div 1$ assure que le résultat de la contraction soit bien une théorie. $K \div 2$ exprime que lors de l'opération de contraction aucune information ne doit être ajoutée. $K \div 3$ permet de montrer que si A n'appartient pas à K, alors il n'y a pas lieu d'apporter le moindre changement à la base. $K \div 4$, quant à lui, garantit que la contraction réussira à condition que A ne soit pas une tautologie. $K \div 5$ est un postulat très important bien que critiqué (voir [Makinson 1987] et [Hansson 1991]) qui affirme que l'on doit retrouver K si l'on contracte K par A et que l'on ajoute A suite à cette opération, ce qui voudrait dire que l'opération de contraction ne retire rien d'autre que A. $K \div 6$ exprime que le résultat de la contraction ne dépend pas de la syntaxe des formules.

Ces six premiers postulats sont les postulats de base pour les opérateurs de contraction. $K \div 7$ et $K \div 8$ sont des postulats supplémentaires qui expriment la minimalité du changement pour la conjonction.

Quand l'opération d'expansion permet d'ajouter une information à une base de connaissances, l'opération de contraction permet elle de la retirer. L'opération de révision, laquelle a pour but de modifier une base de connaissances en ajoutant et en retirant des informations, pourrait donc être exprimée à partir des opérateurs d'expansion et de contraction. À partir de ce raisonnement, Gärdenfors a mis défini une correspondance entre ces différentes opérations de changement de croyances. Ainsi, si l'on dispose d'un opérateur de contraction et d'un opérateur d'expansion, il est possible de définir un opérateur de révision (voir [Gärdenfors 1988]).

Propriété 3 (identité de Lévi [Gärdenfors 1988])**.** $K * A = (K \div \neg A) + A$

Le théorème suivant montre que les opérateurs définis grâce à l'identité de Lévi sont bien des opérateurs de révision.

Théorème 5 ([Gärdenfors 1988])**.** *Si l'opérateur de contraction* $\div$ *satisfait* $(K-1)-(K-4)$ *et* $(K-6)$ *et l'opérateur d'expansion* $+$ *satisfait* $(K+1)-(K+6)$*, alors l'opérateur de révision* $*$ *défini par l'identité de Lévi satisfait* $(K*1)-(K*6)$*. De plus, si* $(K \div 7)$ *est satisfait, alors* $(K*7)$ *l'est aussi, et, si* $(K \div 8)$ *est satisfait, alors* $(K*8)$ *est satisfait.*

L'identité de Harper exprime que la correspondance inverse est vérifiée. Ainsi si l'on dispose d'un opérateur de révision, il est possible de définir un opérateur de contraction.

Propriété 4 (identité de Harper [Gärdenfors 1988])**.** $K \div A = K \cap (K * \neg A)$

On remarque alors que les informations issues de la contraction de K par A sont celles n'ayant rien à voir avec la véracité de A. Le théorème suivant montre que les opérateurs de contraction définis à l'aide de l'identité de Harper sont de bons opérateurs de contraction.

Théorème 6 ([Gärdenfors 1988])**.** *Si l'opérateur de révision* $*$ *satisfait* $(K*1)-(K*6)$*, alors l'opérateur de contraction* $\div$ *définie par l'identité de Harper satisfait* $(K \div 1)-(K \div 6)$*. De plus si* $(K*7)$ *est satisfait alors* $(K \div 7)$ *l'est aussi, et, si* $(K*8)$ *est satisfait, alors* $(K \div 8)$ *l'est aussi.*

3.2.2 Contraction propositionnelle

Pour l'opération de contraction aussi, Katsuno et Mendelzon ont proposé une formulation équivalente des postulats AGM lorsque les bases de connaissances sont exprimées dans un langage propositionnel fini.

Définition 80 (opérateur de contraction AGM [Katsuno & Mendelzon 1991b])**.** *Soient* φ *et* μ *deux formules propositionnelles où* φ *joue le rôle de la base de connaissances et* μ *celui de la nouvelle information. La contraction de* φ *par* μ *est notée* $\varphi \bullet \mu$ *et doit vérifier les postulats suivants :*

(C1)	$\varphi \vdash \varphi \bullet \mu$	*(inclusion)*
(C2)	*Si* $\varphi \nvdash \mu$ *alors* $\varphi \bullet \mu \equiv \varphi$	*(vacuité)*
(C3)	*Si* μ *n'est pas tautologique alors* $\varphi \bullet \mu \nvdash \mu$	*(succès)*
(C4)	*Si* $\varphi_1 \equiv \varphi_2$ *et que* $\mu_1 \equiv \mu_2$ *alors* $\varphi_1 \bullet \mu_1 \equiv \varphi_2 \bullet \mu_2$	*(préservation)*
(C5)	$(\varphi \bullet \mu) \wedge \mu \vdash \varphi$	*(restauration)*

3.2.3 Contraction multiple

Définie par Fuhrmann et Hansson (voir [Fuhrmann & Hansson 1994]), la contraction multiple est une extension de la contraction de croyances du cadre AGM. Celle-ci permet de contracter une base de connaissances K, non pas juste par une seule information comme dans le cadre AGM, mais par un ensemble d'informations Λ de manière à ce qu'aucune information de Λ ne puisse être inférée à partir de l'ensemble de connaissances résultant, lequel est donc un sous-ensemble de (la fermeture déductive de) Γ.

Il est important de noter que cette approche est totalement différente des approches de contraction "répétée" ou "itérée", lesquelles consistent en l'exécution de plusieurs contractions d'une seule information à la suite. En effet, la contraction multiple est une opération unique de *contraction simultanée* par un ensemble d'informations.

Définition 81 (opérateur de contraction AGM [Fuhrmann & Hansson 1994])**.** *Soient K une base de connaissances et Λ un ensemble d'informations. La contraction de K par Λ est notée $K \ominus \Lambda$ et doit vérifier les postulats suivants :*

(M1)	$K \ominus \Lambda \subseteq K$	*(inclusion)*
(M2)	*Si* $\Lambda \cap Cn(\emptyset) = \emptyset$ *alors* $\Lambda \cap (K \ominus \Lambda) = \emptyset$	*(succès)*
(M3)	*Si* $\Lambda \equiv \Theta$ *alors* $K \ominus \Lambda = K \ominus \Theta$	*(extensionalité)*
(M4)	*Si* $\varphi \in K \ominus \Lambda$ *alors* $K \ominus \Lambda \subseteq \Omega \subseteq K$ *pour tout* Ω *t.q.* $\Lambda \cap \Omega = \emptyset$ *et* $\Lambda \cap Cn(\Omega \cup \{\varphi\}) \neq \emptyset$	*(pertinence)*

$(M1)$ indique que lors de l'opération de contraction multiple, aucune information ne doit être ajoutée. $(M2)$ indique que si Λ contient uniquement des théorèmes alors Λ ne peut pas être contracté de K. $(M3)$, quant à lui, exprime que si deux ensembles d'informations sont équivalents alors la contraction de K par l'un ou l'autre mène au même résultat. Enfin, $(M4)$ exprime que toute information retirer de K ne doit pas l'être sans raison.

Fuhrmann et Hansson, au-delà de fournir une caractérisation logique de l'opération de contraction multiple, fournissent aussi un théorème de représentation lequel construit un opérateur de contraction multiple comme l'intersection des plus grands sous-ensembles de K qui n'impliquent aucune information de Λ.

Définition 82 (ensemble $K \perp \Lambda$ [Fuhrmann & Hansson 1994])**.** *Soient K une base de connaissances et Λ un ensemble d'informations. L'ensemble des plus grands sous-ensembles de K qui n'impliquent aucune information de Λ, noté $K \perp \Lambda$ est défini de manière à ce que :*

$$\Phi \in K \perp \Lambda \quad ssi \quad \begin{cases} \Phi \subseteq K, \\ \Phi \cap \Lambda = \emptyset, \\ K \cap Cn(\Omega) \neq \emptyset \text{ pour tout } \Omega \text{ t.q. } \Phi \subset \Omega \subseteq K. \end{cases}$$

Définition 83 (fonction de sélection μ [Fuhrmann & Hansson 1994])**.** *Soient K une base de connaissances, Λ un ensemble d'informations et $K \perp \Lambda$ l'ensemble des plus grands sous-ensembles de K qui n'impliquent aucune information de Λ. Une fonction de sélection μ dans $K \perp \Lambda$*

correspond à toute fonction satisfaisant les deux conditions ci-dessous :

- $\emptyset \neq \mu(K \perp \Lambda) \subseteq K \perp \Lambda$ *si* $K \perp \Lambda \neq \emptyset$
- $\mu(K \perp \Lambda) = K$ *si* $K \perp \Lambda = \emptyset$

Ainsi, à condition de déterminer une fonction de sélection μ dans $K \perp \Lambda$, un opérateur de contraction multiple peut donc être construit.

Théorème 7 ([Fuhrmann & Hansson 1994])**.** *Un opérateur de contraction multiple* $\ominus$ *satisfait* $(M1) - (M4)$ *ssi il existe une fonction de sélection* μ *dans* $K \perp \Lambda$ *t.q. :*

$$K \ominus \Lambda = \bigcap \mu(K \perp \Lambda).$$

3.3 Mise à jour

Katsuno et Mendelzon (voir [Katsuno & Mendelzon 1991a]), précédés par Keller et Winslett (voir [Keller & Winslett 1985]), ont mis en exergue le fait que les postulats AGM ne s'appliquaient pas à tous les types de changements de la connaissance. En effet, les opérateurs de révision permettent d'incorporer une connaissance à propos du monde et ainsi d'*améliorer* sa connaissance du monde. Mais ces opérateurs ne sont pas adaptés quand il s'agit de reporter un changement apporté au monde permettant ainsi d'*actualiser* sa connaissance du monde. Alors que l'opération de révision permet de travailler sur un monde statique (l'état du monde ne change pas, c'est notre connaissance à propos de ce monde qui change), l'opération de mise à jour caractérisée par Katsuno et Mendelzon permet de travailler sur un monde dynamique (l'état du monde change et ces changements sont répercutés sur notre connaissance du monde).

Les méthodes de révision qui satisfont les postulats AGM sont exactement celles qui sélectionnent à partir des modèles de μ ceux qui sont les plus "proches" des modèles de φ. Les modèles sélectionnés déterminent donc la nouvelle théorie. D'un autre côté, les méthodes de mise à jour sélectionnent, pour chaque modèle M de la base φ, l'ensemble des modèles de μ qui sont "proches" de M. La nouvelle théorie correspond donc à l'union de ces modèles.

Définition 84 (opérateur de mise à jour [Katsuno & Mendelzon 1991a])**.** *Soient* φ *et* μ *deux formules propositionnelles où* φ *joue le rôle de la base de connaissances et* μ *celui de la nouvelle information. La mise à jour de* φ *par* μ *est notée* $\varphi \diamond \mu$ *et doit vérifier les postulats suivants :*

(U1)	$\varphi \diamond \mu \vdash \mu$	*(clôture)*
(U2)	*Si* $\varphi \vdash \mu$*, alors* $\varphi \diamond \mu \equiv \varphi$	*(succès faible)*
(U3)	*Si* φ *et* μ *sont cohérents alors* $\varphi \diamond \mu$ *est cohérent*	*(inclusion faible)*
(U4)	*Si* $\varphi_1 \equiv \varphi_2$ *et* $\mu_1 \equiv \mu_2$*, alors* $\varphi_1 \diamond \mu_1 \equiv \varphi_2 \diamond \mu_2$	*(vacuité)*
(U5)	$(\varphi \diamond \mu) \wedge \phi \vdash \varphi \diamond (\mu \wedge \phi)$	*(cohérence)*
(U6)	*Si* $\varphi \diamond \mu_1 \vdash \mu_2$ *et* $\varphi \diamond \mu_2 \vdash \mu_1$*, alors* $\varphi \diamond \mu_1 \equiv \varphi \diamond \mu_2$	
(U7)	*Si* φ *est une formule complète, alors* $(\varphi \diamond \mu_1) \wedge (\varphi \diamond \mu_2) \vdash \varphi \diamond (\mu_1 \vee \mu_2)$	
(U8)	$(\varphi_1 \vee \varphi_2) \diamond \mu \equiv (\varphi_1 \diamond \mu) \vee (\varphi_2 \diamond \mu)$	

$(U1) - (U5)$ correspondent aux postulats $(R1) - (R5)$ de la révision. $(U2)$ et $(U3)$ sont respectivement des versions affaiblies des postulats $(R2)$ et $(R3)$. $(U6) - (U8)$ remplacent le postulat $(R6)$. $(U6)$ dit que, si la mise à jour d'une base par une nouvelle information μ_1 implique μ_2 et que la réciproque est vérifiée (la mise à jour par μ_2 implique μ_1), alors les deux mises à jours doivent donner le même résultat. $(U7)$ exprime le fait que lorsque l'on a une connaissance complète du monde, les modèles communs aux deux mises à jour $\varphi \diamond \mu_1$ et $\varphi \diamond \mu_2$ sont également modèles de $\varphi \diamond (\mu_1 \vee \mu_2)$. Le postulat central, $(U8)$, assure que l'on examine séparément chaque monde possible (modèle) de la base de connaissances.

3.3.1 Effacement

Mis au point par Katsuno et Mendelzon (voir [Katsuno & Mendelzon 1991a]), ces opérateurs sont aux opérateurs de mise à jour ce que les opérateurs de contraction sont aux opérateurs de révision. Intuitivement, effacer par μ signifie que le monde doit avoir changé de telle sorte que μ soit faux. Au contraire, contracter par μ signifie que notre description de l'ensemble des mondes possibles doit être ajustée au fait que μ ne soit pas vrai.

Définition 85 (opérateur d'effacement [Katsuno & Mendelzon 1991a])**.** *Soient φ et μ deux formules propositionnelles où φ joue le rôle de la base de connaissances et μ celui de la nouvelle information. L'effacement de φ par μ est noté $\varphi \triangleleft \mu$ et doit vérifier les postulats suivants :*

(E1)	$\varphi \vdash \varphi \triangleleft \mu$	*(inclusion)*
(E2)	*Si* $\varphi \vdash \neg\mu$ *alors* $\varphi \triangleleft \mu \equiv \varphi$	*(vacuité faible)*
(E3)	*Si* φ *est cohérent et que* μ *n'est pas tautologique alors* $\varphi \triangleleft \mu \nvdash \mu$	*(succès faible)*
(E4)	*Si* $\varphi_1 \equiv \varphi_2$ *et que* $\mu_1 \equiv \mu_2$ *alors* $\varphi_1 \triangleleft \mu_1 \equiv \varphi_2 \triangleleft \mu_2$	*(préservation)*
(E5)	$(\varphi \triangleleft \mu) \wedge \mu \vdash \varphi$	*(restauration)*
(E8)	$(\varphi_1 \vee \varphi_2) \triangleleft \mu \equiv (\varphi_1 \triangleleft \mu) \vee (\varphi_2 \triangleleft \mu)$	*(règle disjonctive)*

Il y a deux différences entre la contraction et l'effacement en terme de postulats. La première est que $(E2)$ est plus faible que $(R2)$; puisque la contraction de μ ne doit pas changer φ si $\varphi \nvdash \mu$, mais l'effacement de μ doit modifier φ si $\varphi \nvdash \neg\mu$. La deuxième est que l'effacement a besoin de la règle disjonctive $(E8)$, alors que la contraction non.

3.3.2 Oubli ou effacement symétrique

Une autre opération, appelée *oubli*, laquelle paraît être plus naturelle que l'effacement, a été définie par Winslett (voir [Winslett 1989]). L'opérateur d'*oubli* en résultant est mis en comparaison avec l'opérateur de contraction défini précédemment. Il en résulte que l'opérateur d'oubli, muni d'un opérateur $\diamond$ de mise à jour, est équivalent à :

$$(\varphi \diamond \mu) \vee (\varphi \diamond \neg\mu).$$

Katsuno et Mendelzon (voir [Katsuno & Mendelzon 1991a]) appellent cet opérateur *effacement symétrique* parce que μ et sa négation jouent le même rôle dans sa définition. La

principale différence entre l'effacement et l'effacement symétrique est que l'effacement n'affecte pas les mondes possibles dans lesquels $\neg\mu$ tient, mais l'effacement symétrique oui.

Travaux faisant référence à la notion d'oubli

Cet opérateur n'est pas à confondre avec la *théorie logique de l'oubli* définie par Lin et Reiter (voir [Lin & Reiter 1994]) dans le cadre de la robotique cognitive. Dans cette théorie, un robot possède une base de connaissances représentant l'état du monde initial qu'il doit "mettre à jour" après avoir exécuté des actions. Pour ce faire, il doit tout "oublier" concernant les connaissances qui ne sont plus *vraies* suite à ces actions, de manière à ce que cela n'affecte aucune des potentielles actions futures.

Une autre confusion doit être évitée. Elle concerne le cadre de travail défini par Lang et Marquis (voir [Lang & Marquis 2002]) pour le raisonnement à partir de bases de connaissances propositionnelles incohérentes. Dans ce cadre, l'*oubli de variables* est utilisé comme une opération de base permettant d'affaiblir des connaissances afin de retrouver la cohérence.

Deuxième partie

Contributions

Chapitre 4

Prédominance dans le cadre propositionnel standard

Sommaire

Lorsqu'une information peut déjà être inférée à partir d'un ensemble de connaissances dans lequel elle doit s'insérer, comment faire en sorte qu'elle vienne préempter les informations qui permettent son inférence ? Dans ce chapitre, nous proposons de répondre à cette question fondamentale de la représentation de la connaissance et du raisonnement, en nous plaçant dans le cadre très étudié de la logique propositionnelle.

Reprenons notre exemple canonique dans lequel nous supposons qu'un ensemble de connaissances contienne une règle f qui exprime que "Si l'interrupteur est enclenché alors la pièce est éclairée". Dans cette situation, il est naturel d'espérer que l'ajout d'une règle additionnelle g exprimant que "Si l'interrupteur est enclenché et si l'ampoule n'est pas cassée alors la pièce est éclairée" puisse venir préempter f au sens où il ne doit plus suffire de savoir que l'interrupteur est enclenché pour en conclure que la pièce est éclairée ; il faut aussi que l'ampoule ne soit pas cassée. g n'est pourtant pas en mesure de préempter f par elle-même et ne peut donc pas espérer pouvoir l'inhiber sans qu'une intervention dans l'ensemble de connaissances ne soit effectuée avant l'ajout de g.

Ce problème peut sembler à première vue trivial. Pour le résoudre, nous pourrions simplement retirer f et toute autre règle étant logiquement "plus fortes" que g de l'ensemble

de connaissance, et, "casser" les cheminements de raisonnement (impliquant potentiellement plusieurs règles) permettant de déduire g. Seulement, cela n'est pas suffisant puisqu'en insérant ensuite g dans l'ensemble de connaissances ainsi transformé, g pourrait être en mesure de "restaurer" des cheminements de raisonnement que nous viendrions à peine de retirer ou même en créer de nouveaux. Une solution plus fine doit être envisagée.

Le chapitre est organisé comme suit. Après avoir, dans la section 4.1, défini de manière formelle notre problème et montré que la solution triviale discutée n'est pas adaptée, nous présentons de manière progressive notre solution, certaines de ses propriétés les plus intéressantes et une contre-partie sémantique dans la section 4.2. Dans la section 4.3, nous étudions notre solution d'un point de vue algorithmique et la soumettons à différentes expérimentations. La section 4.4, quant à elle, met en exergue les liens pouvant exister entre notre travail et des travaux de la littérature. Pour finir, nous discutons de certaines perspectives intéressantes de travaux futurs que ce chapitre laissent entrevoir, dans la section 4.5.

Une grande partie des travaux de ce chapitre ont fait l'objet d'une communication en langue anglaise (voir [Besnard *et al.* 2011a]).

4.1 Caractérisation logique

Dans la suite de ce chapitre, nous viserons à établir des résultats qui s'expriment indépendamment de la syntaxe initiale des formules et qui s'appuient sur la pleine capacité déductive de la logique ; les rôles d'*antécédent* et de *conséquent* d'une règle exprimée par une formule de la logique propositionnelle standard pourront apparaître arbitraires sous cet angle. En particulier, nous prendrons en compte le *modus tollens* qui intervertit, modulo un changement de signes, l'antécédent et le conséquent d'une règle. Par exemple, la règle "Si l'interrupteur est enclenché et si l'ampoule n'est pas cassée alors la pièce est éclairée" pourra se traduire de manière indifférenciée notamment par les formules suivantes :

- $(interrupteur\_on \wedge ampoule\_ok) \Rightarrow piece\_eclairee$,
- $interrupteur\_on \Rightarrow (piece\_eclairee \vee \neg ampoule\_ok)$,
- $(interrupteur\_on \wedge \neg piece\_eclairee) \Rightarrow \neg ampoule\_ok$,
- $\neg piece\_eclairee \Rightarrow (\neg interrupteur\_on \vee \neg ampoule\_ok)$,
- $ampoule\_ok \Rightarrow (piece\_eclairee \vee \neg interrupteur\_on)$,
- ...
- ou encore $\neg interrupteur\_on \vee \neg ampoule\_ok \vee piece\_eclairee$.

Dans la suite, nous privilégierons la dernière représentation, clausale, des règles.

4.1.1 Définition du problème

Soit une formule g représentant une règle que l'on veut voir préempter lors de son insertion dans un ensemble Γ de formules, au sens où toute capacité à déduire une règle f proche mais "plus forte" que g doit disparaître. Dans l'exemple introductif, g est un encodage de la règle "Si l'interrupteur est enclenché et si l'ampoule n'est pas cassée alors la pièce est éclairée" tandis qu'un f spécifique est un encodage de la règle "Si l'interrupteur est enclenché alors la pièce est éclairée".

Dans la suite nous supposerons que f et g sont des clauses qui ne sont ni tautologiques ni incohérentes et que Γ et Γ' sont deux ensembles de formules.

Techniquement, le problème que nous étudions consiste donc à transformer Γ en Γ' tel que Γ' permette de déduire g sans toutefois permettre de déduire une clause f qui soit plus forte que g au sens où f serait un impliquant strict de g (en abrégé, "Γ' ne subsume pas g").

Clairement, dans le cadre clausal, f est un impliquant strict de g ssi f est une sous-clause stricte de g (voir Chapitre 1, Section 1.2 pour plus de détails sur ces notions d'impliquants dans le cadre propositionnel standard).

Définition 86 (subsomption)**.** *Soient Γ' un ensemble de formules, f et g deux clauses ni tautologiques ni incohérentes. Γ' subsume g ssi $\Gamma' \models f$ pour au moins un impliquant strict f de g.*

4.1.2 Considérations épistémologiques

Avant d'aller plus loin, il est nécessaire de prendre certaines décisions d'ordre épistémologiques à propos du rôle que doit jouer Γ. Nous nous attendons ici à ce que toute formule de Γ puisse être potentiellement exclue lors du processus de transformation menant à Γ'. Particulièrement, quand nous avons besoin de distinguer les faits qui, par exemple, représentent des observations incontestables (par exemple le fait "La pièce est éclairée et l'interrupteur est enclenché") ou encore des circonstances additionnelles (par exemple le fait "L'interrupteur est enclenché" permettant d'appliquer une règle), ces faits ne doivent pas être inclus dans l'ensemble Γ sur le point d'être transformé. Pour illustrer cela, considérons les exemples suivants.

Exemple 23. *Soient g une clause et Γ un ensemble constitué des trois clauses suivantes :*

1. *interrupteur_on,*
2. *ampoule_ok,*
3. *g = $\neg$interrupteur_on $\vee$ $\neg$ampoule_ok $\vee$ piece_eclairee.*

Clairement, g représente la règle (interrupteur_on $\wedge$ ampoule_ok) $\Rightarrow$ piece_eclairee : supposons que nous voulons la voir préempter dans Γ. Ici, g se trouve donc déjà dans Γ. Notons que piece_eclairee est un impliquant strict de g. Γ' ne doit pouvoir contenir à la fois interrupteur_on et ampoule_ok car sinon $\Gamma' \models$ piece_eclairee et donc $\Gamma' \models$ interrupteur_on $\Rightarrow$ piece_eclairee, signifiant que g serait subsumé par Γ'.

Dans cet exemple, on pourrait souhaiter laisser intact Γ lors du processus de transformation et fournir en résultat $\Gamma' = \Gamma$. Cependant, de manière plus générale on pourrait vouloir exclure du processus de transformation les "faits" qui ne représenteraient que des circonstances particulières et contextuelles d'application des règles. Selon cette vision des choses, empêcher qu'une règle soit subsumée devrait se concevoir en toute généralité, sans prendre en compte d'éventuelles circonstances factuelles supplémentaires.

Sans cette mise à l'écart des faits, on ne peut empêcher la pleine application de la propriété d'*introduction de la disjonction* de la logique classique qui, dès lors qu'un fait sous la forme d'une clause unitaire d peut être déduit, entraîne que toute clause cohérente contenant d peut également l'être. Ainsi, si g est une clause que l'on ne veut pas voir subsumée par Γ', Γ' ne

doit pas être en mesure de contenir d, ou, de manière plus générale, un quelconque impliquant strict de g contenant d.

Considérons un second exemple où la présence d'un fait dans Γ permet aussi de déduire une règle qui devrait être exclue par g.

Exemple 24. *Soient g une clause et Γ un ensemble constitué des deux clauses suivantes :*

1. $\neg$*interrupteur_on,*
2. $g = \neg$*interrupteur_on* $\vee \neg$*ampoule_ok* $\vee$ *piece_eclairee.*

g représente, ici aussi, la règle (*interrupteur_on* $\wedge$ *ampoule_ok*) $\Rightarrow$ *piece_eclairee que nous voulons voir préempter dans Γ. Γ' ne pourra pas contenir $\neg$interrupteur_on, dans le cas contraire $\Gamma' \models f = \neg$interrupteur_on $\vee$ piece_eclairee, f représentant la règle interrupteur-_on $\Rightarrow$ piece_eclairee subsumant g.*

D'une certaine manière, le fait $\neg$*interrupteur_on* précise que nous ne nous trouvons pas dans les conditions d'application de la règle (*interrupteur_on* $\wedge$ *ampoule_ok*) $\Rightarrow$ *piece_eclairee*. Il semblerait tout à fait légitime de préserver cette information de contexte lorsque nous empêchons g d'être subsumé. Ceci renforce l'idée que, dans certaines applications, la transformation qui conduit une règle à ne pas pouvoir être subsumée doit être conduite en faisant abstraction de faits contextuels particuliers et en examinant uniquement les règles. Aussi, comme dans l'exemple précédent, dans certaines applications, on peut donc souhaiter que notre processus de transformation ne prenne pas en considération les "faits" de Γ.

Ce type de situations est classique dans nombre de problèmes et de techniques d'intelligence artificielle, où l'on sépare très clairement une *base de règles* d'une *base de faits* contextuelle ou de circonstance.

Bien évidemment, il faut alors préciser la nature logique de ces faits qui ne doivent pas être pris en considération lorsque l'on transforme Γ en Γ'. Sont-ce les simples clauses unitaires présentes explicitement dans Γ ? Ces faits peuvent-ils être des disjonctions du type *interrupteur_on* $\vee$ *ampoule_ok* ? Une réponse positive à cette dernière interrogation conduirait alors à différencier deux types de règles : celles qui sont soumises à la technique de transformation de ce chapitre et celles qui ne le sont pas. En effet, une disjonction peut être réécrite de manière équivalente sous forme implicative, donc de règles, qui ne se réduisent pas à une clause unitaire. Une autre question que l'on doit se poser est la suivante. Faut-il non seulement préserver les faits explicitement présents dans Γ mais aussi les faits qui se trouvent dans la fermeture déductive de Γ ? Voyons cela dans l'exemple qui suit.

Exemple 25. *Soient g une clause et Γ un ensemble constitué des cinq clauses suivantes :*

1. $g = \neg$*interrupteur_on* $\vee \neg$*ampoule_ok* $\vee$ *piece_eclairee,*
2. $\neg x \vee$ *interrupteur_on,*
3. $x \vee$ *interrupteur_on,*
4. $y \vee$ *ampoule_ok,*
5. $\neg y \vee$ *ampoule_ok.*

g représente là aussi la règle (*interrupteur_on* $\wedge$ *ampoule_ok*) $\Rightarrow$ *piece_eclairee que nous voulons voir préempter dans Γ. Clairement, $\Gamma \models$ interrupteur_on et $\Gamma \models$ ampoule_ok. Γ*

ne doit pas pouvoir permettre de déduire à la fois interrupteur_on et ampoule_ok, sinon $\Gamma \models$ *piece_eclairee et donc* $\Gamma \models \neg$*interrupteur_on* $\vee$ *piece_eclairee,* Γ' *subsumerait donc g.*

L'exemple précédent nous montre que les faits qui ne se trouvent pas explicitement dans Γ mais qui peuvent en être déduits peuvent, dans certains contextes, jouer un rôle différent de celui joué par les faits explicitement présents dans Γ. Il peut alors sembler justifié, lors de la transformation de Γ qui conduit à empêcher g d'être subsumé, de ne pas nécessairement préserver les règles qui conduisent à pouvoir déduire ces faits.

Ainsi, en accord avec ces considérations épistémologiques, nous considérons, par la suite, que toute formule de Γ peut être exclue lors du processus de transformation de Γ en Γ' afin de permettre à g de ne pas être subsumée par Γ'.

4.1.3 Opérateur $\ominus$ de contraction

Considérons maintenant une famille d'opérateurs binaires de contraction $\ominus$ qui à partir d'un ensemble cohérent de formules Γ et d'une formule h non tautologique produisent en résultat un ensemble de formules répondant aux contraintes suivantes.

Définition 87 (opérateur de contraction $\ominus$)**.** *Soit* Γ *un ensemble cohérent de formules et une formule h non tautologique. On appelle opérateur de contraction tout opérateur* $\ominus$ *qui à partir de* Γ *et de h produit un ensemble* $\Gamma \ominus h$ *de formules tel que :*

1. $\Gamma \ominus h \not\models h$,
2. $\Gamma \ominus h \subseteq Cn(\Gamma)$,
3. $\Gamma \ominus h = Cn(\Gamma \ominus h)$.

La première condition impose que la formule au sujet de laquelle Γ doit être contracté ne soit pas une conséquence logique de l'ensemble résultat. Cette condition impose de plus que le résultat de toute contraction soit cohérent, principe de base connu sous le nom de *principe de cohérence* des approches de révision de croyances (voir [Alchourrón *et al.* 1985], [Gärdenfors 1988]) et de mise à jour ([Katsuno & Mendelzon 1991a]). La seconde condition impose que le résultat d'une contraction soit un sous-ensemble des conséquences déductives de Γ. En ce sens, une contraction ne peut donc pas ajouter de nouvelles informations. La dernière condition, quant à elle, impose que le résultat d'une contraction soit un ensemble fermé déductivement.

Différents opérateurs concrets de contraction peuvent ainsi être envisagés. Par exemple, un opérateur pourrait produire en résultat le singleton $\{\top\}$, ce qui serait bien évidemment une approche extrêmement radicale et manifestement excessive. Bien qu'il ne soit pas nécessaire d'appliquer des restrictions supplémentaires à $\ominus$, de manière générale, des opérateurs plus acceptables se devront de respecter un autre principe fondamental des approches en révision et mise à jour, à savoir le *principe de changement minimal*, afin d'altérer Γ le moins possible et donc d'*oublier* le moins d'information possible. Les résultats obtenus dans ce chapitre ne requièrent pas pareilles conditions de minimalité et portent donc sur des familles plus larges d'opérateurs de contraction.

Lorsque Γ est cohérent, une approche naturelle mais incorrecte pour transformer Γ en Γ' tel que Γ' permette de déduire g mais pas de subsumer g consisterait à contracter Γ par rapport

à tous les impliquants stricts f de g et à ensuite insérer g pour garantir la capacité à déduire g. Une telle approche est pourtant incorrecte puisque l'insertion de g pourrait "activer" des cheminements de raisonnement menant à f. Pour montrer l'inexactitude de cette approche qui pourrait venir naturellement à l'esprit, considérons les exemples suivants.

Exemple 26. *Soient* $g = interrupteur\_on \vee ampoule\_ok \vee piece\_eclairee$ *une clause et* Γ *un ensemble composé de l'unique formule :*

1. $piece\_eclairee \Rightarrow interrupteur\_on \vee ampoule\_ok$.

Supposons que nous voulions transformer Γ *en* Γ' *tel que* Γ' *permette de déduire* g *sans le subsumer.* Γ *ne permet de déduire aucun impliquant strict de* g*. De nombreux opérateurs* $\ominus$ *qui apparaissent très naturels produiront en résultat un ensemble* Γ' *équivalent à* Γ *lorsque l'on contracte* Γ *par rapport à tous les impliquants stricts de* g*. Clairement,* $\Gamma' \cup \{g\}$ *permet cependant de déduire* $interrupteur\_on \vee ampoule\_ok$*, lequel est impliquant strict de* g*.* Γ' *permet donc de déduire* g *ainsi qu'un de ses impliquants stricts, ce qui est contraire à l'objectif poursuivi.*

Exemple 27. *Soient* $g = interrupteur\_on \vee ampoule\_ok \vee piece\_eclairee$ *une clause et* Γ *un ensemble composé des deux formules suivantes :*

1. $piece\_eclairee \Rightarrow interrupteur\_on \vee ampoule\_ok$,
2. $\neg interrupteur\_on \wedge \neg ampoule\_ok$

Supposons que nous voulions là aussi transformer Γ *en* Γ' *tel que* Γ' *permette de déduire* g *sans le subsumer.* Γ *ne permet de déduire aucun impliquant strict de* g *et infère même la négation de certains d'entre eux (par exemple, la négation de* $interrupteur\_on \vee ampoule\_ok$ *ou celle de* $piece\_eclairee$*). Comme pour l'exemple précédent, de nombreux opérateurs* $\ominus$ *qui apparaissent très naturels produiront en résultat un ensemble* Γ' *équivalent à* Γ *lorsque l'on contracte* Γ *par rapport à tous les impliquants stricts de* g*. Clairement,* $\Gamma' \cup \{g\}$ *permet cependant de déduire* $interrupteur\_on \vee ampoule\_ok$*. En effet,* $\Gamma' \cup \{g\}$ *trivialise* [1] *et ainsi* Γ' *permet aussi de déduire* g *ainsi qu'un de ses impliquants stricts, ce qui, là aussi, est contraire à l'objectif poursuivi.*

4.1.3.1 Utilisation d'un opérateur de contraction

Considérons que nous utilisions un opérateur concret de contraction à la AGM. En se positionnant dans le cadre propositionnel, celui-ci doit satisfaire les postulats de l'opérateur $\bullet$ de Katsuno et Mendelzon (voir [Katsuno & Mendelzon 1991a]) présenté au Chapitre 3 (Section 3.2).

Un tel opérateur respecte-t-il les propriétés de notre opérateur $\ominus$? Les deux premières propriétés de $\ominus$ trouvent respectivement une correspondance directe avec les postulats $(C3)$ de *succès* et $(C1)$ d'*inclusion*. La troisième propriété est, quant à elle, vérifiée par toute méthode de contraction se basant sur $\bullet$, par définition. Les propriétés de notre opérateur $\ominus$ sont bien respectées.

1. Propriété bien connue sous le nom latin de *ex falso sequitur quodlibet* signifiant *du faux tout peut être déduit*.

Nous devons tout de même nous poser une autre question toute aussi importante. Les propriétés supplémentaires que doit satisfaire un tel opérateur pour être un opérateur de contraction $\bullet$ sont-elles effectivement compatibles avec notre opérateur $\ominus$? Commençons par le postulat ($C2$), dit de *vacuité*, qui garantit que la base ne soit pas changée si h n'appartient pas aux conséquences de Γ. Dans l'optique d'utiliser un opérateur concret de contraction afin de transformer Γ en Γ' en contractant Γ par tous les impliquants stricts de g puis en ajoutant g, l'exemple 26 illustre bien que prendre un opérateur qui satisfasse cette propriété change fondamentalement le résultat [2]. Continuons avec le postulat ($C4$), dit de *préservation* qui indique que le résultat de la contraction par $\bullet$ peut dépendre de la syntaxe. Contrairement à la propriété de vacuité, prendre un opérateur qui satisfasse la propriété de préservation ne change pas fondamentalement le résultat, le statut de la préservation est neutre. Terminons par le postulat ($C5$) dit de *restauration*. Ce postulat est certainement le plus critiqué (voir [Makinson 1987] et [Hansson 1991]), mais là encore, prendre un opérateur qui satisfasse cette propriété ne change pas fondamentalement le résultat.

Un opérateur concret de contraction qui satisfasse les postulats de l'opérateur $\bullet$ fait donc bien partie de la famille des opérateurs $\ominus$.

4.1.3.2 Utilisation d'un opérateur d'effacement

Considérons maintenant que nous utilisions un opérateur concret de contraction qui satisfasse les postulats de l'opérateur $\triangleleft$ d'effacement défini par Katsuno et Mendelzon (voir [Katsuno & Mendelzon 1991a]) présenté au Chapitre 3 (Section 3.3).

Un tel opérateur respecte-t-il les propriétés de notre opérateur $\ominus$? Étant donnés les liens étroits qui existent entre l'opérateur de contraction $\bullet$ et l'opérateur d'effacement $\triangleleft$, les résultats devraient être semblables. En effet, les deux premières propriétés de $\ominus$ trouvent respectivement une correspondance avec les postulats ($E3$) de *succès faible* et ($E1$) d'*inclusion*. Et pour les mêmes raisons que celles mises en évidence lors de la considération de l'opérateur $\bullet$, la troisième propriété de $\ominus$ est par définition vérifiée.

Mais là aussi, nous devons tout de même nous demander si les propriétés supplémentaires que doit satisfaire un tel opérateur pour être un opérateur d'effacement $\triangleleft$ sont effectivement compatibles avec notre opérateur $\ominus$? Concernant le postulat ($E2$) de *vacuité faible*, lequel garantit que la base ne soit pas changée si $\neg h$ appartient aux conséquences de Γ, l'exemple 27 illustre bien la faillite de la vacuité faible et montre que prendre un opérateur qui satisfasse cette propriété change fondamentalement le résultat [3]. Concernant les postulats ($E4$) et ($E5$) de *préservation* et de *restauration*, respectivement, on retrouve les mêmes résultats que lors de la considération de l'opérateur $\bullet$. Aussi, pour le postulat ($E8$), dit de *règle disjonctive* qui examine séparément chaque monde possible pour rétracter h, prendre un opérateur qui satisfasse cette propriété ne change pas fondamentalement le résultat.

Notons aussi que le résultat ne serait pas affecté fondamentalement non plus, si l'on prenait un opérateur d'oubli (voir [Winslett 1989] et [Katsuno & Mendelzon 1991a]) présenté dans la

2. Notons cependant que par la suite, nous proposons une solution générale faisant intervenir de manière plus fine une famille d'opérateurs $\ominus$, solution pour laquelle prendre un opérateur de contraction qui satisfasse la propriété de vacuité n'altère pas fondamentalement le résultat.

3. Ce qui, rappelons le, n'est plus vérifié dans la solution générale que nous proposons ensuite.

section 3.3, qui n'a pour différence avec un opérateur d'effacement $\triangleleft$ que le fait d'affecter les mondes possibles dans lesquels $\neg\mu$ tient.

Un opérateur concret de contraction qui satisfasse les postulats des opérateurs d'effacement ou d'oubli discutés fait bien partie de la famille des opérateurs $\ominus$.

4.2 Solution générale

Une famille d'opérateurs $\oplus$ qui permettent de transformer Γ en Γ' de manière à ce que g puisse en être déduit mais pas subsumé, sont basés sur une utilisation un peu plus élaborée des opérateurs $\ominus$ que celle dont nous venons de montrer la maladresse.

Intuitivement, l'observation clé menant à notre solution est la suivante. Notons f un impliquant strict de g. Nous devons retirer f de Γ, aussi bien que toute possibilité de dériver f de Γ. Si nous retirons $g \Rightarrow f$ (dont l'équivalent clausal est $\neg g \vee f$), alors au-delà de retirer f de Γ, nous prémunissons f d'être dérivable lorsque g est ensuite ajouté.

Par souci de lisibilité et par abus de notation, par la suite nous noterons $\Gamma \ominus \{h\}$ en lieu et place de $\Gamma \ominus h$. Aussi, nous considérons que l'opérateur d'union ensembliste $\cup$ sur des ensembles de formules est toujours suivi par une fermeture déductive [4].

Aussi, il est important de signaler que dans cette solution nous considérons que Γ est un ensemble cohérent. Nous verrons dans la suite comment traiter le cas où Γ est incohérent.

4.2.1 Opérateur $\oplus$ de prédominance

Commençons par définir un opérateur $\oplus_f$ qui ne considère qu'un *seul* impliquant strict f de g.

Définition 88 (opérateur $\oplus_f$)**.** *Soit f un impliquant strict de g.*

$$\Gamma \oplus_f g =_{def} \Gamma \ominus \{g \Rightarrow f\} \cup \{g\}.$$

Le théorème qui suit établit que le résultat de $\Gamma \oplus_f g$ est cohérent (1), que g peut être déduit (2) mais pas subsumé par f (3) à partir de $\Gamma \oplus_f g$.

Théorème 8.
(1) $\Gamma \oplus_f g$ *est cohérent.*
(2) $\Gamma \oplus_f g \models g$.
(3) $\Gamma \oplus_f g \not\models f$.

Démonstration.
(1) Selon la définition 87, la propriété (1) de *succès* de $\ominus$ implique que $\Gamma \ominus \{g \Rightarrow f\} \not\models g \Rightarrow f$. Ainsi, $\Gamma \ominus \{g \Rightarrow f\} \not\models \neg g \vee f$ et donc que $\Gamma \ominus \{g \Rightarrow f\} \not\models \neg g$. En conséquence à cela et au fait que Γ doit nécessairement être cohérent, $\Gamma \oplus_f g = \Gamma \ominus \{g \Rightarrow f\} \cup \{g\} \not\models \bot$.
(2) Évident puisque $\Gamma \oplus_f g$ contient g.
(3) Supposons le contraire. Si $\Gamma \oplus_f g \models f$ alors $\Gamma \ominus \{g \Rightarrow f\} \cup \{g\} \models f$. Ainsi, en utilisant le théorème de la déduction, $\Gamma \ominus \{g \Rightarrow f\} \models g \Rightarrow f$, ce qui contradictoire avec la propriété 1 de *succès* de l'opérateur $\ominus$ de contraction. □

4. Notons que seul un opérateur d'union ensembliste "clos déductivement" satisfait les postulats de l'opérateur + d'expansion AGM (voir Théorème 2).

Il reste maintenant à étudier comment cette approche peut être étendue pour considérer tous les impliquants stricts de g. A cet égard, notons immédiatement que si g est une clause formée de n littéraux différents, il suffit de considérer les n *impliquants premiers* de g, c.-à-d. les n plus grandes sous-clauses strictes de g. De fait, si Γ' ne permet pas de déduire le moindre premier impliquant de g alors aucun impliquant strict de g ne peut en être déduit.

Notons aussi qu'une généralisation immédiate en $\Gamma \oplus g =_{def} \Gamma \ominus \{g \Rightarrow \bigvee_i f_i\} \cup \{g\}$ où $\bigvee_i f_i$ représenterait la disjonction des impliquants premiers de g serait inappropriée dans la mesure où $\bigvee_i f_i$ est équivalent à g (n'oublions pas que $\ominus$ s'applique uniquement à des formules non tautologiques). Une généralisation très simple de la définition 88 d'opérateur $\oplus_f$ à tous les impliquants stricts f de g consiste à contracter Γ de $\{g \Rightarrow f\}$ puis à ajouter g, et cela pour chaque impliquant strict f de g, les uns à la suite des autres.

Définition 89 (opérateur $\oplus$)**.**

$$\Gamma \oplus g =_{def} \Gamma \ominus \{g \Rightarrow f\} \cup \{g\},$$

pour tout impliquant premier f de g.

Par généralisation immédiate des propriétés prouvées pour l'opérateur $\oplus_f$ (voir Théorème 8), $\oplus$ obtient les propriétés suivantes nous indiquant que le résultat de $\Gamma \oplus g$ est cohérent (1), d'une part, que g peut être déduit (2) mais pas subsumé (3) à partir de $\Gamma \oplus g$, d'autre part.

Propriété 5.
(1) $\Gamma \oplus g$ *est cohérent.*
(2) $\Gamma \oplus g \models g$.
(3) $\Gamma \oplus g \not\models f$*, pour tout impliquant strict f de g.*

Cependant, d'un point de vue opérationnel, la définition 89 d'opérateur $\oplus$ ne nous indique pas quelles contraintes supplémentaires doivent intervenir sur la nature de l'opérateur $\ominus$ pour assurer l'*unicité* de $\Gamma \oplus g$ pour tout impliquant strict f de g.

Nous proposons ici, afin de garantir l'unicité de notre processus de transformation de Γ en Γ', d'avoir recours à la *contraction multiple* (voir [Fuhrmann & Hansson 1994]) présentée au Chapitre 3, Section 3.2. Pour être plus précis, $\ominus$ est maintenant considéré comme une opération binaire qui n'opère plus sur une simple clause h mais sur ensemble de clauses Λ. Quand Γ et Λ sont des ensembles de clauses, $\Gamma \ominus \Lambda$ produit un sous-ensemble de (la fermeture déductive clausale de) Γ qui n'implique aucune clause de Λ.

Nous définissons donc $\oplus'$ comme suit en faisant référence à un opérateur $\ominus$ de contraction multiple.

Définition 90 (opérateur $\oplus'$)**.** *Soit $f_1, \ldots, f_n$ l'énumération des impliquants premiers de g.*

$$\Gamma \oplus' g =_{def} \Gamma \ominus \{g \Rightarrow f_i\}_{i=1..n} \cup \{g\}.$$

Il est aisé de montrer que cette définition jouit des propriétés prouvées pour l'opérateur $\oplus_f$ (voir Théorème 8). Ainsi, le résultat de $\Gamma \oplus' \Lambda$ est cohérent (1) et permet bien de déduire g (2) et de ne pas le subsumer (3).

Propriété 6.
(1) $\Gamma \oplus' g$ *est cohérent.*
(2) $\Gamma \oplus' g \models g$.
(3) $\Gamma \oplus' g \not\models f$, *pour tout impliquant strict* f *de* g.

Cas particulier où Γ est incohérent

Lorsque Γ est incohérent, les opérateurs $\oplus$ et $\oplus'$ ne peuvent s'appliquer car l'application de $\ominus$ à un ensemble incohérent de formules n'a pas été définie. Dans cette situation particulière, nous préconisons une première étape de révision de Γ par g à l'aide d'un opérateur sémantique $*$ de révision à la AGM (voir [Alchourrón *et al.* 1985]), étape permettant de rétablir la cohérence de Γ en la présence de g. A l'issue de cette opération, nous obtenons un ensemble cohérent de formules qui permet accessoirement de déduire g et nous pouvons ensuite appliquer les opérateurs $\oplus$ ou $\oplus'$ pour obtenir un ensemble qui permette de déduire g sans le subsumer.

Notons que plutôt que d'utiliser un opérateur de révision à la AGM qui insère g, nous aurions pu utiliser toute opération qui conduit à rendre Γ cohérent et donc à transformer Γ de manière à pouvoir appliquer les opérateurs $\oplus$ et $\oplus'$. En optant pour une révision par g selon des opérateurs à la AGM, nous optons pour une approche indépendante de la syntaxe, pour une révision modulo des *changements minimaux* privilégiant g, en accord avec notre cadre sémantique et l'objectif final au sujet de g.

4.2.2 Propriétés intéressantes de l'opérateur $\oplus$

Intéressons nous maintenant à quelques propriétés que revêt notre opérateur $\oplus$ de prédominance. Tout d'abord, comme l'exprime formellement la propriété 7, si Γ n'infère aucun $g \Rightarrow f$ pour tout impliquant strict f de g alors faire prédominer g dans Γ revient simplement à ajouter g dans Γ.

Propriété 7. *Si* $\Gamma \not\models (g \Rightarrow f)$, *pour tout impliquant strict* f *de* g, *alors* $\Gamma \oplus g \equiv \Gamma \cup \{g\}$.

Démonstration. Par définition d'un opérateur $\ominus$ de contraction pratique, pour tout impliquant strict f de g, si $\Gamma \not\models (g \Rightarrow f)$ alors $\Gamma \ominus (g \Rightarrow f) \equiv \Gamma$. Ainsi, par définition de $\oplus$, si $\Gamma \not\models (g \Rightarrow f)$, pour tout impliquant strict f de g, alors $\Gamma \oplus g \equiv \Gamma \cup \{g\}$. □

De plus, lorsque g appartient déjà à Γ alors si $\Gamma \not\models (g \Rightarrow f)$, pour tout impliquant strict f de g, faire prédominer g dans Γ n'entraîne aucune modification dans Γ. La propriété 8 exprime cela.

Propriété 8. *Si* $\Gamma \not\models (g \Rightarrow f)$, *pour tout impliquant strict* f *de* g, *et que* $g \in \Gamma$ *alors* $\Gamma \oplus g \equiv \Gamma$.

Démonstration. Selon la propriété 7, si $\Gamma \not\models (g \Rightarrow f)$, pour tout impliquant strict f de g, alors $\Gamma \oplus g \equiv \Gamma \cup \{g\}$. Par définition, si $g \in \Gamma$ alors $\Gamma \cup \{g\} \equiv \Gamma$. Ainsi, si $\Gamma \not\models (g \Rightarrow f)$, pour tout impliquant strict f de g, et que $g \in \Gamma$ alors $\Gamma \oplus g \equiv \Gamma$. □

Une extension de cette dernière propriété est que suite à une opération de prédominance de g dans Γ, faire prédominer g à nouveau n'entraîne aucune modification dans l'ensemble résultant de la première opération de prédominance.

Propriété 9. $(\Gamma \oplus g) \oplus g \equiv \Gamma \oplus g$.

Démonstration. Par définition de l'opérateur $\oplus$, $(\Gamma \oplus g)$ ne subsume pas g et $g \in (\Gamma \oplus g)$. Ainsi, en accord avec la propriété 8, $(\Gamma \oplus g) \oplus g \equiv \Gamma \oplus g$. □

Il est important de noter à ce stade que l'opération de prédominance via $\oplus$ n'est pas associative. Ainsi, $(\Gamma \oplus g_1) \oplus g_2 \not\equiv (\Gamma \oplus g_2) \oplus g_1$. En effet, lors de l'opération de prédominance de g_2 dans l'ensemble résultant de $\Gamma \oplus g_1$, l'ajout de g_2 pourrait activer des chemins de raisonnements permettant de déduire $g_1 \Rightarrow f_1$, pour au moins un impliquant strict f_1 de g_1. Pour s'en convaincre, considérons l'exemple suivant.

Exemple 28. *Soit Γ un ensemble contenant les deux formules suivantes :*

1. $jour \Rightarrow lumiere\_jour$,
2. $lumiere\_jour \Rightarrow piece\_eclairee$,

exprimant respectivement que "S'il fait jour alors la lumière du jour peut être observée" et que "Si la lumière du jour peut être observée alors la pièce est éclairée".

Soient g_1, g_2 deux formules telles que :

1. $g_1 = (interrupteur\_on \wedge ampoule\_ok) \Rightarrow piece\_eclairee$,
2. $g_2 = jour$,

où g_1 et g_2 exprime respectivement "Si l'interrupteur est enclenché et si l'ampoule n'est pas cassée alors la pièce est éclairée"g_2 et "Il fait jour".

Considérons l'opération $(\Gamma \oplus g_1) \oplus g_2$:

– *Γ ne subsumant pas g_1, $\Gamma \oplus g_1 = \Gamma \cup \{g_1\}$.*
– *$(\Gamma \oplus g_1)$ ne subsumant pas g_2, $(\Gamma \oplus g_1) \oplus g_2 = \Gamma \cup \{g_1, g_2\}$.*

Considérons maintenant l'opération $(\Gamma \oplus g_2) \oplus g_1$:

– *Γ ne subsumant pas g_2, $\Gamma \oplus g_2 = \Gamma \cup \{g_2\}$.*
– *Par contre, il est clair que $(\Gamma \oplus g_2)$ subsume g_1, une modification dans $\Gamma \cup \{g_2\}$ doit nécessairement être opérée.*

Ainsi, $(\Gamma \oplus g_1) \oplus g_2 \not\equiv (\Gamma \oplus g_2) \oplus g_1$.

4.2.3 Contre-partie sémantique

Il est aisé de fournir un opérateur sémantique alternatif à notre opérateur $\oplus'$ dans le cas général. En considérant $g = a_1 \vee \ldots \vee a_n$ où $a_1, \ldots, a_n$ sont les littéraux de g, un tel opérateur peut être défini comme suit :

1. $\mathcal{M}(\Gamma \oplus' g) \neq \emptyset$,
2. $\forall i \in [1 \ldots n], \exists m_i \in \mathcal{M}(\Gamma \oplus' g)$ t.q. $\{\neg a_1, \ldots, \neg a_n\} \setminus \{\neg a_i\} \subseteq m_i$,
3. $\mathcal{M}(\Gamma \cup \{g\}) \subseteq \mathcal{M}(\Gamma \oplus' g) \subseteq \mathcal{M}(\{g\})$.

Les deux premiers items et la seconde partie du troisième assurent la satisfaction de la propriété 6 (le second item empêche tout impliquant strict f de g d'être dérivable). La première partie du troisième item assure que les opérateurs de contraction utilisés par $\oplus'$ délivrent

bien des sous-ensembles de la fermeture déductive de Γ. De manière assez évidente, des caractérisations sémantiques plus fines dépendront de l'opérateur de contraction effectivement sélectionné.

Un opérateur de contraction permettant une caractérisation sémantique plus fine peut être construit à partir des assignements fidèles de Katsuno et Mendelzon et de l'identité de Harper (voir [Katsuno & Mendelzon 1991a] et [Gärdenfors 1988], respectivement). Rappelons que les assignements fidèles sont des pré-ordres sur les interprétations (voir Définition 78) qui permettent d'exprimer la révision comme une sélection de modèles minimaux (voir Théorème 4) et que l'identité de Harper (voir Propriété 4) permet de définir un opérateur de contraction, lequel respecte les propriétés de l'opérateur de contraction AGM $\div$ (voir Théorème 6), à partir d'un opérateur de révision et de l'intersection ensembliste. [5]

Ainsi, $\Gamma \ominus \{g \Rightarrow f\}$ peut être défini comme suit.

$$\Gamma \ominus \{g \Rightarrow f\} \equiv \Gamma \cap \{\gamma \circ \neg(g \Rightarrow f)\} \text{ où}$$
$$\gamma \text{ est la conjonction des formules de } \Gamma \text{ et}$$
$$Mod(\gamma \circ \neg(g \Rightarrow f)) = min(Mod(\neg(g \Rightarrow f)); \leq_\gamma) \text{ où}$$

$\leq_\gamma$ est un pré-ordre sur l'ensemble des interprétations de γ composé de deux niveaux :
- les modèles de γ au premier niveau (c.-à-d. les interprétations les plus préférées),
- les contres-modèles de γ au second (c.-à-d. les interprétations les moins préférées).

Voyons cela dans un exemple où nous considérons l'opérateur sémantique alternatif à notre opérateur $\oplus'$ comme défini à partir d'un opérateur de contraction construit à partir des assignements fidèles de Katsuno et Mendelzon et de l'identité de Harper.

Puisque nous devrons énumérer l'ensemble des modèles de γ, dans un souci de lisibilité, nous limiterons notre langage aux trois variables propositionnelles de notre exemple introductif, à savoir *interrupteur_on*, *ampoule_ok* et *piece_eclairee*, considérées dans cet ordre pour la valuation. On notera donc $(1, 0, 0)$ l'interprétation qui rend *interrupteur_on* a *vrai*, *ampoule_ok* et *piece_eclairee* à *faux*.

Exemple 29. *Considérons une règle* $g = (interrupteur\_on \wedge ampoule\_ok) \Rightarrow piece\_eclairee$ *exprimant que "Si l'interrupteur est enclenché et si l'ampoule n'est pas cassée alors la pièce est éclairée", que nous voudrions voir préempter dans un ensemble* $\Gamma = \{interrupteur\_on \Rightarrow piece\_eclairee\}$ *composé d'une unique règle exprimant que "Si l'interrupteur est enclenché alors la pièce est éclairée", laquelle implique g et devrait donc être retirée de* Γ*, intuitivement.*

Considérons maintenant g et Γ *au format clausal. Soit* $g = \neg interrupteur\_on \vee \neg ampoule\_ok \vee piece\_eclairee$ *une clause que nous voudrions voir préempter dans* $\Gamma = \{\neg interrupteur\_on \vee piece\_eclairee\}$. Γ *peut être vu comme la formule* $\gamma = \neg interrupteur\_on \vee piece\_eclairee$.

Les impliquants premiers de g sont les clauses $f_1 = \neg interrupteur\_on \vee \neg ampoule\_ok$, $f_2 = \neg interrupteur\_on \vee piece\_eclairee$ *et* $f_3 = \neg ampoule\_ok \vee piece\_eclairee$.

Considérons tout d'abord la contraction de Γ *par* $\{g \Rightarrow f_2\}$ [6] *:*

5. Bien que l'identité de Harper se base sur l'opérateur général $*$ de révision AGM et que les assignements fidèles se basent, eux, sur l'opérateur de révision propositionnel $\circ$ défini par Katsuno et Mendelzon, le résultat reste valable puisqu'il y a bien correspondance entre ces deux opérateurs (voir Théorème 3).

6. Notons que l'ordre dans lequel les impliquants premiers de g sont choisis n'a aucune importance dans notre approche sémantique puisque le résultat d'ajouts successifs de modèles à un ensemble de modèles ne dépend pas de l'ordre dans lequel ces modèles sont ajoutés.

$$\Gamma \ominus \{g \Rightarrow f_2\} \equiv \Gamma \cap \{\gamma \circ \neg(g \Rightarrow f_2)\} \text{ où}$$
$$Mod(\gamma \circ \neg(g \Rightarrow f_2)) = min(Mod(\neg(g \Rightarrow f_2)); \leq_\gamma).$$

En nous aidant des tables de vérité de la table 4.1, on peut établir que

$$\leq_\gamma = \{\{(0,0,0),(0,0,1),(0,1,0),(0,1,1),(1,0,1),(1,1,1)\} < \{(1,0,0),(1,1,0)\}\},$$
$$\text{et que } min(Mod(\neg(g \Rightarrow f_2)); \leq_\gamma) = Mod(\neg(g \Rightarrow f_2)) = \{(1,0,0)\}.$$

Ainsi, l'intersection de Γ *avec l'ensemble* $\{interrupteur\_on \wedge \neg ampoule\_ok \wedge \neg piece\_eclairee\}$ *(représentant, à l'équivalence logique près les mondes de* $(\gamma \circ \neg(g \Rightarrow f_2))$ *est vide. Nous venons donc de retirer de* Γ *la formule* $\neg interrupteur\_on \vee piece\_eclairee$*, qui est un impliquant strict de g.*

Γ *étant maintenant vide, il paraît évident que les étapes de contraction de* Γ *par les impliquants premiers de g restants sont inutiles.*

Ainsi, suite à l'opération d'ajout de g, l'ensemble résultant de l'application de notre opérateur sémantique alternatif à $\oplus'$*, à savoir* $\Gamma' = \{\neg interrupteur\_on \vee \neg ampoule\_ok \vee piece\_eclairee\}$*, correspond bien au résultat escompté.*

interrupteur_on	*ampoule_ok*	*piece_eclairee*	g	f_2	$\neg(g \Rightarrow f_2)$
0	0	0	1	1	0
0	0	1	1	1	0
0	1	0	1	1	0
0	1	1	1	1	0
1	0	0	1	0	1
1	0	1	1	1	0
1	1	0	0	0	0
1	1	1	1	1	0

TABLE 4.1 – Tables de vérité de l'exemple 29.

4.3 Étude algorithmique

L'approche proposée à la section précédente repose sur des opérateurs de révision AGM et sur des tests de satisfiabilité booléens qui souffrent tout deux d'une complexité élevée dans le pire des cas. En effet, d'un côté le problème de vérification SAT est un problème NP-complet, et, d'un autre côté les opérateurs de révision de croyances AGM appartiennent au second niveau de la hiérarchie polynomiale (voir [Eiter & Gottlob 1992]). Cependant, des progrès récents de la communauté SAT permettent la manipulation efficace d'ensembles de clauses, bien que la manipulation dans le pire des cas soit toujours intraitable [7].

L'objectif de cette "étude algorithmique" n'est pas de développer une autre plateforme expérimentale pour les problèmes relatifs à la révision booléenne. Au lieu de cela, nous avons

7. Du moins tant que l'affirmation $P = NP$ n'est pas définitivement établie.

implémenté un outil dont le but est simplement d'aider un utilisateur à la compréhension étape par étape de notre approche dans le cadre clausal dont une brève description suit.

Grosso modo, quand un utilisateur cherche à insérer une clause g (qui n'est ni incohérente ni tautologique) dans un ensemble de clauses Γ existant (lequel est cohérent) qui ne doit en aucun cas la subsumer, et, afin de reproduire la solution générale employant la contraction multiple, les n plus grandes sous-clauses strictes f_i de g sont considérées successivement. Pour chaque f_i, le système vérifie la cohérence de $\Gamma \cup \{\neg(g \Rightarrow f_i)\}$ [8]. Une fois l'ensemble des f_i considérés, il est alors précisé à l'utilisateur, pour chaque source d'incohérence détectée, comment la cohérence peut être retrouvée afin d'éviter à g d'être subsumée. Différentes politiques pour la restauration de la cohérence à la AGM, lesquelles se focalisent principalement sur une politique de changement minimal, sont alors proposées à l'utilisateur. Finalement, une fois que la cohérence a été restaurée et que par conséquent la théorie résultante est assurée de ne pas subsumer g, g est ajoutée à la dite théorie.

4.3.1 Extraction des sources minimales d'incohérence

Nous présentons ici différents outils permettant de faire prédominer une clause g dans un ensemble de clauses Γ. Ceux-ci se basent sur deux méthodes différentes d'extraction de sources minimales d'incohérence. Notons bien que pour notre problème, le nombre de sources minimales d'incohérence est supposé comme relativement limité, il en est de même concernant leur taille. Nous reviendrons sur ce point dans la suite.

4.3.1.1 Extraction de l'ensemble des MUSes

Une particularité intéressante de notre plateforme est que celle-ci est en mesure de présenter à l'utilisateur les différentes clauses menant à l'incohérence, lui permettant ainsi de prendre les décisions adéquates en ayant une connaissance complète des différents aspects des conflits à résoudre. Pour ce faire, nous faisons appel à la notion de MUSes ("Minimally Unsatisfiable Subformulae" ou "noyaux minimaux incohérents") qui sont des ensembles de clauses incohérents tels que le retrait de l'une de leurs clauses permet de retrouver la cohérence. Les MUSes représentent donc les "explications" de l'incohérence les plus fines en terme du nombre de clauses mises en jeu menant à une contradiction (voir Chapitre 1, Section 1.4 pour plus de détails).

L'algorithme `PreempteM` que nous proposons ici (voir Algorithme 1), se base donc sur la notion de MUSes pour l'extraction des sources minimales d'incohérence. Ainsi, pour toute plus grande sous-clause stricte de g, les différents MUSes de $\Gamma \cup \{\neg(g \Rightarrow f)\}$ sont extraits puis ajoutés à l'ensemble $Muses$ lorsque $\Gamma \cup \{\neg(g \Rightarrow f)\}$ est incohérent (lignes 3 à 5). Une fois tous les impliquants premiers f de g considérés, les différents MUSes sont présentés à l'utilisateur afin qu'il choisisse une ou plusieurs clauses appartenant à Γ [9] à retirer pour chacun d'eux (lignes 6 à 8). Les clauses choisies sont ensuite retirées de Γ (ligne 9) avant que la clause g n'y soit ajoutée (ligne 10).

8. En effet, si $g \Rightarrow f_i$ est déductible de Γ, Γ sera alors rendu incohérent par l'ajout de $\neg(g \Rightarrow f_i)$.

9. Notons que certaines clauses des MUSes n'appartiennent pas à Γ mais appartiennent aux différents $\neg(g \Rightarrow f)$ considérés lors de l'opération d'extraction des MUSes de $\Gamma \cup \{\neg(g \Rightarrow f)\}$.

Données : Γ un ensemble cohérent de clauses et g une clause cohérente et non tautologique
Résultat : Γ' un ensemble cohérent de clauses t.q. $\Gamma' \models g$ et Γ' ne subsume pas g

début
  $Muses \longleftarrow \emptyset$;
  pour tous les plus grandes sous-clauses strictes f de g **faire**
    si $\Gamma \cup \{\neg(g \Rightarrow f)\}$ est incohérent **alors**
      $Muses \longleftarrow Muses \cup \text{MUSES}(\Gamma \cup \{\neg(g \Rightarrow f)\})$;
  $SUPR \longleftarrow \emptyset$;
  pour tous les $mus \in Muses$ **faire**
    $SUPR \longleftarrow SUPR \cup$ { clauses de mus appartenant à Γ choisies par l'utilisateur };
  $\Gamma \longleftarrow \Gamma \setminus SUPR$;
  retourner $\Gamma \cup \{g\}$;

Algorithme 1: `PreempteM`.

Notons que $\neg(g \Rightarrow f)$ est un ensemble de clauses composé de la négation de chaque littéral de f et de la clause g. f étant une plus grande sous-clause stricte de g, ceci conduit la clause g a être réduit, par propagation unitaire, au seul de ses littéraux qui n'est pas un littéral de f. Par exemple, quand $g = a \vee b \vee c$ et que $f = a \vee b$, $\neg(g \Rightarrow f) = \{\neg a, \neg b, c\}$.

Notons aussi que, dans l'objectif de reproduire le plus fidèlement une opération de contraction multiple et ainsi de s'approcher au plus près de la définition 90 de notre opérateur $\oplus'$ de prédominance, l'ordre dans lequel les premiers impliquants f de g sont considérés n'influe pas sur le résultat. En effet, les MUSes sont extraits pour chaque premier impliquant dans un premier temps et réparés de manière indépendante, selon les directives de l'utilisateur, dans un second temps.

Voyons cela dans l'exemple qui suit, lequel s'inspire de notre exemple canonique. Nous supposons dans cet exemple que l'utilisateur adopte une approche neutre quant à la sélection des clauses à retirer de Γ, ne privilégiant pas une clause plutôt qu'une autre et choisisse ainsi de retirer toutes les clauses des MUSes que nous lui présentons.

Exemple 30. *Considérons une règle $g = (interrupteur\_on \wedge ampoule\_ok) \Rightarrow piece\_eclairee$ exprimant que "Si l'interrupteur est enclenché et si l'ampoule n'est pas cassée alors la pièce est éclairée", que nous voudrions voir préempter dans un ensemble Γ composé des règles suivantes :*

1. $interrupteur\_on \Rightarrow piece\_eclairee$,
2. $jour \Rightarrow piece\_eclairee$,
3. $(ampoule\_ok \wedge interrupteur\_auto) \Rightarrow piece\_eclairee$,
4. $interrupteur\_auto$,
5. $interrupteur\_on \Rightarrow \neg interrupteur\_auto$.

La première règle de Γ exprime que "Si l'interrupteur est enclenché alors la pièce est éclairée", elle implique g et nous voudrions donc qu'elle ne puisse plus être dérivée de Γ.

La deuxième règle exprime que "Si il fait jour alors la pièce est éclairée", elle n'implique pas g et n'a donc aucune raison d'être retirée de Γ. *La troisième règle exprime quant à elle que "Si l'ampoule n'est pas cassée et que l'interrupteur est en mode automatique alors la pièce est éclairée", elle implique g en la présence de la quatrième règle qui exprime que "L'interrupteur est en mode automatique", nous voudrions donc que* Γ *ne puisse plus dériver ces deux règles à la fois. La cinquième règle de* Γ *exprime que "Si l'interrupteur est enclenché alors il n'est pas en mode automatique" et implique g en la présence de la quatrième règle, nous voudrions donc, là aussi, que* Γ *ne puisse dériver ces deux règles à la fois.*

Considérons maintenant g et Γ *sous leur forme clausale. Soit* $g = \neg interrupteur\_on \vee \neg ampoule\_ok \vee piece\_eclairee$ *une clause cohérente et non tautologique que nous voudrions voir préempter dans un ensemble* Γ *cohérent composé des clauses suivantes :*

1. $\neg interrupteur\_on \vee piece\_eclairee$,
2. $\neg jour \vee piece\_eclairee$,
3. $\neg ampoule\_ok \vee \neg interrupteur\_auto \vee piece\_eclairee$,
4. $interrupteur\_auto$,
5. $\neg interrupteur\_on \vee \neg interrupteur\_auto$.

Trivialement, l'ensemble des plus grandes sous-clauses strictes de g est composé des clauses suivantes :

1. $\neg interrupteur\_on \vee \neg ampoule\_ok$,
2. $\neg interrupteur\_on \vee piece\_eclairee$,
3. $\neg ampoule\_ok \vee piece\_eclairee$.

Il faut maintenant, pour toute plus grande sous-clause f de g, extraire les MUSes de $\Gamma \cup \{\neg(g \Rightarrow f)\}$ *lorsque* $\Gamma \cup \{\neg(g \Rightarrow f)\}$ *est incohérent.*

$$\begin{gathered}
\text{Commençons par } f = \neg interrupteur_on \vee \neg ampoule_ok \\
\neg(g \Rightarrow f) = \{interrupteur_on, ampoule_ok, piece_eclairee\} \\
\Gamma \cup \{\neg(g \Rightarrow f)\} \text{ est incohérent} \\
\text{On en extrait l'unique MUS } M_1 \\
M_1 = \{interrupteur_on, \neg interrupteur_on \vee \neg interrupteur_auto, interrupteur_auto\} \\
Muses = \{M_1\}
\end{gathered}$$

$$\begin{gathered}
\text{Poursuivons avec } f = \neg interrupteur_on \vee piece_eclairee \\
\neg(g \Rightarrow f) = \{interrupteur_on, \neg piece_eclairee, \neg ampoule_ok\} \\
\Gamma \cup \{\neg(g \Rightarrow f)\} \text{ est incohérent} \\
\text{On en extrait à nouveau le MUS } M_1 \text{ et le MUS } M_2 \\
M_2 = \{interrupteur_on, \neg interrupteur_on \vee piece_eclairee, \neg piece_eclairee\} \\
Muses = \{M_1, M_2\}
\end{gathered}$$

$$\begin{gathered}
\text{Terminons avec } f = \neg ampoule_ok \vee piece_eclairee \\
\neg(g \Rightarrow f) = \{ampoule_ok, \neg piece_eclairee, \neg interrupteur_on\} \\
\Gamma \cup \{\neg(g \Rightarrow f)\} \text{ est lui aussi incohérent} \\
\text{On en extrait l'unique MUS } M_3
\end{gathered}$$

$$M_3 = \{\neg ampoule\_ok \vee \neg interrupteur\_auto \vee piece\_eclairee, interrupteur\_auto, ampoule\_ok, \neg piece\_eclairee\}$$
$$Muses = \{M_1, M_2, M_3\}$$

Suite à l'étape d'extraction des MUSes des différents $\Gamma \cup \{\neg(g \Rightarrow f)\}$*, les MUSes sont présentés à l'utilisateur afin qu'il puisse choisir quelles clauses retirer pour chaque MUS. Ici, trois MUSes sont présentés à l'utilisateur, composés au total de dix clauses, dont sept clauses différentes et quatre clauses appartenant à* Γ*.*

En supposant que l'utilisateur adopte une approche neutre quant à la sélection des clauses à retirer de Γ *et choisisse ainsi de retirer toutes les clauses des MUSes appartenant à* Γ*, alors* Γ *n'est plus composé que de l'unique clause* $\neg jour \vee piece\_eclairee$*, à savoir la règle sous forme clausale de notre* Γ *de départ exprimant que "Si il fait jour alors la pièce est éclairée", laquelle n'implique pas g et n'a donc aucune raison d'être retirée de* Γ*.*

Finalement, suite à l'opération d'ajout de g, Γ' *est composé des clauses suivantes :*

1. $\neg jour \vee piece\_eclairee$*,*
2. $\neg interrupteur\_on \vee \neg ampoule\_ok \vee piece\_eclairee$*.*

Comme escompté, $\Gamma' \models g$ *et* $\Gamma' \not\models f$*, pour tout* f *impliquant strict de* g*.*

4.3.1.2 Extraction d'une couverture minimale incohérente

Au chapitre 1 et particulièrement à la section 1.4, nous avons pu voir que la détection et l'énumération exhaustive de tous les MUSes d'une formule CNF est intraitable dans le pire des cas. Rappelons en effet que vérifier si un ensemble de clauses est un MUS est *DP*-complet (voir [Papadimitriou & Wolfe 1988]) et que vérifier si une formule appartient à l'ensemble des MUSes d'une formule incohérente est Σ_2^p-complet (voir [Eiter & Gottlob 1992]). De plus, rappelons aussi qu'une formule composée de n clauses peut contenir un nombre exponentiel de MUSes en la taille de la formule puisque celle-ci peut contenir $C_n^{n/2}$ MUSes dans le pire des cas.

C'est pourquoi nous proposons une approche alternative concernant l'extraction des incohérences issues des différents $\Gamma \cup \{\neg(g \Rightarrow f)\}$ considérés. En effet, l'algorithme `PreempteC` (voir Algorithme 2) que nous proposons, contrairement à notre algorithme `PreempteM` (voir Algorithme 1), ne nécessite pas de calculer exhaustivement les MUSes des différents $\Gamma \cup \{\neg(g \Rightarrow f)\}$ considérés, mais un sous-ensemble de ceux-ci appelée *couverture incohérente stricte*. Rappelons qu'une couverture incohérente stricte est un ensemble de MUSes, qui « couvre » assez de sources d'incohérence indépendantes pour que la cohérence soit rétablie lors de son retrait (voir Chapitre 1, Section 1.4).

Ainsi, seule la méthode d'extraction des incohérences de l'algorithme `PreempteC` diffère de celle de l'algorithme `PreempteM` (ligne 5).

Pour extraire une couverture incohérente stricte nous faisons appel à l'algorithme `couvMuses` (voir Algorithme 3), lequel extrait une couverture incohérente stricte *Couv* de l'union ensembliste d'un ensemble de clauses Γ et d'un ensemble de clauses *Nec* de manière à ce que $(\Gamma \setminus Couv) \cup Nec$ ne soit pas incohérent.

L'algorithme `CouvMuses` permet donc d'extraire une couverture incohérente stricte de $\Gamma \cup Nec$ où *Nec* joue le rôle d'un ensemble de clauses qui ne peut être extrait de Γ. En effet,

Données : Γ un ensemble cohérent de clauses et g une clause cohérente et non tautologique
Résultat : Γ' un ensemble cohérent de clauses t.q. $\Gamma' \models g$ et Γ' ne subsume pas g

début
  $Muses \longleftarrow \emptyset$;
  pour tous les plus grandes sous-clauses strictes f de g **faire**
    si $\Gamma \cup \{\neg(g \Rightarrow f)\}$ est incohérent **alors**
      $Muses \longleftarrow Muses \cup$ `CouvMuses`$(\Gamma \cup \{\neg(g \Rightarrow f)\})$;
  $SUPR \longleftarrow \emptyset$;
  pour tous les $mus \in Muses$ **faire**
    $SUPR \longleftarrow SUPR \cup$ { clauses de mus appartenant à Γ choisies par l'utilisateur };
  $\Gamma \longleftarrow \Gamma \setminus SUPR$;
  retourner $\Gamma \cup \{g\}$;

Algorithme 2: `PreempteC`.

Données : Γ et Nec deux ensembles cohérents de clauses t.q. $\Gamma \cup Nec$ soit incohérent
Résultat : $Couv$ une couverture incohérente stricte de $\Gamma \cup Nec$ t.q. $(\Gamma \setminus Couv) \cup Nec$ soit cohérent

début
  $Couv \longleftarrow \emptyset$;
  tant que $\Gamma \cup Nec$ est incohérent **faire**
    $Mus \longleftarrow$ MUS$(\Gamma \cup Nec)$;
    $Couv \longleftarrow Couv \cup \{Mus\}$;
    $\Gamma \longleftarrow \Gamma \setminus \{Mus\}$;
  retourner $Couv$;

Algorithme 3: `CouvMuses`.

une couverture incohérente stricte étant par définition un ensemble de MUS indépendants, son calcul revient à extraire un unique MUS (ligne 4), l'ajouter à la couverture (ligne 5), puis le retirer (ligne 6), en itérant ces trois étapes jusqu'à ce que la cohérence soit retrouvée (ligne 3) ; seulement, les clauses que nous ajoutons pour créer l'incohérence et qui constituent *Nec* ne doivent pas être retirées lors de l'extraction d'un MUS, celles-ci peuvent faire partie d'autres MUSes, d'où la nécessité de faire jouer à *Nec* le rôle d'un ensemble de clauses qui ne peut être extrait de Γ lors de l'extraction de la couverture incohérente stricte.

Il est important de noter toutefois qu'extraire une couverture incohérente stricte plutôt que l'ensemble exhaustif des MUSes ne garantit pas l'unicité des résultats de l'algorithme `PreempteC`. En effet, dans le cas où plusieurs couvertures incohérentes strictes existent, l'ensemble de clauses Γ' résultant peut différer selon la couverture incohérente stricte considérée lors de l'opération de réparation des sources d'incohérence.

4.3.2 Filtrage topographique

Dans les approches précédentes, quand il s'agit de présenter à l'utilisateur les sources d'incohérence détectées, nous optons pour une politique neutre de restauration de la cohérence. En effet, nous présentons de manière complète les clauses qui appartiennent aux différentes sources d'incohérence, en ne choisissant pas de présenter une clause plutôt qu'une autre. Ici, nous présentons une toute autre politique de restauration de la cohérence, laquelle se focalise sur le principe de *changement minimal* évoqué précédemment.

L'idée principale est de tenir compte de la *topographie* des sources d'incohérence afin de n'avoir à présenter à l'utilisateur qu'un nombre minimal de clauses pour chaque source d'incohérence. En effet, il peut être intéressant de privilégier, pour chaque source d'incohérence, les clauses se trouvant dans le plus de sources d'incohérence possibles.

Les algorithmes `PreempteMT` et `PreempteCT` que nous proposons ici (voir algorithmes 4 et 5 respectifs), revêtent cette politique de restauration de la cohérence soucieuse du principe de *changement minimal*. Ceux-ci ne diffèrent des algorithmes respectifs `PreempteM` et `PreempteT`, que par leur étape de "filtrage topographique" effectuée avant la présentation à l'utilisateur des sources d'incohérence détectées (ligne 6).

Ainsi, l'algorithme `FiltrageTopo` (voir Algorithme 6) a donc pour objectif, pour chaque MUS d'un ensemble de MUSes [10], de ne conserver dans un nouvel ensemble de MUSes $Muses'$, que les clauses appartenant à Γ qui apparaissent dans le plus grand nombre de MUS de l'ensemble.

Pour ce faire, il est nécessaire, dans un premier temps, d'établir un dictionnaire $Score$ indiquant dans combien de MUS une clause de Γ apparaît (ligne 3) [11]. L'algorithme `ScoreTopo` (voir Algorithme 7) permet d'arriver à cela. Ensuite, pour chaque MUS (ligne 4) et pour chaque clause du MUS appartenant à Γ (ligne 7), nous conservons les clauses du MUS ayant le "score" le plus élevé (vis-à-vis des clauses du même MUS) dans un nouveau MUS mus' (lignes 8 à 12) puis ajoutons ce MUS au nouvel ensemble de MUSes $Muses'$ (ligne 13).

10. Ensemble de MUSes pouvant représenter l'ensemble exhaustif des MUSes extrait dans la méthode `PreempteM`, ou alors une couverture incohérente stricte de cet ensemble extraite dans la méthode `PreempteC`.

11. Par exemple, un tuple (*interrupteur_auto*, 2) de $Score$ exprime que la clause *interrupteur_auto* appartient à deux MUSes de Γ.

Données : Γ un ensemble cohérent de clauses et g une clause cohérente et non tautologique
Résultat : Γ' un ensemble cohérent de clauses t.q. $\Gamma' \models g$ et Γ' ne subsume pas g

début
  $Muses \longleftarrow \emptyset$;
  pour tous les les plus grandes sous-clauses strictes f de g **faire**
    si $\Gamma \cup \{\neg(g \Rightarrow f)\}$ est incohérent **alors**
      $Muses \longleftarrow Muses \cup \text{MUSES}(\Gamma \cup \{\neg(g \Rightarrow f)\})$;
  $Muses \longleftarrow$ `FiltrageTopo`$(Muses, \Gamma)$;
  $SUPR \longleftarrow \emptyset$;
  pour tous les $mus \in Muses$ **faire**
    $SUPR \longleftarrow SUPR \cup \{$ clauses de mus appartenant à Γ choisies par l'utilisateur $\}$;
  $\Gamma \longleftarrow \Gamma \setminus SUPR$;
  retourner $\Gamma \cup \{g\}$;

Algorithme 4: `PreempteMT`.

Données : Γ un ensemble cohérent de clauses et g une clause cohérente et non tautologique
Résultat : Γ' un ensemble cohérent de clauses t.q. $\Gamma' \models g$ et Γ' ne subsume pas g

début
  $Muses \longleftarrow \emptyset$;
  pour tous les les plus grandes sous-clauses strictes f de g **faire**
    si $\Gamma \cup \{\neg(g \Rightarrow f)\}$ est incohérent **alors**
      $Muses \longleftarrow Muses \cup$ `CouvMuses`$(\Gamma \cup \{\neg(g \Rightarrow f)\})$;
  $Muses \longleftarrow$ `FiltrageTopo`$(Muses, \Gamma)$;
  $SUPR \longleftarrow \emptyset$;
  pour tous les $mus \in Muses$ **faire**
    $SUPR \longleftarrow SUPR \cup \{$ clauses de mus appartenant à Γ choisies par l'utilisateur $\}$;
  $\Gamma \longleftarrow \Gamma \setminus SUPR$;
  retourner $\Gamma \cup \{g\}$;

Algorithme 5: `PreempteCT`.

Données : $Muses$ un ensemble de MUSes et Γ un ensemble cohérent de clauses
Résultat : $Muses'$ l'ensemble cohérent de MUSes $Muses$ où chaque MUS a été filtré "topographiquement"

```
1  début
2      Muses' ⟵ ∅;
3      Score ⟵ ScoreTopo(Muses, Γ);
4      pour tous les mus ∈ Muses faire
5          ScoreMax ⟵ 0;
6          mus' ⟵ ∅;
7          pour tous les cl ∈ mus t.q. cl ∈ Γ faire
8              si Score[cl] > ScoreMax alors
9                  mus' ⟵ {cl};
10                 ScoreMax ⟵ Score[cl];
11             sinon si Score[cl] = ScoreMax alors
12                 mus' ⟵ mus' ∪ {cl};
13         Muses' ⟵ Muses' ∪ {mus'};
14     retourner Muses';
```

Algorithme 6: `FiltrageTopo`.

Données : $Muses$ un ensemble de MUSes et Γ un ensemble cohérent de clauses
Résultat : $Score$ un dictionnaire indiquant dans combien de MUS de $Muses$ une clause de Γ appartient

```
1  début
2      Soit Score un dictionnaire;
3      pour tous les cl ∈ Γ faire
4          Score[cl] ⟵ 0;
5      pour tous les mus ∈ Muses faire
6          pour tous les cl ∈ mus t.q. cl ∈ Γ faire
7              Score[cl] ⟵ Score[cl] + 1;
8      retourner Score;
```

Algorithme 7: `ScoreTopo`.

En procédant de la sorte, les sources d'incohérence que nous allons présenter à l'utilisateur ne contiennent plus que des clauses de Γ telles que si l'utilisateur choisit de ne retirer qu'une seule clause pour chaque source, alors le nombre total de clauses retirées de Γ sera le plus petit possible qui puisse permettre, suite à l'opération d'ajout de g, de garantir que g ne soit pas subsumé par l'ensemble résultant Γ'.

En retirant ainsi le plus petit nombre possible de clauses de Γ, Γ' est en quelque sorte, vis-à-vis de notre problème, un "sous-ensemble maximalement cohérent" de Γ. Selon les choix opérés par l'utilisateur quant à la sélection d'une clause à retirer de Γ pour chaque source d'incohérence, les différents "sous-ensembles maximalement cohérents" Γ' de Γ peuvent être énumérés. La notion de *sous-ensemble maximalement cohérent* étant duale de celle de MUS, il n'est alors pas étonnant de retrouver ces résultats [12].

En effectuant un filtrage topographique, nous ne présentons à l'utilisateur que les clauses *les plus responsables* au sein des différentes sources d'incohérences. Notons que ce principe se retrouve dans l'utilisation que fait Reiter des *hitting sets* pour le diagnostic (voir [Reiter 1987]), que l'on retrouve aussi dans le cadre de la révision (voir [Wassermann 2000]).

Il pourrait alors être tentant d'avoir recours à des algorithmes dédiés au problème Max-SAT [13] présenté au chapitre 1 (voir Section 1.4), lesquels ne nécessiteraient pas de calculer exhaustivement les MUSes pour fournir un "sous-ensemble maximalement cohérent". Cependant nous serions alors dans l'incapacité de fournir à l'utilisateur les explications complètes qui nous amènent à écarter certaines clauses. Pour ce faire, il serait nécessaire de calculer exhaustivement tous les "sous-ensembles maximalement cohérent" via ces algorithmes, ce que nos algorithmes tenant compte de la topographie des MUSes permettent de faire de manière assez compétitive grâce aux technologies d'extraction des MUSes existantes. Nous reviendrons plus tard sur ces questions calculatoires.

Voyons maintenant l'application de l'algorithme `PreempteMT` sur l'exemple 30.

Exemple 31. *Considérons donc* $g = \neg interrupteur\_on \vee \neg ampoule\_ok \vee piece\_eclairee$ *une clause cohérente et non tautologique que nous voudrions voir préempter dans un ensemble* Γ *cohérent composé des clauses suivantes :*

1. $\neg interrupteur\_on \vee piece\_eclairee$,
2. $\neg jour \vee piece\_eclairee$,
3. $\neg ampoule\_ok \vee \neg interrupteur\_auto \vee piece\_eclairee$,
4. $interrupteur\_auto$,
5. $\neg interrupteur\_on \vee \neg interrupteur\_auto$.

Suite à l'étape d'extraction des MUSes (voir Exemple 30),

$$Muses = \{M_1, M_2, M_3\} \text{ où :}$$
$$M_1 = \{interrupteur\_on, \neg interrupteur\_on \vee \neg interrupteur\_auto, interrupteur\_auto\}$$
$$M_2 = \{interrupteur\_on, \neg interrupteur\_on \vee piece\_eclairee, \neg piece\_eclairee\}$$

12. Notons toutefois, que la communauté qui s'intéresse aux aspects conceptuels de la représentation de la connaissance s'est surtout penchée sur le concept de sous-ensemble maximalement cohérent que ce soit en révision de croyances ou au sujet des bases stratifiées.

13. voir http://www.satcompetition.org/2011/.

$$M_3 = \{\neg ampoule\_ok \vee \neg interrupteur\_auto \vee piece\_eclairee, interrupteur\_auto, ampoule\_ok, \neg piece\_eclairee\}$$

Il convient maintenant, avant de présenter ces MUSes à l'utilisateur, de les filtrer selon la méthode présentée dans l'algorithme `FiltrageTopo`*. La première étape consiste donc à établir le dictionnaire Score en accord avec l'algorithme* `ScoreTopo`*. Pour chaque clause de* Γ*, la table 4.2 indique son "score" (ou nombre de MUS de Muses contenant la clause considérée) ainsi que le nom des MUSes de Muses qui la contienne.*

Ensuite, pour chaque MUS de Muses et pour chaque clause du MUS appartenant à Γ*, nous conservons les clauses du MUS ayant le "score" le plus élevé (vis-à-vis des clauses du même MUS) dans un MUS mus′ puis ajouter ce MUS à l'ensemble de MUSes Muses′.*

Commençons avec le MUS M_1

$$M_1 \cap \Gamma = \{\neg interrupteur\_on \vee \neg interrupteur\_auto, interrupteur\_auto\}$$
$$Score[\neg interrupteur\_on \vee \neg interrupteur\_auto] = 1,\ Score[interrupteur\_auto] = 2$$
$$mus' = \{interrupteur\_auto\}$$
$$Muses' = \{\{interrupteur\_auto\}\}$$

Poursuivons avec le MUS M_2

$$M_2 \cap \Gamma = \{\neg interrupteur\_on \vee piece\_eclairee\}$$
$M_2 \cap \Gamma$ n'étant composé que d'une seule clause
$$mus' = \{\neg interrupteur\_on \vee piece\_eclairee\}$$
$$Muses' = \{\{interrupteur\_auto\}, \{\neg interrupteur\_on \vee piece\_eclairee\}\}$$

Terminons avec le MUS M_3

$$M_3 \cap \Gamma = \{\neg ampoule\_ok \vee \neg interrupteur\_auto \vee piece\_eclairee, interrupteur\_auto\}$$
$$Score[\neg ampoule\_ok \vee \neg interrupteur\_auto \vee piece\_eclairee] = 1,$$
$$Score[interrupteur\_auto] = 2.$$
$$mus' = \{interrupteur\_auto\}$$
Muses′ ne change pas puisque $\{interrupteur\_auto\}$ *appartient déjà à Muses′*
$$Muses' = \{\{interrupteur\_auto\}, \{\neg interrupteur\_on \vee piece\_eclairee\}\}$$

Suite à l'étape de filtrage, les MUSes de Muses′ sont présentés à l'utilisateur afin qu'il puisse choisir quelles clauses retirer pour chaque MUS. Ici, deux MUSes sont présentés à l'utilisateur, composés au total de deux clauses différentes appartenant toutes à Γ*.*

Ici, l'utilisateur n'a pas d'autre choix que de retirer ces deux clauses. En effet, l'étape de filtrage s'étant avérée fructueuse, les MUSes présentés ne sont composés que d'une seule clause. Cela revient à dire et en quelque sorte, que vis-à-vis de notre problème, il n'existe qu'un seul "sous-ensemble maximalement cohérent" de Γ*.*

Ainsi, seules les clauses $\neg interrupteur\_on \vee piece\_eclairee$ *et* $interrupteur\_auto$ *sont retirées de* Γ*. La première d'entre elles implique g directement, son retrait de* Γ *ne peut donc pas être évité. La seconde, quant à elle, implique g en la présence de deux clauses conservée dans* Γ*, de manière indépendante. En retirant cette clause de* Γ *nous permettons donc à deux clauses de* Γ *d'être conservées.*

Finalement, suite à l'opération d'ajout de g, Γ' *est composé des clauses suivantes :*

1. $\neg jour \vee piece\_eclairee$,
2. $\neg ampoule\_ok \vee \neg interrupteur\_auto \vee piece\_eclairee$,
3. $\neg interrupteur\_on \vee \neg interrupteur\_auto$,
4. $\neg interrupteur\_on \vee \neg ampoule\_ok \vee piece\_eclairee$.

Comme escompté, $\Gamma' \models g$ *et* $\Gamma' \not\models f$*, pour tout* f *impliquant strict de* g.

cl $\in \Gamma$	**Score[cl]**	
$\neg interrupteur\_on \vee piece\_eclairee$	1	(M_2)
$\neg jour \vee piece\_eclairee$	0	
$\neg ampoule\_ok \vee \neg interrupteur\_auto \vee piece\_eclairee$	1	(M_3)
$interrupteur\_auto$	2	(M_1, M_3)
$\neg interrupteur\_on \vee \neg interrupteur\_auto$	1	(M_1)

TABLE 4.2 – Dictionnaire *Score* de l'exemple 31.

4.3.3 Cas particulier où Γ est incohérent

Nous avons vu à la section précédente (voir sous-section 4.2.1) que lorsque Γ est incohérent et que l'on souhaite faire préempter une clause g dans Γ, il est nécessaire de rétablir la cohérence de Γ auparavant. Pour ce faire, nous préconisions d'effectuer une première étape de révision de Γ par g à l'aide d'un opérateur sémantique $*$ de révision à la AGM (voir [Alchourrón *et al.* 1985]), approche tout à fait en accord avec notre cadre sémantique et l'objectif final au sujet de g.

Ici, nous proposons les algorithmes `ReviseM`, `ReviseC`, `ReviseMT` et `ReviseCT` (voir Algorithmes 8, 9, 10 et 11, respectivement), lesquels permettent de rétablir la cohérence de Γ tout en y "renforçant" g, à la manière des approches des algorithmes respectifs `PreempteM`, `PreempteC`, `PreempteMT` et `PreempteCT` (voir Algorithmes 1, 2, 4 et 5, respectivement).

Le principe de ces algorithmes est donc d'extraire les sources d'incohérence de $\Gamma \cup \{g\}$ (ligne 2) et de les soumettre à l'utilisateur afin que, pour chacune d'entre elles, il puisse choisir quelles clauses de Γ retirer (lignes 3 à 5 des algorithmes 8 et 9, et, lignes 4 à 6 des algorithmes 10 et 11). Les clauses choisies sont ensuite retirées avant que g y soit ajoutée (lignes 6 et 7 des algorithmes 8 et 9, et, lignes 7 et 8 des algorithmes 10 et 11). Notons que suite à l'extraction des MUSes, une étape de filtrage "topographique" de ceux-ci est effectuée pour les algorithmes 10 et 11 (ligne 3).

4.3.4 Expérimentations

La plateforme, laquelle regroupe entre autres les algorithmes présentés précédemment, a été implémentée en langage `C++` [14]. Les expérimentations ont toutes été conduites sur un pro-

14. Un exécutable du programme accompagné d'explications complémentaires quant à son utilisation est disponible au téléchargement (voir `http://www.cril.fr/~ramon/preempte`).

Données : Γ un ensemble incohérent de clauses et g une clause cohérente et non tautologique
Résultat : Γ' un ensemble cohérent de clauses t.q. $\Gamma' \models g$

début
  $Muses \longleftarrow$ MUSES($\Gamma \cup \{g\}$);
  $SUPR \longleftarrow \emptyset$;
  pour tous les $mus \in Muses$ **faire**
    $SUPR \longleftarrow SUPR \cup$ { clauses de mus appartenant à Γ choisies par l'utilisateur };
  $\Gamma \longleftarrow \Gamma \setminus SUPR$;
  retourner $\Gamma \cup \{g\}$;

Algorithme 8: `ReviseM`.

Données : Γ un ensemble incohérent de clauses et g une clause cohérente et non tautologique
Résultat : Γ' un ensemble cohérent de clauses t.q. $\Gamma' \models g$

début
  $Muses \longleftarrow$ `CouvMuses`($\Gamma \cup \{g\}$);
  $SUPR \longleftarrow \emptyset$;
  pour tous les $mus \in Muses$ **faire**
    $SUPR \longleftarrow SUPR \cup$ { clauses de mus appartenant à Γ choisies par l'utilisateur };
  $\Gamma \longleftarrow \Gamma \setminus SUPR$;
  retourner $\Gamma \cup \{g\}$;

Algorithme 9: `ReviseC`.

Données : Γ un ensemble incohérent de clauses et g une clause cohérente et non tautologique
Résultat : Γ' un ensemble cohérent de clauses t.q. $\Gamma' \models g$

début
  $Muses \longleftarrow$ MUSES($\Gamma \cup \{g\}$);
  $Muses \longleftarrow$ `FiltrageTopo`($Muses, \Gamma$);
  $SUPR \longleftarrow \emptyset$;
  pour tous les $mus \in Muses$ **faire**
    $SUPR \longleftarrow SUPR \cup$ { clauses de mus appartenant à Γ choisies par l'utilisateur };
  $\Gamma \longleftarrow \Gamma \setminus SUPR$;
  retourner $\Gamma \cup \{g\}$;

Algorithme 10: `ReviseMT`.

Données : Γ un ensemble incohérent de clauses et g une clause cohérente et non tautologique
Résultat : Γ' un ensemble cohérent de clauses t.q. $\Gamma' \models g$

début
  $Muses \longleftarrow$ `CouvMuses`$(\Gamma \cup \{g\})$;
  $Muses \longleftarrow$ `FiltrageTopo`$(Muses, \Gamma)$;
  $SUPR \longleftarrow \emptyset$;
  pour tous les $mus \in Muses$ **faire**
    $SUPR \longleftarrow SUPR \cup$ { clauses de mus appartenant à Γ choisies par l'utilisateur };
  $\Gamma \longleftarrow \Gamma \setminus SUPR$;
  retourner $\Gamma \cup \{g\}$;

Algorithme 11: `ReviseCT`.

cesseur `Intel Core2Quad` de $2,66Ghz$, avec une limite de `RAM` de $4GB$, sous `Linux Ubuntu 11.04 (noyau 2.6.38-8) generic`.

Pour le calcul exhaustif des MUSes, nous avons fait appel à l'algorithme `HYCAM` de Grégoire, Mazure et Piette (voir [Grégoire *et al.* 2007a] et [Grégoire *et al.* 2006a]). Celui-ci est l'un des algorithmes les plus efficaces pour le calcul exhaustif des MUSes d'un ensemble de clauses booléennes, qui est une hybridation avec une recherche locale de l'algorithme `CAMUS` de Liffiton et Sakallah (voir [Liffiton & Sakallah 2005]) que nous utilisons en mode "résolution d'instances SAT" pour réaliser les différents tests de cohérence. L'efficacité de l'algorithme `HYCAM` sur des instances de test variées et particulièrement des instances issues des dernières compétitions SAT a été démontrée à plusieurs reprises (voir [Grégoire *et al.* 2009a] et [Grégoire *et al.* 2009b]).

Pour le calcul d'un seul MUS, nous utilisons l'algorithme `OMUS` de Grégoire, Mazure et Piette (voir [Grégoire *et al.* 2006b] et [Grégoire *et al.* 2007b]). Ce dernier approxime un MUS dans un premier temps à l'aide d'une recherche locale, puis le minimise.

L'ensemble de clauses Γ et la clause g devant préempter dans Γ sont fournis en entrée des différents programmes ainsi développés sous la forme de deux fichiers distincts au format `CNF DIMACS` [15]. L'ensemble de clauses Γ' résultant de l'opération de préemption est fourni en sortie des programmes sous la forme, lui aussi, d'un fichier au format `CNF DIMACS`.

4.3.4.1 Instances de test structurées

Premièrement, nous avons concentré notre attention sur des instances de test structurées issues de diverses compétitions SAT [16], dans le but de vérifier la faisabilité de notre approche. Pour cela nous avons considéré le cas particulier où Γ est incohérent (voir sous-section 4.3.3), qui nécessite d'extraire les différentes sources d'incohérence de $\Gamma \cup \{g\}$ pour rétablir la cohérence de Γ tout en "renforçant" g.

Nous avons donc évalué les algorithmes `ReviseM`, `ReviseC`, `ReviseMT` et `ReviseCT` sur

15. Voir `ftp://dimacs.rutgers.edu/pub/challenge/satisfiability/` pour de plus amples détails sur le format `CNF DIMACS`.
16. Voir `http://www.satcompetition.org/2011/` pour les instances de l'édition 2011.

différentes instances SAT jouant le rôle de l'ensemble de clauses Γ, tout en choisissant une clause g de manière à ce qu'elle n'interagisse d'aucune manière avec les sources d'incohérence des instances. Un *time out* a été fixé à 250 secondes.

La table 4.3 présente les résultats les plus pertinents de cette expérimentation. Le nom de l'instance est fourni dans la première colonne, suivi par le nombre de clauses (#cla), de variables (#var) et de MUSes (#mus) de l'instance. Les deux dernières colonnes principales fournissent respectivement les résultats expérimentaux pour les approches calculant exhaustivement les MUSes et celles ne calculant qu'une couverture incohérente stricte. Pour chaque approche, le temps d'exécution des méthodes (#sec) est fourni [17], ainsi que le nombre de clauses présentées à l'utilisateur pour chaque méthode (#cl présentées). Pour les approches ne calculant qu'une simple couverture incohérente stricte, le nombre de MUSes de la dite couverture est aussi fourni (#mus).

				Méthodes *Muses*			Méthodes *Couv*			
					#cl présentées Revise				#cl présentées Revise	
instances	#cla	#var	#mus	#sec	M	MT	#mus	#sec	C	CT
barrel2	159	50	27	0.098	99	73	1	0.191	77	77
barrel3	942	275	67765	127	546	435	1	6.707	456	456
aim-100-1_6-no-4	160	100	1	0.084	48	48	1	0.153	48	48
aim-200-1_6-no-2	320	200	2	0.082	81	79	1	0.281	80	80
aim-200-2_0-no-4	400	200	2	0.084	43	41	1	0.432	42	42
dubois29	232	87	1	0.102	232	232	1	0.302	232	232
C168_FW_UT_851	6758	1909	102	1.798	30	6	1	47.120	8	8
C170_FR_RZ_32	4067	1659	32768	18.640	243	212	1	9.871	227	227
C202_FW_RZ_57	7434	1799	1	1.414	213	213	1	14.955	213	213
C220_FV_RZ_12	4017	1728	80272	5.057	56	3	1	21.607	11	11
C220_FV_SZ_65	4014	1728	103442	8.953	103	18	1	8.758	23	23

TABLE 4.3 – Extrait des résultats expérimentaux sur des instances structurées.

Sans entrer dans les détails, le but de cette expérimentation étant uniquement de montrer la faisabilité de notre approche, nous pouvons remarquer les points suivants. Premièrement, les couvertures incohérentes strictes extraites ne contiennent toutes qu'un seul MUS, laissant penser que les MUSes des instances partagent un nombre très important d'intersections non-vides. Deuxièmement, en terme du nombre de clauses présentées à l'utilisateur, la méthode `ReviseMT` se montre la plus efficace, talonnée de près par les méthodes qui extraient une couverture incohérente stricte, alors que la méthode `ReviseM` est quant à elle, de loin la moins efficace. Pour finir, notons qu'aucune conclusion tangible ne peut être faite concernant l'efficacité des différentes méthodes, les unes par rapport aux autres, en terme de temps d'exécution sur ces instances.

Cette expérimentation a été menée sur des instances de test académiques (par exemple `barrel`, `aim` et `dubois`) et industrielles (par exemple `C168`, `C170`, `C202` et `C220` codant des problèmes de configuration de produits automobiles). Celles-ci son variées tant sur leur taille

17. Notons que le temps d'exécution de l'étape de filtrage "topographique" des méthodes `ReviseMT` et `ReviseCT` étant tout à fait négligeable, le temps d'exécution de ces méthodes est considéré comme identique à celui des méthodes respectives `ReviseM` et `ReviseC`.

(le nombre de clauses allant de 159 à 7434 et le nombre de variables de 50 à 1909) que sur le nombre de MUSes qu'elles possèdent (d'un seul MUS à 103 442 MUSes). Notons toutefois, que le nombre de clauses présentées à l'utilisateur n'excède pas les 546, montrant bien que notre approche n'est faisable que pour des instances dont la taille globale des MUSes à extraire reste très modeste.

4.3.4.2 Instances de test générées

Dans le cas général, la plupart des instances de test issues des compétitions SAT s'avèrent être expérimentalement intraitables pour notre approche. En effet, ces instances jugées "difficiles" ont été proposées uniquement dans le but d'évaluer les solveurs SAT, la plupart d'entre elles contiennent un nombre important de MUSes de très grandes tailles. Pourtant, dans les systèmes d'aide à la décision pouvant avoir recours aux méthodes de notre plateforme, une nouvelle information ne contredit souvent qu'une partie minime de l'information déjà présente, menant alors à un nombre limité de chaînes de raisonnement menant à l'incohérence, lesquelles sont souvent de taille modeste [18]. De manière évidente, les instances de test des compétitions SAT ne suivent que très rarement cette intuition.

Aussi, il est important de noter qu'un lien fort entre l'instance jouant le rôle de Γ et celle jouant celui de g doit exister. En effet, les situations nous intéressant sont celles où Γ subsume g, lesquelles nécessitent l'extraction des MUSes des différents $\Gamma \cup \{\neg(g \Rightarrow f)\}$. Là encore, l'utilisation d'instances SAT ne serait pas chose facile.

Ce sont ces diverses raisons qui nous ont amenés à signer notre propre *générateur d'instances* [19], lequel permet de générer une instance jouant le rôle de Γ et une autre jouant celui de g, de manière à ce que l'opération de préemption de g dans Γ exhibe des MUSes en nombre réduit et ayant une taille modeste. Nous faisons aussi en sorte que ces MUSes partagent une intersection non vide, montrant que souvent les sources d'incohérence d'un problème sont corrélées ; et que le nombre de clauses de Γ qui appartiennent à au moins un MUS soit modeste par rapport au nombre total de clauses de Γ, afin de coller à l'intuition dictant qu'une nouvelle information ne contredit qu'une infime partie du système.

Notre générateur permet donc de générer des instances de test tenant compte de ces contraintes inhérentes à notre problème, en faisant varier les paramètres suivants :

- n : le nombre total de MUSes à extraire,
- m : la taille de ces n MUSes (en nombre de clauses),
- tg : la taille de g (en nombre de littéraux),
- p : le pourcentage de clauses de Γ qui n'appartiennent à aucun MUS,

de manière à ce que :

- tout MUS soit constitué des $tg - 1$ clauses représentant les négations des littéraux d'une plus grande sous-clause stricte f de g [20], de f (pour créer la contradiction), et, d'une

18. Pour exemple, on peut voir l'ensemble de clauses de notre exemple canonique, comme traitant des moyens permettant à une pièce particulière d'être éclairée, comme un sous-ensemble d'informations d'un système d'aide à la décision d'une "Maison intelligente" dans lequel notre nouvelle information devant préempter n'est intuitivement pas subsumée par toute autre information du système ne traitant pas de cette pièce particulière.

19. Notre générateur d'instance fait partie intégrante de la plateforme dont l'exécutable est mis à disposition au téléchargement (voir http://www.cril.fr/~ramon/preempte).

20. En prenant soin de considérer toutes les plus grandes sous-clauses strictes f de g un même nombre de fois

clause permettant de faire en sorte que chaque MUS (sauf 1 si n est impair) partage cette clause avec un autre MUS,
- cette clause est constituée d'un seul littéral,
- les MUSes possèdent des clauses de taille 1 à $tg - 1$ (la plupart étant de taille 2),
- les clauses qui n'appartiennent à aucun MUS sont de taille tg,
- les variables (choisies aléatoirement) de ces clauses n'appartiennent à aucune clause appartenant à un MUS et constituent $p\%$ des variables de Γ.

Nous avons donc évalué les algorithmes `PreempteM`, `PreempteMT`, `PreempteC` et `PreempteCT` sur des instances issues de notre générateur en faisant varier n de 1 à 50, m de 5 à 200, tg de 3 à 49 et p de 80 à 99. Un *time out* a été fixé à 250 secondes.

La table 4.4 présente les résultats les plus pertinents de cette expérimentation. Dans la première colonne principale, les différents paramètres de l'instance générée sont fournis, à savoir le nombre total (n) de MUSes à extraire des différents $\Gamma \cup \neg(g \Rightarrow f)$, la taille (m) de ces MUSes, la taille (tg) de la clause g et le pourcentage (p) de clauses de Γ qui n'appartiennent à aucun MUS. La deuxième colonne principale indique le nombre de clauses (#cla) et de variables (#var) de l'instance représentant Γ. Les deux dernières colonnes principales fournissent respectivement les résultats expérimentaux pour les approches calculant exhaustivement les MUSes et celles ne calculant qu'une couverture incohérente stricte. Pour chaque approche, le temps d'exécution des méthodes (#sec) est fourni [21], ainsi que le nombre de clauses présentées à l'utilisateur pour chaque méthode (#cl présentées). Pour les approches ne calculant qu'une simple couverture incohérente stricte, le nombre de MUSes de la dite couverture est aussi fourni (#mus).

Comme attendu, les méthodes qui n'extraient qu'une simple couverture incohérente stricte permettent de manipuler des instances plus complexes que les méthodes qui extraient tous les MUSes de manière exhaustive. En effet, `PreempteC` et `PreempteCT` traitent de nombreuses instances pour lesquelles `PreempteM` et `PreempteMT` atteignent le *time out*. Par exemple, le premier *time out* de la table est lié à une instance où Γ contient 300 clauses (dont 80% n'appartiennent à aucun MUS) et 217 variables et pour laquelle la préemption de g (constituée de 3 littéraux) dans Γ nécessite de traiter 20 MUSes (constitués de 5 clauses chacun). `PreempteC` et `PreempteCT` détectent tous deux 14 MUSes en 4 secondes approximativement.

Une conséquence directe de cela est que ces méthodes s'exécutent plus rapidement que celles basées sur l'extraction exhaustive des MUSes. Le diagramme de la figure 4.1 montre que c'est bien le cas quand nous faisons varier le nombre de MUSes des instances considérées de 1 à 50 tout en fixant leur taille à 5, la taille de g à 3 et le pourcentage de clause qui n'appartiennent à aucun MUS à 80 (voir la première partie de la table). Nous retrouvons les mêmes résultats lorsque nous faisons varier la taille des MUSes de 10 à 200 clauses tout en fixant leur nombre à 5, la taille de g à 3 et le pourcentage de clauses qui n'appartiennent à aucun MUS à 80 (voir la deuxième partie de la table). Même résultats encore lorsque nous faisons varier la taille de g de 3 à 49 littéraux tout en fixant le nombre de MUSes à 5, leur taille à 10 (puis 20, 30, 40

quand cela est possible.

21. Notons ici aussi que le temps d'exécution de l'étape de filtrage "topographique" des méthodes `PreempteMT` et `PreempteCT` étant tout à fait négligeable, le temps d'exécution de ces méthodes est considéré comme identique à celui des méthodes respectives `PreempteM` et `PreempteC`.

instances				Γ		Méthodes *Muses*			Méthodes *Couv*			
							#cl présentées Preempte				#cl présentées Preempte	
n	m	tg	p	#cla	#var	#sec	M	MT	#mus	#sec	C	CT
1	5	3	80	15	26	0.067	3	3	1	0.076	3	3
2				30	37	0.159	6	1	2	0.208	6	1
3				45	46	0.188	9	4	3	0.285	9	4
4				60	57	0.253	12	2	3	0.376	10	5
5				75	66	0.267	15	5	4	0.496	13	8
6				90	77	0.502	18	3	5	0.610	16	6
7				105	86	0.751	21	6	6	0.676	19	9
8				120	97	3.154	24	4	6	0.814	20	10
9				135	106	7.246	27	7	7	0.944	23	13
10				150	117	36.509	30	5	7	1.111	24	14
20				300	217	*time out*			14	4.053	48	28
30				450	317	*time out*			21	10.969	72	42
40				600	417	*time out*			27	24.874	94	59
50				750	517	*time out*			34	44.212	118	73
5	10	3	80	200	191	0.596	40	10	4	0.666	33	18
	20			450	441	3.856	90	20		1.610	73	38
	30			700	691	25.590	140	30		3.128	113	58
	40			950	941	*time out*				5.277	153	78
	50			1200	1191	*time out*				8.958	193	98
	100			2450	2441	*time out*				43.682	393	198
	150			3700	3691	*time out*				123.118	593	298
	200			4950	4941	*time out*				239.120	793	398
5	10	3	80	200	191	0.602	40	10	4	0.680	33	18
		4		185	181	1.113	37	9		0.857	31	17
		5		170	171	1.225	34	8		1.013	29	16
		6		155	161	1.337	31	7		1.171	27	15
		7		140	151	1.442	28	6		1.442	25	14
		8		125	141	1.543	25	5		1.543	23	13
		9		110	131	1.656	22	4		1.656	21	12
	20	19		210	281	19.063	42	4		5.506	41	22
	30	29		310	431	*time out*				12.886	61	32
	40	39		410	581	*time out*				24.367	81	42
	50	49		510	731	*time out*				41.378	101	52
5	10	3	81	200	191	0.593	40	10	4	0.655	33	18
			83	200	191	0.605				0.680		
			85	240	229	0.618				0.732		
			87	280	267	0.600				0.819		
			89	360	343	0.623				0.944		
			91	440	419	0.617				1.121		
			93	560	533	0.630				1.390		
			95	800	761	0.661				2.048		
			97	1320	1255	0.733				4.094		
			99	4000	3801	1.392				22.587		

TABLE 4.4 – Extrait des résultats expérimentaux sur des instances générées.

et 50) [22] et le pourcentage de clauses qui n'appartiennent à aucun MUS à 80 (voir la troisième partie de la table).

Pourtant, lorsque nous faisons varier le pourcentage de clauses qui n'appartiennent à aucun

22. Notons qu'il est nécessaire que la taille de g soit toujours strictement inférieure à la taille des MUSes, un MUS étant constitué de $tg - 1$ clauses représentant les négations de chaque littéraux d'une plus grande sous-clause stricte f de g et de 2 autres clauses.

MUS de 81 à 99 tout en fixant le nombre de MUS à 5, leur taille à 10 et la taille de g à 3 (voir quatrième partie de la table), nous ne retrouvons plus ces résultats. En effet, comme le montre le diagramme de la figure 4.2, dans ces configurations, ce sont les méthodes qui calculent exhaustivement les MUSes qui s'exécutent plus rapidement que celles ne calculant qu'une couverture incohérente stricte. Cela est vraisemblablement lié au manque de réutilisation de l'information par ces méthodes qui calculent les MUSes successivement, ce qui s'avère être un inconvénient lorsque le nombre de MUSes et leurs tailles ne sont pas élevées et que dans le même temps la taille de la base est très importante. Dans pareille situation, les sources d'incohérence sont comme "noyées" dans l'énorme ensemble de clauses de départ.

Notons aussi, que comme le montre très clairement le diagramme de la figure 4.3, en ce qui concerne le nombre de clauses présentées à l'utilisateur, la méthode `PreempteMT` se montre la plus efficace, talonnée de près par la méthode `PreempteCT`. Les méthodes `PreempteM` et `PreempteC`, dans cet ordre, sont elles de loin les moins efficaces sur ce critère. Ce résultat n'est pas surprenant puisqu'en extrayant tous les MUSes, le "filtrage topographique" est plus efficace que lorsque l'on extrait uniquement une couverture incohérente stricte. L'explication est que le nombre de MUSes étant plus important dans la première approche, le nombre potentiel d'intersections non-vides est plus important lui aussi. De la même manière, en se passant de "filtrage topographique", l'extraction d'une couverture incohérente stricte n'ayant pas besoin d'extraire tous les MUSes, permet de présenter moins de clauses à l'utilisateur que lorsque l'on extrait exhaustivement les MUSes.

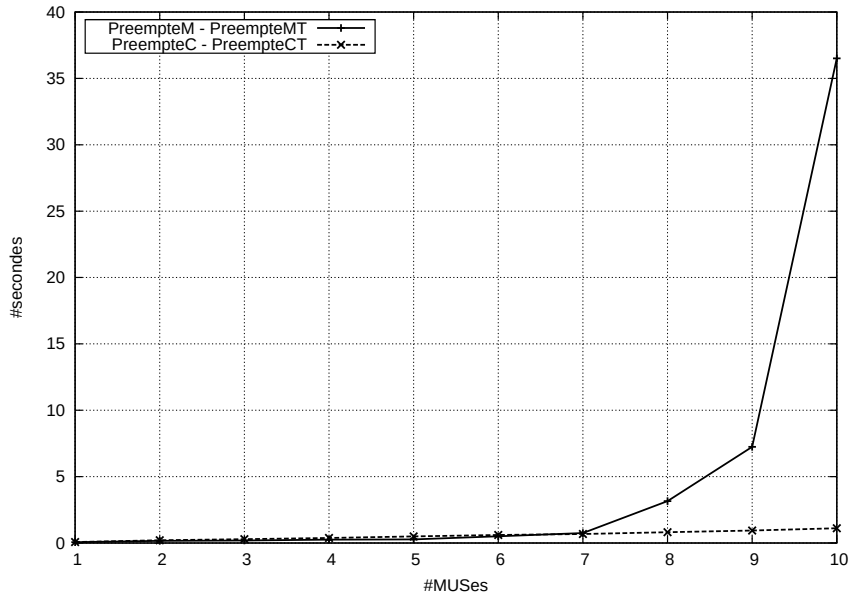

FIGURE 4.1 – Temps d'exécution des méthodes selon n.

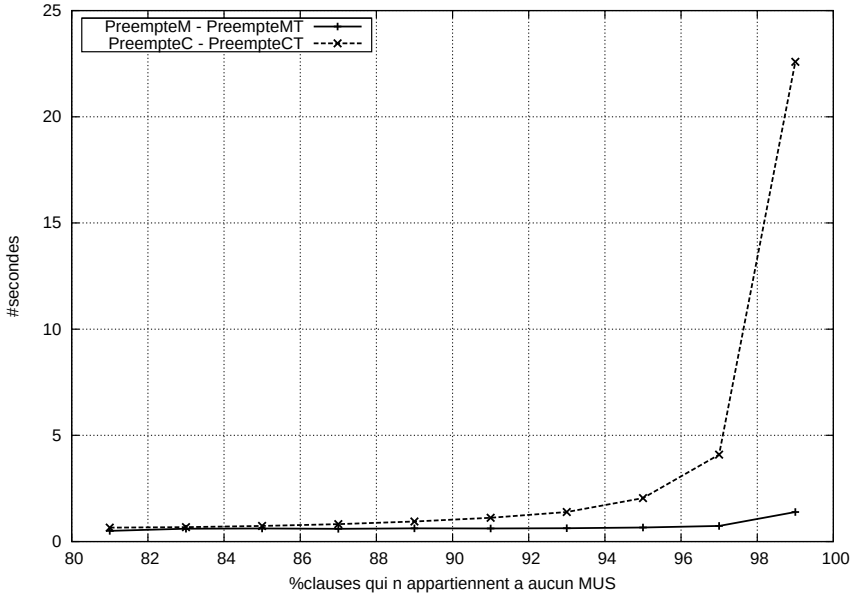

FIGURE 4.2 – Temps d'exécution des méthodes selon p.

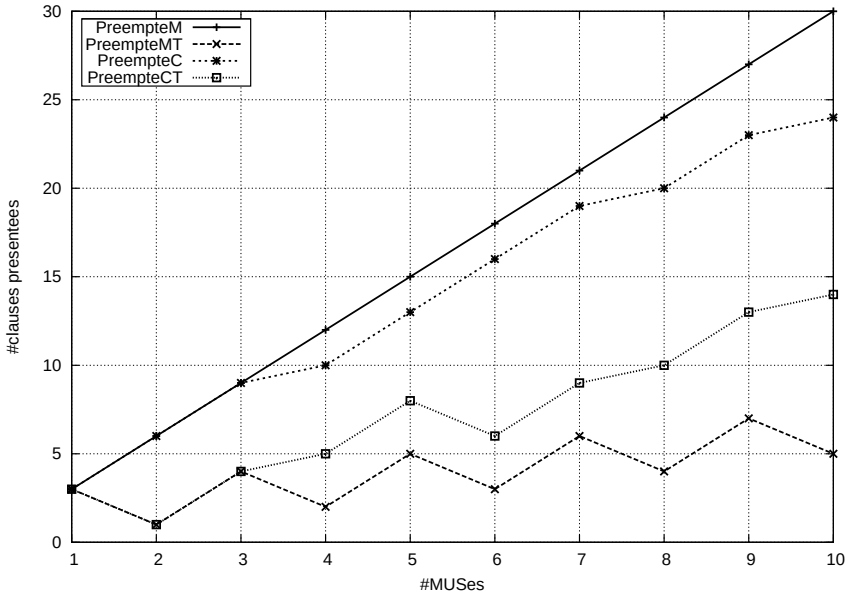

FIGURE 4.3 – Nombre de clauses présentées par les méthodes selon n.

4.4 Dans la littérature

De manière assez surprenante, ce problème n'a pas reçu une attention importante de la communauté scientifique en I.A.. Citons tout de même Grégoire, qui, dans des travaux récents, a proposé une solution au problème du renforcement de règles logiquement plus faibles dans un cadre spécifique non monotone orienté diagnostic (voir [Grégoire 2002], [Grégoire 2003]), ne considérant qu'un cas très particulier de préemption uniquement ; et une ébauche de formalisation de la préemption de formules logiquement plus faibles en utilisant un schéma double de révision de croyances (voir [Grégoire 2007]). Les travaux de ce chapitre sont totalement différents et couvrent le fragment clausal entier de la logique booléenne.

Sur certains points, le problème qui est manipulé dans ce chapitre partage des similarités avec le problème connu sous le nom de "raffinement de connaissances", problème qui a été l'objet de nombreux travaux en *machine learning* (voir [Richards & Mooney 1995] et [Wrobel 1996], travaux faisant école). En raffinement de connaissances, l'objectif est de raffiner une règle de la logique du premier ordre en présence d'*exemples* et/ou de *contre-exemples*. Le problème que nous considérons dans ce chapitre est très clairement différent puisque nous considérons comme nouvelles informations des formules et pas seulement des *faits*. La principale différence est bien plus fondamentale. En effet, supposons que nous essayions de raffiner Γ à la lumière de g, où g est interprété comme un exemple ou un contre-exemple de Γ. Considérons que g encode la disjonction "La personne que j'ai vue était Pierre, Georges ou André" et que Γ soit en mesure de conclure que la disjonction "La personne que j'ai vue était Pierre ou Georges". L'*exemple* g ne contredit pas Γ. De plus, d'un point de vue logique, g n'apporte pas d'information complémentaire en accord avec Γ puisque Γ est déjà en mesure de conclure g. Ainsi, d'un point de vue déductif, g n'apporte pas d'information complémentaire qui permette de raffiner Γ en conséquence. Nous pensons que les travaux présentés dans ce chapitre peuvent être une première étape menant à la définition d'un cadre logique d'apprentissage par l'exemple qui puisse permettre à une information logiquement plus faible de raffiner une connaissance logiquement plus forte.

4.5 Conclusion & perspectives

Dans ce chapitre, nous avons proposé une solution au problème pouvant se produire lorsque, lors de l'insertion dans un ensemble de connaissances de la logique classique, une information doit préempter les informations qui permettent son inférence. Notre solution générale est basée sur une utilisation fine d'une famille d'opérateurs $\ominus$ de contraction, laquelle prend soin d'inhiber tous les cheminements de raisonnement permettant de déduire g et particulièrement ceux s'*activant* lors de son insertion. En restreignant notre étude au cadre propositionnel clausal, nous avons été jusqu'à proposer une solution algorithmique du problème basée sur la manipulation simple de plus grandes sous-clauses, bénéficiant alors d'outils récents performants tant pour le problème de vérification SAT que celui de l'extraction de MUSes.

Il est immédiat de transformer l'approche proposée dans ce chapitre pour assurer la préemption partielle d'une clause g au sens où g devrait pouvoir être déduit tandis que *certains* impliquants premiers de g (modulo l'équivalence logique) ne pourraient plus l'être. Pour ce

faire, il suffit d'appliquer les opérateurs $\oplus$ ou $\oplus'$ restreints à (la version clausale de) ces impliquants. Ceci pourrait s'avérer utile lorsque l'on souhaite introduire une règle "plus forte" tout en voulant empêcher la capacité de déduire une ou plusieurs règles "plus faibles" spécifiques.

Bien que les définitions et propriétés de ce chapitre ne soient pas limitées au seul fragment clausal mais se placent dans le cadre booléen entier, dériver un calcul pour manipuler la préemption de manière pratique dans le cadre booléen entier reste un exercice à réaliser.

Aussi, il serait naturel d'étendre ce cadre de travail au cas de la logique du premier ordre, tenant compte alors de quantification. Dans le chapitre suivant, nous étendons le cadre représentatif de ce chapitre aux logiques non monotones et en particulier à celles permettant la représentation de règles révisables, règles bien connues pour leur expression de cas d'exceptions qui peuvent être exprimées en utilisant des tests de cohérence.

Chapitre 5

Prédominance dans le cadre de logiques non monotones

Sommaire

Au chapitre précédent, nous avons proposé une solution au problème de l'insertion dans un ensemble de connaissances d'une information qui doit préempter celles permettant son inférence, en nous plaçant dans le cadre propositionnel standard et plus particulièrement dans son fragment clausal.

Ici, nous proposons d'étendre le cadre expressif à celui de logiques non monotones, lesquelles sont dotées d'une plus grande expressivité et permettent la modélisation d'une plus large palette de raisonnements (voir Chapitre 2 pour plus de détails). Nous considérerons plus particulièrement les logiques permettant une représentation de règles révisables avec exceptions qui peuvent être exprimées par des tests de cohérence et montrons à cette occasion que l'utilisation d'un formalisme non monotone ne résout pas le problème rencontré dans le cadre propositionnel standard.

Reprenons notre exemple canonique. Considérons qu'un ensemble de connaissances contienne une règle $\mathcal{R}'$ exprimant que "Si l'interrupteur est enclenché et s'il est cohérent de penser que l'ampoule n'est pas cassée, alors la pièce est éclairée". Intuitivement, l'insertion d'une règle $\mathcal{R}$ exprimant que "Si l'interrupteur est enclenché et s'il est à la fois cohérent de penser que l'ampoule n'est pas cassée et que le disjoncteur n'est pas en panne, alors la pièce est éclairée" doit venir inhiber la règle $\mathcal{R}'$ puisqu'il ne suffit plus que l'interrupteur soit enclenché et qu'il soit cohérent de penser que l'ampoule n'est pas cassée pour pouvoir conclure que la pièce est

éclairée ; il doit aussi être cohérent de penser que le disjoncteur n'est pas en panne. Comme dans le cadre propositionnel standard, la règle $\mathcal{R}$ ne peut par elle-même préempter la règle $\mathcal{R}'$ et on ne peut donc pas espérer inhiber cette dernière sans qu'une intervention dans l'ensemble de connaissances ne soit réalisée avant l'ajout de $\mathcal{R}$.

Techniquement, le problème peut être décrit de la manière suivante. Soit un ensemble Γ de connaissances et une règle $\mathcal{R}$. Comment transformer Γ en Γ' pour que Γ' puisse inférer $\mathcal{R}$ mais qu'aucun $\mathcal{R}'$ qui entraîne $\mathcal{R}$ ne puisse en être inféré ? Il est clair que la résolution de ce problème passe d'abord par la définition des éléments qui le composent. Premièrement, la syntaxe des règles considérées (dans laquelle $\mathcal{R}, \mathcal{R}', \ldots$ sont exprimées) doit être introduite. Deuxièmement, une relation d'inférence permettant de manipuler ce genre de règles doit être définie. Troisièmement, un concept d'impliquant permettant d'exprimer ce que l'expression "$\mathcal{R}'$ entraîne $\mathcal{R}$" signifie doit être proposé. Ceci avant même qu'une approche permettant de résoudre ce problème de préemption puisse être discutée.

Le chapitre est organisé comme suit. Dans la section 5.1, un formalisme général pour la représentation de règles avec exceptions est introduit dans le but d'englober différentes approches logiques permettant la manipulation de telles règles, comme le raisonnement par défaut. Dans la section 5.2, différents outils d'inférence permettant de raisonner à partir de ces règles, en ayant la possibilité de considérer des hypothèses additionnelles, sont présentés. La section 5.3, quant à elle, est consacrée à l'étude d'un concept d'impliquant pour des règles avec exceptions. Dans la section 5.4, nous présentons notre solution au problème de la préemption pour ce cadre, en nous basant sur les notions définies précédemment. Pour conclure, dans la section 5.5 nous discutons des perspectives de travaux futurs qui peuvent être envisagées.

Une grande partie des travaux de ce chapitre ont fait l'objet d'une communication en langue anglaise (voir [Besnard *et al.* 2011b]).

5.1 Règles avec exceptions

Si la manière la plus directe de représenter des connaissances en logique passe par leur encodage au sein de formules du langage de la logique classique, certaines connaissances nécessitent cependant un encodage plus élaboré pour permettre un raisonnement révisable. Ainsi, dans ce chapitre nous nous intéressons à des règles d'inférence sujettes à exceptions dont l'absence peut être supposée par des tests de cohérence et que nous voulons manipuler au même titre que les informations encodées à l'aide de formules du langage de la logique classique.

5.1.1 Règles de défaut

L'un des outils les plus populaires permettant de manipuler des règles avec exceptions soumises à des tests de cohérence est sans aucun doute la *logique des défauts* initiée par Reiter (voir [Reiter 1980]). Les règles avec exceptions de cette logique sont appelées *règles de défauts* et permettent à un système d'inférer des conclusions par défaut et de pouvoir les rétracter à la lueur de nouvelles informations montrant que ces conclusions mènent maintenant à l'incohérence (voir Chapitre 2, Section 2.2 pour plus de détails).

Pour reprendre notre exemple canonique, l'information "Si l'interrupteur est enclenché et s'il est à la fois cohérent de penser que l'ampoule n'est pas cassée et que le disjoncteur n'est pas en panne alors la pièce est éclairée" peut se coder très naturellement sous la forme de la règle de défaut suivante en logique des défauts :

$$\frac{interrupteur\_on : ampoule\_ok, disjoncteur\_ok}{piece\_eclairee}$$

Intuitivement, cette règle de défaut est censée permettre un raisonnement du type : "À condition que le fait que l'interrupteur soit enclenché puisse être inféré et à condition que les faits énonçant respectivement que l'ampoule n'est pas cassée et que le disjoncteur n'est pas en panne soient cohérents avec ce qui peut être inféré, inférer le fait que la pièce est éclairée". Notons que la notion d'inférence à laquelle il est fait allusion est basée sur la logique classique dans la définition de Reiter (voir [Reiter 1980]), mais qu'il est tout à fait possible de considérer une autre logique, comme une *logique paraconsistante* par exemple (voir [Pequeno & Buchsbaum 1991]).

Comme nous avons pu le voir au chapitre 2 (voir Section 2.3), en circonscription (voir [McCarthy 1980] et [McCarthy 1986]), il est également possible de représenter cette information sous la forme suivante :

$$interrupteur\_on \wedge \neg anormal \Rightarrow piece\_eclairee.$$

La règle introduit pour ce faire un prédicat d'anormalité, lequel est déclenché par les exceptions suivantes :

$$\neg ampoule\_ok \Rightarrow anormal,$$
$$\neg disjoncteur\_ok \Rightarrow anormal.$$

5.1.2 Règles PEC

En accord avec les représentations précédentes, l'information de notre exemple canonique "Si l'interrupteur est enclenché et s'il est à la fois cohérent de penser que l'ampoule n'est pas cassée et que le disjoncteur n'est pas en panne alors la pièce est éclairée", se représente naturellement comme composée de trois éléments : ses *prémisses* ("L'interrupteur est enclenché"), ses *exceptions* ("L'ampoule est cassée" et "Le disjoncteur est en panne") et sa *conclusion* ("La pièce est éclairée").

Notre objectif est donc de traiter de manière uniforme les connaissances représentées à l'aide de simples formules (comme peuvent l'être par exemple les faits du type "L'interrupteur est enclenché"), les règles non révisables (comme "Si l'interrupteur est enclenché alors la pièce est éclairée") et les règles d'inférence avec exceptions soumises à des tests de cohérence (comme la règle de défaut plus haut). Ceci dans un cadre suffisamment générique et donc totalement indépendant de la logique des défauts par exemple, de manière à pouvoir instancier notre approche aussi à d'autres logiques.

À cet effet, nous introduisons le concept de règle PEC (pour Prémisses-Exceptions-Conclusions), pour un langage donné, comme celui de triplets constitués de trois ensembles de formules permettant de représenter des règles d'inférence qui peuvent souffrir d'exceptions.

Quand les prémisses sont les conditions nécessaires d'application de la règle, les exceptions sont elles basées sur des tests de cohérence. Les conclusions, quant à elles, listent les affirmations qui peuvent être faites à condition que la règle s'applique. Une formule classique pourra se représenter comme une règle PEC dont les prémisses sont $\{\top\}$ et dont l'ensemble d'exceptions est vide. Les formules implicatives pourront se représenter de cette manière, ou de manière équivalente, en distinguant un antécédent d'un conséquent, représentés respectivement par les prémisses et les conclusions de la règle.

Définition 91 (règle PEC)**.** *Une règle PEC est un triplet $\mathcal{R} = (\mathcal{P}, \mathcal{E}, C)$ où $\mathcal{P} = \{\rho_1, \ldots, \rho_k\}$ et $C = \{\varsigma_1, \ldots, \varsigma_n\}$ sont des ensembles cohérents de formules et où $\mathcal{E} = \{\epsilon_1, \ldots, \epsilon_m\}$ est un ensemble de formules non tautologiques.*

Chose importante, nous n'imposons aucune contrainte sur le langage : il peut, ou non, contenir la négation, la conjonction, la disjonction ou tout autre connecteur. Il peut aussi n'être pourvu d'aucun connecteur. Cependant, une relation d'inférence implicite $\Vdash$ doit être mise à disposition. Bien sûr, ceci signifie que le formalisme logique utilisé doit posséder une forme de tautologie (notons bien la subtilité ici : ceci ne veux pas dire que le formalisme logique utilisé doit posséder des tautologies). Une règle PEC peut donc être sujette à différentes interprétations selon la manière dont les ensembles de prémisses et de conclusions sont logiquement considérés (principalement, de manière conjonctive ou disjonctive). Dans la suite, nous ne considérerons que des règles PEC unaires, c'est-à-dire des règles PEC dont l'ensemble des prémisses et des conclusions sont des singletons. Très naturellement, toute règle PEC peut se transformer en une règle PEC unaire en considérant la conjonction (ou la disjonction, selon l'option considérée) de ses prémisses et de ses conclusions. Par abus de notation et pour faciliter la lecture, nous omettrons systématiquement de préciser qu'il s'agit d'ensembles et dénoterons les ensembles de prémisses et la conclusion par leurs uniques formules, en omettant les accolades de ces singletons.

Définition 92 (règle PEC unaire)**.** *Une règle PEC $\mathcal{R} = (\mathcal{P}, \mathcal{E}, C)$ est unaire ssi $\mathcal{P}$ et C sont des singletons.*

Dans la suite, nous omettrons systématiquement le qualificatif "unaire" et considérerons logiquement de manière disjonctive l'ensemble d'exceptions.

Exemple 32. *La règle PEC $(interrupteur\_on, \{\neg ampoule\_ok\}, piece\_eclairee)$ est un encodage de la règle avec exception "Si l'interrupteur est enclenché et s'il est cohérent de penser que l'ampoule n'est pas cassée, alors la pièce est éclairée" de notre exemple canonique.*

Exemple 33. *La règle PEC $(interrupteur\_on, \{\neg ampoule\_ok, \neg disjoncteur\_ok\}, piece\_eclairee)$ est un encodage de la règle avec exception "Si l'interrupteur est enclenché et s'il est à la fois cohérent de penser que l'ampoule n'est pas cassée et que le disjoncteur n'est pas en panne, alors la pièce est éclairée" de notre exemple canonique.*

Exemple 34. *La règle PEC $(interrupteur\_on \wedge ampoule\_ok, \emptyset, piece\_eclairee)$ est un encodage d'une règle similaire à celle de l'exemple 32 où l'impossibilité de dériver e.g. ampoule_ok peut bloquer l'inférence de piece_eclairee. De plus, notons que $\neg ampoule\_ok$ serait une exception de la règle, qui n'est cependant pas inclue par convention dans l'ensemble des exceptions de la règle PEC, puisque ce n'est pas une exception basée sur la cohérence.*

Exemple 35. *La règle PEC* $(\top, \emptyset, interrupteur\_on)$ *est un encodage du fait "L'interrupteur est enclenché" de notre exemple canonique.*

Exemple 36. *Les règles PEC* $(\top, \emptyset, interrupteur\_on \Rightarrow piece\_eclairee)$ *et* $(interrupteur\_on, \emptyset, piece\_eclairee)$ *sont deux encodages possibles de la règle non révisable "Si l'interrupteur est enclenché alors la pièce est éclairée" de notre exemple canonique. Notons que dans le deuxième encodage, nous distinguons l'antécédent du conséquent.*

Comme le montrent les exemples précédents, les exceptions qui peuvent s'exprimer à l'aide de la logique classique ne sont pas inclues dans l'ensemble des exceptions d'une règle PEC, lequel est réservé aux exceptions basées sur des tests de cohérence. En ce qui concerne les formules, nous choisirons souvent de les représenter comme des règles PEC dont les ensembles de prémisses et d'exceptions sont vides ; plus précisément, nous les représenterons au sein de règles PEC unaires dont les ensembles de prémisses est un singleton tautologique et dont les ensembles d'exceptions sont vides. Il en résulte que la relation d'inférence $\Vdash$ est considérée comme admettant $\top$ pour représenter en réalité certaines formules. L'existence de différents encodages possibles de connaissances entre prémisses et conclusions est similaire par exemple à la différence bien connue en logique des défauts entre des défauts avec pré-requis et les défauts libres de pré-requis leur correspondant (voir [Brass 1993]).

Les règles PEC forment ainsi un outil de représentation très général qui offre l'avantage de représenter dans un cadre unifié des formules et des règles avec exceptions, tout en conservant le moyen de distinguer clairement les prémisses, les exceptions et les conclusion de ces règles. De fait, nous prendrons plus loin parfois le parti de prendre en compte un certain biais syntaxique en considérant par exemple des règles PEC de conclusions logiques (déductivement) équivalentes, en se refusant notamment une intervention possible avec changement de signe entre prémisses et conclusions par *modus tollens*. [1]

Par la suite, nous nous limiterons à ne considérer que le cadre propositionnel standard et omettrons de le rappeler.

5.2 Raisonner avec et à propos de règles PEC

Après avoir défini, à la section précédente, le concept de règles PEC qui permet de représenter de manière uniforme à la fois de simples formules, des règles non révisables et des règles d'inférence avec exceptions soumises à des tests de cohérence, notre problème technique peut maintenant s'écrire comme suit. Soit Γ un ensemble de règles PEC et $\mathcal{R}$ une règle PEC devant préempter dans Γ. Comment transformer Γ en Γ' de telle manière à ce que Γ' puisse inférer $\mathcal{R}$ mais qu'aucune règle PEC $\mathcal{R}'$ qui entraîne $\mathcal{R}$ ne puisse en être inférée ?

Dans cette section, nous définissons maintenant différents outils d'inférence permettant de raisonner avec et à propos de règles PEC : ces outils ne permettent donc pas uniquement de manipuler des règles PEC, mais ils permettent aussi d'inférer des règles PEC, le tout dans un même cadre.

1. Dans une transposition en règles PEC des formules de la logique classique, par application de la règle de *modus tollens* les règles PEC $(interrupteur\_on, \emptyset, piece\_eclairee)$ et $(\neg piece\_eclairee, \emptyset, \neg interrupteur\_on)$ seraient équivalentes.

5.2.1 Dérivations

Dans un premier temps, nous définissons un concept de "dérivation" pour le langage très général des règles PEC, qui permet de dériver une règle PEC à partir d'un ensemble de règles PEC, en ayant la possibilité de "se placer" dans les conditions d'application de la règle.

Un mot d'avertissement : Dans la suite, $\vdash$ ne représente pas une relation d'inférence. Dans la réalité, $\vdash \alpha$ (resp. $\nvdash \alpha$) signifie que α (ou un équivalent logique élémentaire) possède (resp. n'a pas) le statut "inféré" *au sein de la dérivation*. Aussi, "non inféré au sein de la dérivation" ne signifie pas "dont la forme négative ne peut être inférée à l'aide des formules inférées occurrant dans la dérivation" (notion plus faible, clairement inintéressante).

Un mot de terminologie : $\vdash \alpha$ et $\nvdash \alpha$ sont vues comme des formules signées. Naturellement, $\vdash \alpha$ (resp. $\nvdash \alpha$) est dit positif (resp. négatif).

Définition 93 (dérivation)**.** *Soient* Γ *un ensemble de règles PEC et* $\aleph = (\rho, \{\epsilon_1, \ldots, \epsilon_n\}, \varsigma)$ *une règle PEC. Une dérivation de* $\aleph$ *à partir de* Γ *est un arbre* T *dont les nœuds sont des formules classiques signées t.q. :*

1. *pour toute feuille de la forme* $\vdash \alpha$*,*
   - *soit* $(\top, \emptyset, \alpha) \in \Gamma$*,*
   - *soit* $(\alpha_1, \emptyset, \alpha_2) \in \Gamma$ *et* $\alpha = \alpha_1 \Rightarrow \alpha_2$*,*
   - *soit* $\alpha = \rho$*,*
2. *pour toute feuille de la forme* $\nvdash \beta$*,*
   $\beta \notin Cn(\{\gamma \text{ t.q. } (\top, \emptyset, \gamma) \in \Gamma\} \cup \{\gamma_1 \Rightarrow \gamma_2 \text{ t.q. } (\gamma_1, \emptyset, \gamma_2) \in \Gamma\} \cup \{\alpha \text{ t.q. } \vdash \alpha \text{ est un nœud de } T\})$*,*
3. *si* $\nvdash \beta$ *est un nœud alors c'est une feuille,*
4. *tout nœud qui n'est pas une feuille a comme parents,*
   - *soit un tuple* $(\vdash \alpha_1, \ldots, \vdash \alpha_k)$*,*
   - *soit un tuple* $(\vdash \alpha_1, \nvdash \beta_1, \ldots, \nvdash \beta_m)$ *où* $m \geq 1$*,*
5. *si* $\vdash \alpha$ *est un nœud dont les parents sont un tuple* $(\vdash \alpha_1, \ldots, \vdash \alpha_k)$*,*
   $\alpha \in Cn(\{\alpha_1, \ldots, \alpha_k\})$*,*
6. *si* $\vdash \alpha$ *est un nœud dont les parents sont un tuple* $(\vdash \alpha_1, \nvdash \beta_1, \ldots, \nvdash \beta_m)$ *où* $m \geq 1$*,*
   $(\alpha_1, \{\beta_1, \ldots, \beta_m\}, \alpha) \in \Gamma$*,*
7. $\rho \in \{\alpha \text{ t.q. } \vdash \alpha \text{ est une feuille de } T\} \cup \{\top\}$*,*
   $\{\epsilon_1, \ldots, \epsilon_n\} = \{\beta \text{ t.q. } \nvdash \beta \text{ est un nœud de } T\}$*,*
   $\vdash \varsigma$ *est la racine de* T*.*

On note $\Gamma \mathrel{|\!\sim}^{\{\epsilon_1, \ldots, \epsilon_n\}} \aleph$ *et, lorsque* $\{\epsilon_1, \ldots, \epsilon_n\}$ *est vide,* $\Gamma \mathrel{|\!\sim} \aleph$*.*

Intuitions et exemples dans le cadre classique

Présentons les intuitions et exemples permettant d'appréhender cette définition. Tout d'abord, analysons ce qu'elle devient dans un cadre pouvant se réduire au cadre propositionnel standard, c'est-à-dire en l'absence de règles PEC munies d'ensembles non vides d'exceptions. L'arbre sera dépourvu de nœuds négatifs et par conséquent les items 2, 3 et 6 ne s'appliquent plus et les items 4 et 7 se simplifient.

L'item 1 définit une feuille positive de l'arbre comme étant soit une formule classique encodée dans une règle PEC de Γ sous la forme classique (1^{er} item) ou sous la forme antécédent/conséquent pour les formules implicatives (2^{ime} item), ou la prémisse ρ de la règle PEC $\aleph$ qui est la règle PEC dérivée (3^{ime} item). Les "prémisses" de la dérivation, que sont les feuilles positives de l'arbre, peuvent donc être des formules classiques de Γ ou la prémisse de $\aleph$. Notons qu'il est donc possible de "se placer" dans les conditions d'application de la règle que l'on dérive. L'item 4, se simplifie en *tout nœud qui n'est pas une feuille a comme parents un tuple* ($\vdash \alpha_1, \ldots, \vdash \alpha_k$) et montre donc que seules des dérivations classiques, dont la définition est donnée dans l'item 5, sont permises.

L'arbre se réduit donc à un arbre de dérivation de la logique propositionnelle : il ne contient que des nœuds positifs et ne fait usage que de règles classiques de déduction. L'item 7, exprime qu'un tel arbre permet de conclure déductivement de Γ des règles $\aleph = (\rho, \emptyset, \varsigma)$ où ς est la racine de l'arbre et où ρ est soit :

- $\top$ (voir la règle $(\top, \emptyset, c)$ dérivée, Exemple 37),
- une formule encodée dans une règle de Γ [2] (voir a, $a \Rightarrow b$ et $b \Rightarrow c$, Exemple 37),
- ou une formule additionnelle qui joue le rôle d'hypothèse additionnelle (voir a dans l'exemple 38, $a \Rightarrow b$ dans l'exemple 39 et $\neg a$ dans l'exemple 40).

Il est important de noter qu'un arbre de dérivation peut contenir des nœuds qui contredisent des formules classiques encodées sous la forme de règles PEC appartenant à Γ (voir a dans l'exemple 39). Cela exprime que, dans le cas où aucune exception n'est mentionnée, il est possible de raisonner à partir d'un ensemble de règles PEC, en ne considérant qu'une partie de cet ensemble. Toujours dans le cas où aucune exception n'est mentionnée, un phénomène classique de trivialisation risque naturellement de voir le jour (voir Exemple 40).

Pour résumer, en se restreignant au cadre propositionnel standard, une dérivation permet de raisonner classiquement à partir d'un ensemble de règles PEC pour inférer une règle PEC, en ayant la possibilité de se placer dans les conditions d'application de celle-ci et en ne considérant qu'une partie des règles PEC de Γ.

Exemple 37. *Soit* $\Gamma = \{(a, \emptyset, b), (\top, \emptyset, a), (b, \emptyset, c)\}$ *un ensemble de règles PEC.*

L'arbre de la figure 5.1 est une dérivation de $(\top, \emptyset, c)$ *à partir de* Γ. *Remarquons que la première partie de l'item 7 est satisfaite par* $\rho \in \{\top\}$.

Notons que cet arbre est aussi une dérivation de $(a, \emptyset, c)$, *de* $(a \Rightarrow b, \emptyset, c)$ *et de* $(b \Rightarrow c, \emptyset, c)$ *à partir de* Γ. *Ici,* ρ *joue le rôle d'une formule encodée dans une règle de* Γ.

$$\dfrac{\dfrac{\vdash a \quad \vdash a \Rightarrow b}{\vdash b} \qquad \vdash b \Rightarrow c}{\vdash c}$$

FIGURE 5.1 – Arbre des exemples 37, 38 et 39.

Exemple 38. *Soit* $\Gamma = \{(a, \emptyset, b), (b, \emptyset, c)\}$ *un ensemble de règles PEC.*

2. Notons que si ρ est une formule encodée dans une règle de Γ, ρ est donc "inutile" dans le sens où il est déjà déductible de Γ, en remplaçant ρ par $\top$ la dérivation resterait valide.

L'arbre de la figure 5.1 est une dérivation de $(a, \emptyset, c)$ *à partir de* Γ. *Ici* ρ *joue le rôle d'une hypothèse additionnelle.*

Exemple 39. *Soit* $\Gamma = \{(\top, \emptyset, a), (b, \emptyset, c)\}$ *un ensemble de règles PEC.*

L'arbre de la figure 5.1 est une dérivation de $(a \Rightarrow b, \emptyset, c)$ *à partir de* Γ. *Ici,* ρ *joue le rôle d'un hypothèse additionnelle.*

Notons que cet arbre est aussi une dérivation de $(a \Rightarrow b, \emptyset, c)$ *à partir de* $\Gamma \cup \{(\top, \emptyset, \neg a)\}$, *bien que le nœud* $\vdash a$ *de l'arbre soit incohérent avec la formule classique* $\neg a$ *encodée dans la règle PEC* $(\top, \emptyset, \neg a)$ *ajoutée à* Γ.

Exemple 40. *Soit* $\Gamma = \{(\top, \emptyset, a)\}$ *un ensemble de règles PEC.*

L'arbre de la figure 5.2 est une dérivation de $(\neg a, \emptyset, c)$ *à partir de* Γ. *Ici,* ρ *joue le rôle d'un hypothèse additionnelle.*

$$\dfrac{\vdash \neg a \quad \vdash a}{\vdash c}$$

FIGURE 5.2 – Arbre de l'exemple 40.

Intuitions et exemples dans le cadre général

En présence de règles PEC dans Γ qui possèdent un ensemble d'exceptions non vide, l'arbre de dérivation peut traduire un raisonnement sous certaines conditions (en somme, sous réserve de possibles exceptions), comme le montre l'exemple 41. Dans ce cas général, l'arbre sera pourvu de nœuds négatifs et par conséquent les items 2, 3 et 6 s'appliquent et les items 4 et 7 retrouvent leur forme générale.

L'item 2 assure un raisonnement cohérent au sens où ni les informations sans exceptions dans Γ, ni les nœuds positifs de l'arbre n'entraînent d'exception dont l'absence est requise dans le cheminement de raisonnement (voir Exemple 42, *Cn* étant une relation de conséquence de la logique classique et Exemple 43, *Cn* étant une relation de conséquence d'une logique arbitraire). L'item 3, stipule qu'*a contrario*, les tests de cohérence apparaissent comme hypothèses, ils ne peuvent être inférés : notre notion de dérivation n'est pas dédiée au raisonnement sur les exceptions (voir $\nvdash d \wedge e$ dans l'exemple 44). L'item 4, indique que la déduction classique et l'application d'une règle révisable avec exceptions sont les seuls moyens de dérivation autorisés. L'item 6 décrit précisément comment l'application d'une règle révisable avec exceptions est régie (naturellement, dans ce cas au moins un nœud négatif doit intervenir). L'item 7 précise notamment que les exceptions de la règle dérivée traduisent exactement toutes les hypothèses de cohérence utilisées dans l'arbre.

Il est important de noter qu'un arbre de dérivation qui contient des nœuds négatifs ne peut contenir des nœuds qui contredisent des formules classiques encodées sous la forme de règles PEC appartenant à Γ (voir $\neg f$ dans l'exemple 42). Cela exprime ici que, dans le cas où au moins une exception est mentionnée, il est impossible de raisonner à partir d'un ensemble de règles PEC en ne considérant qu'une partie de ses règles traduisant des formules classiques.

Pour résumer, dans le cas général, une dérivation permet à la fois de raisonner classiquement et d'appliquer des règles révisables avec exceptions, à partir d'un ensemble de règles PEC pour inférer une règle PEC, en ayant la possibilité de se placer dans les conditions d'application de celle-ci.

Exemple 41. *Soit* $\Gamma = \{(a \wedge b, \{d, e\}, f), (f, \emptyset, c)\}$ *un ensemble de règles PEC.*

L'arbre de la figure 5.3 est une dérivation de $(a \wedge b, \{d, e\}, c)$ *à partir de* Γ. *Ici,* ρ *joue le rôle d'une hypothèse additionnelle et* $\{d, e\}$ *regroupe tous les nœuds négatifs de l'arbre.*

Cet arbre n'est pas une dérivation de $(a \wedge b, \{d, e, g\}, c)$ *ni de* $(a \wedge b, \{d\}, c)$ *à partir de* Γ. *Ici, l'item 7 de la définition 93 n'est pas respecté puisque, dans le premier cas,* g *est listée comme appartenant à l'ensemble des exceptions de la règle dérivée mais* $\nvdash g$ *n'est pas un nœud de l'arbre et dans le deuxième cas,* $\nvdash e$ *est un nœud de l'arbre mais n'apparaît pas dans l'ensemble d'exceptions de la règle dérivée.*

Cet arbre n'est pas non plus une dérivation de $(a, \{d, e\}, c)$ *ni de* $(a \wedge b \wedge g, \{d, e\}, c)$ *à partir de* Γ. *Ici, l'item 7 n'est pas respecté pour une raison différente : dans les deux cas* ρ *n'appartient pas aux feuilles positives de l'arbre.*

$$\dfrac{\dfrac{\vdash a \wedge b \quad \nvdash d \quad \nvdash e}{\vdash f} \qquad \vdash f \Rightarrow c}{\vdash c}$$

Figure 5.3 – Arbre de l'exemple 41.

Exemple 42. *Soit* $\Gamma = \{(a \wedge b, \{d, e\}, f), (f, \emptyset, c), (\top, \emptyset, \neg f)\}$ *un ensemble de règles PEC.*

Si Cn est considérée comme une relation classique, l'arbre de la figure 5.3 n'est pas une dérivation de $(a \wedge b, \{d, e\}, c)$ *à partir de* Γ. *La raison est que l'item 2 de la définition 93 n'est pas respecté comme démontré ci-après. Premièrement,* $\vdash f$ *est un nœud de l'arbre de dérivation donc* $f \in \{\alpha \text{ t.q. } \vdash \alpha \text{ est un nœud de } T\}$. *Deuxièmement,* $(\top, \emptyset, \neg f)$ *appartient à* Γ *donc* $\neg f \in \{\gamma \text{ t.q. } (\top, \emptyset, \gamma) \in \Gamma\}$. *Troisièmement, l'item 2 devient alors* $\beta \notin Cn(\{f, \ldots, \top \Rightarrow \neg f\})$ *qui doit être vérifié pour* β *valant d puis e. Cependant, comme Cn est une relation classique,* $Cn(\{f, \ldots, \top \Rightarrow \neg f\})$ *contient toutes les formules du langage, dont d et e.*

Exemple 43. *Soit* $\Gamma = \{(a \wedge b, \{d, e\}, f), (f, \emptyset, d), (d, \{\neg c\}, c)\}$ *un ensemble de règles PEC.*

L'arbre de la figure 5.4 n'est pas une dérivation de $(a \wedge b, \{d, e, \neg c\}, c)$ *à partir de* Γ. *Ici, l'item 2 de la définition 93 n'est pas respecté comme démontré ci-après. Premièrement,* $\vdash d$ *est un nœud de l'arbre de dérivation donc* $d \in \{\alpha \text{ t.q. } \vdash \alpha \text{ est un nœud de } T\}$. *Comme* $\nvdash d$ *est une feuille,* $\beta \notin Cn(\{d, \ldots\})$ *doit être vérifié pour* β *valant d et l'impossibilité est évidente.*

Exemple 44. *Soit* $\Gamma = \{(a \wedge b, \{d, e\}, f), (f, \emptyset, c)\}$ *un ensemble de règles PEC.*

L'arbre de la figure 5.5 n'est pas une dérivation de $(a \wedge b, \{d \wedge e\}, c)$ *à partir de* Γ. *Ici, l'item 3 de la définition 93 n'est pas respecté puisque les nœuds négatifs* $\nvdash d$ *et* $\nvdash e$ *sont "dérivés" de la feuille négative* $\nvdash d \wedge e$.

$$\dfrac{\dfrac{\vdash a \wedge b \quad \nvdash d \quad \nvdash e}{\vdash f} \qquad \vdash f \Rightarrow d}{\vdash d} \qquad \nvdash \neg c}{\vdash c}$$

FIGURE 5.4 – Arbre de l'exemple 43.

$$\dfrac{\vdash a \wedge b \qquad \dfrac{\nvdash d \wedge e}{\nvdash d} \qquad \dfrac{\nvdash d \wedge e}{\nvdash e}}{\vdash f} \qquad \vdash f \Rightarrow c}{\vdash c}$$

FIGURE 5.5 – Arbre de l'exemple 44.

Une approche polyvalente

Il importe à ce stade de préciser que l'objectif de ce travail n'est pas de définir une nouvelle logique non monotone et sa théorie de la preuve, mais de définir un canevas expressivement suffisant et général pour traiter du problème de la préemption d'une information subsumée lors de son insertion dans un ensemble de connaissances, pour différentes logiques. À cet égard, le concept de dérivation n'est qu'un outil qui peut devoir être contraint pour pouvoir s'identifier à un outil de preuve dans certaines logiques. Effectivement, le prix de la généralité de notre cadre de représentation est la possible nécessité d'implanter des contraintes supplémentaires pour l'instanciation à certaines logiques.

Par exemple, le concept de dérivation ne s'identifie pas immédiatement à celui habituel de démonstration en logique des défauts de Reiter (voir [Reiter 1980]), aux sens suivants. Nous n'avons pas défini de concept d'extension et contrairement à la logique des défauts qui requiert que "tout défaut pouvant s'appliquer s'applique" dans la preuve qu'une formule appartient à une extension, la traduction immédiate du concept de dérivation dans le langage de la logique des défauts de Reiter n'impose pas cette contrainte. Pour montrer cela, il est nécessaire avant tout de définir le concept de traduction d'une théorie avec défauts sous la forme d'un ensemble de règles PEC correspondant.

Définition 94 (traduction d'une théorie avec défauts)**.** *Soit* $\Gamma = (\Delta, \Sigma)$ *une théorie avec défauts. Un ensemble* $\Gamma^\star$ *de règles PEC est une traduction de* Γ *ssi :*

1. *pour tout* $\gamma \in \Sigma$ *:*
   - *soit* $(\top, \emptyset, \gamma) \in \Gamma^\star$,
   - *soit* $\gamma = \gamma_1 \Rightarrow \gamma_2$ *et* $(\gamma_1, \emptyset, \gamma_2) \in \Gamma^\star$,
2. *pour tout* $\dfrac{\alpha : \neg\beta_1, \ldots, \neg\beta_m}{\gamma} \in \Delta$, $(\alpha, \{\beta_1, \ldots, \beta_m\}, \gamma) \in \Gamma^\star$,
3. *il n'existe pas d'autres règles PEC dans* $\Gamma^\star$ *que celles mentionnées en 1 et 2.*

À l'aide de cette définition, l'exemple suivant montre qu'une théorie avec défauts peut ne posséder aucune extension (comme nous avions pu le voir au chapitre 2, Section 2.2) alors qu'il existe au moins une dérivation à partir de l'ensemble de règles PEC traduisant la théorie.

Exemple 45. *Soient une théorie avec défauts* $\Gamma = (\Delta, \Sigma)$ *où* $\Delta = \{\frac{\top:a}{\neg a}, \frac{b:d,e}{c}\}$ *et* $\Sigma = \{b\}$*, et,* $\Gamma^\star = \{(\top, \{\neg a\}, \neg a), (b, \{\neg d, \neg e\}, c), (\top, \emptyset, b)\}$ *un ensemble de règles PEC traduisant* Γ*.*

Dû à la présence du défaut $\frac{\top:a}{\neg a}$ *dans* Δ*,* Γ *ne possède aucune extension. Pourtant il existe bien une dérivation de la règle PEC* $(\top, \{\neg d, \neg e\}, c)$ *à partir de* $\Gamma^\star$*, comme le montre l'arbre de la figure 5.6.*

$$\frac{\vdash b \quad \nvdash \neg d \quad \nvdash \neg e}{\vdash c}$$

Figure 5.6 – Arbre de l'exemple 45.

De la même manière, comme le montre cet autre exemple, il se peut que la conclusion d'une dérivation à partir d'un ensemble $\Gamma^\star$ de règles PEC ne fasse partie d'aucune extension d'une théorie avec défauts Γ dont $\Gamma^\star$ est la traduction.

Exemple 46. *Soient une théorie avec défauts* $\Gamma = (\Delta, \Sigma)$ *où* $\Delta = \{\frac{\top:a}{b}, \frac{a:\neg b}{c}\}$ *et* $\Sigma = \{a\}$*, et,* $\Gamma^\star = \{(\top, \{\neg a\}, b), (a, \{b\}, c), (\top, \emptyset, a)\}$ *un ensemble de règles PEC traduisant* Γ*.*

Γ *possède l'unique extension* $E = Cn(\{a, b\})$ *qui ne contient pas la formule* c*. Pourtant il existe bien une dérivation de la règle PEC* $(\top, \{b\}, c)$ *à partir de* $\Gamma^\star$*, comme le montre l'arbre de la figure 5.7.*

$$\frac{\vdash a \quad \nvdash b}{\vdash c}$$

Figure 5.7 – Arbre de l'exemple 46.

En revanche, le formalisme de dérivation est assez puissant pour couvrir l'inférence crédule en logique des défauts de Reiter, comme le montre la propriété immédiate suivante.

Propriété 10. *Soient* $\Gamma = (\Delta, \Sigma)$ *une théorie avec défauts et* $\Gamma^\star$ *un ensemble de règles PEC traduisant* Γ*. Pour toute formule* f *appartenant à une extension* E *de* Γ*, il existe une dérivation de* $(\top, \{\epsilon_1, \ldots, \epsilon_n\}, f)$ *à partir de* $\Gamma^\star$ *où* $\{\neg\epsilon_1, \ldots, \neg\epsilon_n\}$ *est un sous-ensemble des justificatifs des défauts générateurs de* Γ *pour* E*.*

Il est important de noter que les dérivations ne sont pas forcément des preuves optimales : il n'y a pas d'effort comme celui d'éviter les détours ou celui d'imposer des raccourcis.

Soulignons qu'un concept de cohérence peut être introduit pour notre cadre de travail.

Définition 95 (cohérence)**.** Γ *est cohérent si et seulement si* $\Gamma \nvdash (\top, \emptyset, \bot)$*.*

Comme à l'accoutumée, une notion de cohérence ouvre la question du choix des négations. Le choix d'une négation $\sim$ pour des règles PEC, comme étant telle qu'à la fois $\Gamma \vdash R$ et $\Gamma \vdash \sim R$ est possible, sans que $\Gamma \vdash R\& \sim R$ (où & représente une "conjonction" de règles PEC). Intentionnellement, nous avons laissé en dehors de ce document toute notion de fermeture déductive et de la même manière tout sous-ensemble de conséquences comme on le ferait, e.g. à la manière des extensions *à la* logique des défauts.

5.2.2 X-dérivations

La suite de notre travail va nécessiter de considérer un concept de "X-dérivation", basé sur celui de dérivation venant d'être défini (voir Définition 93). Ce concept permet de dériver une règle PEC à partir d'un ensemble de règles PEC, en ayant non seulement la possibilité de "se placer" dans les conditions d'application de la règle dérivée, mais en permettant aussi de considérer une règle PEC additionnelle X comme faisant en quelque sorte partie de Γ.

Définition 96 (X-dérivation)**.** *Soient Γ un ensemble de règles PEC et X une règle PEC. Une X-dérivation de $\aleph = (\rho, \{\epsilon_1, \ldots, \epsilon_n\}, \varsigma)$ à partir de Γ est un arbre T dont les nœuds sont des formules classiques signées t.q. :*

1. *pour toute feuille de la forme $\vdash \alpha$,*
   - *soit $(\top, \emptyset, \alpha) \in \Gamma \cup \{X\}$,*
   - *soit $(\alpha_1, \emptyset, \alpha_2) \in \Gamma \cup \{X\}$ et $\alpha = \alpha_1 \Rightarrow \alpha_2$,*
   - *soit $\alpha = \rho$,*
2. *pour toute feuille de la forme $\nvdash \beta$,*
   $\beta \notin Cn(\{\gamma$ t.q. $(\top, \emptyset, \gamma) \in \Gamma \cup \{X\}\} \cup \{\gamma_1 \Rightarrow \gamma_2$ t.q. $(\gamma_1, \emptyset, \gamma_2) \in \Gamma \cup \{X\}\} \cup \{\alpha$ t.q. $\vdash \alpha$ est un nœud de $T\})$,
3. *si $\nvdash \beta$ est un nœud alors c'est une feuille,*
4. *tout nœud qui n'est pas une feuille a comme parents,*
   - *soit un tuple $(\vdash \alpha_1, \ldots, \vdash \alpha_k)$,*
   - *soit un tuple $(\vdash \alpha_1, \nvdash \beta_1, \ldots, \nvdash \beta_m)$ où $m \geq 1$,*
5. *si $\vdash \alpha$ est un nœud dont les parents sont un tuple $(\vdash \alpha_1, \ldots, \vdash \alpha_k)$,*
   $\alpha \in Cn(\{\alpha_1, \ldots, \alpha_k\})$,
6. *si $\vdash \alpha$ est un nœud dont les parents sont un tuple $(\vdash \alpha_1, \nvdash \beta_1, \ldots, \nvdash \beta_m)$ où $m \geq 1$,*
   $(\alpha_1, \{\beta_1, \ldots, \beta_m\}, \alpha) \in \Gamma \cup \{X\}$,
7. *$\rho \in \{\alpha$ t.q. $\vdash \alpha$ est une feuille de $T\} \cup \{\top\}$,*
   $\{\epsilon_1, \ldots, \epsilon_n\} = \{\beta$ t.q. $\nvdash \beta$ est un nœud de $T\}$ et
   $\vdash \varsigma$ est la racine de T.

On note $\Gamma \mathrel{\overset{\{\epsilon_1,\ldots,\epsilon_n\}}{\underset{X}{\sim}} } \aleph$ et $\Gamma \mathrel{\overset{}{\underset{X}{\sim}}} \aleph$ quand $\{\epsilon_1, \ldots, \epsilon_n\}$ est vide.

Analysons les modifications que cette définition apporte à celle de dérivation. L'hypothèse additionnelle X intervient principalement au niveau des items 1, 2 et 6.

L'item 1 indique maintenant que les "prémisses" de la dérivation que sont les feuilles positives de l'arbre, en plus de pouvoir être une formule classique de Γ ou la prémisse de $\aleph$, peuvent être une hypothèse supplémentaire traduite au sein de la règle additionnelle X quand celle-ci se réduit à une formule classique (voir $a \wedge b$, Exemple 47). La présence de X dans l'item 2, exprime que lorsque X représente une formule classique, tout comme les informations sans exceptions dans Γ et les nœuds positifs de l'arbre, il ne doit pas entraîner d'exception dont l'absence est requise dans le cheminement de raisonnement. L'item 6, décrivant quant à lui comment l'application d'une règle révisable avec exceptions est régie, indique que la règle additionnelle X peut être appliquée comme toute règle PEC de Γ possédant un ensemble

d'exceptions non vide. Quand X est une règle PEC révisable avec exceptions, sa prémisse, soit résulte d'un raisonnement en amont dans l'arbre (voir f, Exemple 48), soit est la prémisse ρ de la règle PEC dérivée (voir $a \wedge b$, Exemple 49).

Notons que lorsque $X = (\top, \emptyset, \top)$, une X-dérivation de $\aleph$ à partir de Γ est une dérivation de $\aleph$ à partir de Γ (voir Exemple 50.).

Notons aussi, que lorsqu'il existe une dérivation de $\mathcal{R}$ à partir de Γ, pour certaines règles PEC $\mathcal{R}'$ il existe une $\mathcal{R}'$-dérivation de $\mathcal{R}$ à partir de Γ, sans pour autant que $\mathcal{R}'$ ne joue un rôle dans la dérivation (voir Exemple 51).

Pour résumer, une X-dérivation permet un raisonnement hypothétique à plusieurs niveaux : elle manipule des hypothèses révisables de cohérence, elle peut inclure une règle additionnelle X et supposer comme vraie la prémisse de $\aleph$ (la règle PEC devant être inférée).

Exemple 47. *Soit $\Gamma = \{(a \wedge b, \{d, e\}, f), (f, \emptyset, c)\}$ un ensemble de règles PEC.*

L'arbre de la figure 5.8 est une $(\top, \emptyset, a \wedge b)$-dérivation de $(\top, \{d, e\}, c)$ à partir de Γ. Ici, X joue le rôle d'une formule classique additionnelle encodée sous la forme d'une règle PEC. Notons que le prémisse de X est $\top$.

Notons que cet arbre est aussi une $(\top, \emptyset, a \wedge b)$-dérivation de $(a \wedge b, \{d, e\}, c)$ à partir de Γ. Dans cette X-dérivation X est redondant puisque déjà encodée dans la prémisse ρ de la règle dérivée.

$$\dfrac{\dfrac{\vdash a \wedge b \quad \nvdash d \quad \nvdash e}{\vdash f} \qquad \vdash f \Rightarrow c}{\vdash c}$$

FIGURE 5.8 – Arbre des exemples 47 et 50.

Exemple 48. *Soit $\Gamma = \{(a \wedge b, \{d, e\}, f)\}$ un ensemble de règles PEC.*

L'arbre de la figure 5.9 est une $(f, \{g\}, c)$-dérivation de $(a \wedge b, \{d, e, g\}, c)$ à partir de Γ. Ici, X joue le rôle d'une hypothèse additionnelle sous la forme d'une règle révisable avec exceptions. Notons que la prémisse de X est une conclusion intermédiaire de l'arbre.

Cet arbre est aussi une $(f, \{g\}, c)$-dérivation de $(a \wedge b, \{d, e, g\}, c)$ à partir de $\Gamma \cup \{(f, \{g\}, c)\}$, bien que de manière inopportune puisque X appartient déjà à Γ.

$$\dfrac{\dfrac{\vdash a \wedge b \quad \nvdash d \quad \nvdash e}{\vdash f} \qquad \nvdash g}{\vdash c}$$

FIGURE 5.9 – Arbre des exemples 48 et 49.

Exemple 49. *Soit $\Gamma = \{(f, \{g\}, c)\}$ un ensemble de règles PEC.*

L'arbre de la figure 5.9 est une $(a \wedge b, \{d, e\}, f)$-dérivation de $(a \wedge b, \{d, e, g\}, c)$ à partir de Γ. Ici, X joue le rôle d'une hypothèse additionnelle sous la forme d'une règle révisable avec exceptions. Notons que la prémisse de X est la prémisse ρ de la régle dérivée.

Exemple 50. *Soit* $\Gamma = \{(a \wedge b, \{d, e\}, f), (f, \emptyset, c)\}$ *un ensemble de règles PEC.*

L'arbre de la figure 5.8 est à la fois une dérivation et une $(\top, \emptyset, \top)$*-dérivation de* $(a \wedge b, \{d, e\}, c)$ *à partir de* Γ.

Exemple 51. *Soit* $\Gamma = \{(a \wedge b, \{d, e\}, f), (f, \emptyset, c)\}$ *un ensemble de règles PEC,* $\mathcal{R} = (a \wedge b, \{d, e\}, c)$ *et* $\mathcal{R}' = (x, \{y\}, z)$ *deux règles PEC.*

L'arbre de la figure 5.10 est une dérivation de $\mathcal{R}$ *à partir de* Γ.

Cet arbre est aussi une $\mathcal{R}'$*-dérivation de* $\mathcal{R}$ *à partir de* Γ. *Notons que* $\mathcal{R}'$ *ne joue aucun rôle dans la dérivation.*

$$\dfrac{\dfrac{\vdash a \wedge b \quad \nvdash d \quad \nvdash e}{\vdash f} \qquad \vdash f \Rightarrow c}{\vdash c}$$

FIGURE 5.10 – Arbre de l'exemple 51.

5.3 Concepts d'impliquants pour des règles PEC

À la section précédente, nous avons défini un outil de dérivation (voir Définition 93) qui permet de raisonner à partir d'un ensemble de règles PEC pour en dériver une règle PEC. Nous pouvons donc maintenant écrire notre problème technique comme suit. Soit Γ un ensemble de règles PEC et $\mathcal{R}$ une règle PEC devant préempter dans Γ. Comment transformer Γ en Γ' de telle manière à ce que $\Gamma' \mathrel{|\!\sim}^{\mathcal{E}} \mathcal{R}$, mais que $\Gamma' \not\mathrel{|\!\sim}^{\mathcal{E}'} \mathcal{R}'$, pour tout $\mathcal{R}'$ qui entraîne $\mathcal{R}$.

Dans cette section, nous définissons un concept d'impliquant pour des règles PEC permettant d'exprimer ce que "$\mathcal{R}'$ entraîne $\mathcal{R}$" signifie.

5.3.1 Impliquants modulo un ensemble de règles PEC

Le problème est ici de savoir si une règle PEC $\mathcal{R}'$ implique une règle PEC $\mathcal{R}$ modulo un ensemble Γ de règles PEC. Pour montrer qu'une règle PEC en implique une autre, ce concept considère donc une relation de conséquence logique modulo un ensemble de règles PEC, à l'instar de la notion bien connue d'*impliquant modulo une théorie* du cadre propositionnel standard (voir Définition 52). Notre outil de X-dérivation présenté à la section précédente (voir Définition 96) est parfaitement adapté à la résolution de ce problème. En effet, celui-ci revient intuitivement à se demander s'il existe une $\mathcal{R}'$-dérivation de $\mathcal{R}$ à partir de Γ.

Définition 97 (impliquant)**.** *Soient* Γ *un ensemble de règles PEC et* $\mathcal{R} = (\rho, \{\epsilon_1, \ldots, \epsilon_m\}, \varsigma)$ *une règle PEC.* $\mathcal{R}'$ *est un impliquant de* $\mathcal{R}$ *modulo* Γ *ssi il existe* $\mathcal{D}$ *une* $\mathcal{R}'$*-dérivation de* (ρ, E^*, ς) *à partir de* Γ *t.q. :*

1. $\forall e' \in E^*, \exists e \in \{\epsilon_1, \ldots, \epsilon_m\}$ *t.q.* $e \in Cn(\{e'\})$,
2. $\forall e'' \in \{\epsilon_1, \ldots, \epsilon_m\} \setminus E^*$, $e'' \notin Cn(\{\alpha$ *t.q.* $\vdash \alpha$ *est un nœud de* $\mathcal{D}\})$,

Définition 98 (impliquant strict)**.** *Soit* Γ *un ensemble de règles PEC. Soient* $\mathcal{R}$ *et* $\mathcal{R}'$ *deux règles PEC.* $\mathcal{R}'$ *est un impliquant strict de* $\mathcal{R}$ *modulo* Γ *ssi* $\mathcal{R}'$ *est un impliquant de* $\mathcal{R}$ *modulo* Γ *et* $\mathcal{R}$ *n'est pas un impliquant de* $\mathcal{R}'$ *modulo* Γ.

Dans la suite, nous considérons que *Cn* représente une relation de conséquence classique.

Intuitions et exemples dans le cadre classique

Présentons les intuitions et exemples permettant d'appréhender ces définitions dans un cadre pouvant se réduire au cadre propositionnel standard, c'est-à-dire en l'absence de règles PEC munies d'ensembles non vides d'exceptions. Dans ce cas, les items 1 et 2 de la définition 97 ne s'appliquent pas et la définition se simplifie en : $\mathcal{R}'$ *est un impliquant de* $\mathcal{R}$ *modulo* Γ *ssi il existe* $\mathcal{D}$ *une* $\mathcal{R}'$*-dérivation de* $\mathcal{R}$ *à partir de* Γ (notée $\Gamma \vdash_{\mathcal{R}'} \mathcal{R}$).

Ainsi, la question de savoir si $\mathcal{R}'$ est un impliquant de $\mathcal{R}$ modulo Γ revient donc à se demander si, en se plaçant dans les conditions d'application ρ de $\mathcal{R}$, l'adjonction de $\mathcal{R}'$ à Γ lui permet de dériver $\mathcal{R}$ (au sens de la définition 93). Cette définition permet donc de se placer dans les conditions d'application ρ de $\mathcal{R}$ pour montrer l'implication (voir $a \wedge b$, Exemples 52, 53 et 54).

Notons que s'il existe une dérivation de $\mathcal{R}$ à partir de Γ (au sens de la définition 93), pour certaines règles PEC $\mathcal{R}' = (\rho', \emptyset, \varsigma')$, $\mathcal{R}'$ est un impliquant de $\mathcal{R}$ sans pour autant que $\mathcal{R}'$ ne joue un rôle dans la dérivation (voir Exemple 54).

Intuitivement, notre concept d'impliquant pour des règles PEC dans le cadre classique correspond à la notion d'impliquant modulo une théorie du cadre propositionnel standard (voir Définition 52). Nous montrons cela dans les exemples qui suivent en nous aidant du concept de traduction d'une théorie du cadre propositionnel standard sous la forme d'un ensemble de règles PEC correspondant, traduction définie ci-après.

Définition 99 (traduction d'une théorie du cadre propositionnel standard)**.** *Soit* K *une théorie du cadre propositionnel standard. Un ensemble de règles PEC* Γ *est une traduction de* K *ssi il existe une bijection entre* K *et* Γ *t.q. pour tout* $\gamma \in K$ *:*

- *soit* $(\top, \emptyset, \gamma) \in \Gamma$,
- *soit* $\gamma = \gamma_1 \Rightarrow \gamma_2$ *et* $(\gamma_1, \emptyset, \gamma_2) \in \Gamma$.

Exemple 52. *Soit* $\Gamma = \{(g \wedge h, \emptyset, c), (\top, \emptyset, h)\}$ *un ensemble de règles PEC. Soient* $\mathcal{R} = (a \wedge b, \emptyset, c)$ *et* $\mathcal{R}' = (a, \emptyset, g)$ *deux règles PEC.*

$\mathcal{R}'$ *est un impliquant de* $\mathcal{R}$ *modulo* Γ*. En effet, l'arbre de la figure 5.11 est une* $\mathcal{R}'$*-dérivation de* $\mathcal{R}$ *à partir de* Γ*. Ici, on se place dans les conditions d'application de* $\mathcal{R}$ *(à savoir* $a \wedge b$*) pour montrer l'implication.*

Soit $K = \{(g \wedge h) \Rightarrow c, h\}$ *une théorie du cadre propositionnel standard t.q.* Γ *soit une traduction de* K*. Soient* $r = (a \wedge b) \Rightarrow c$ *et* $r' = a \Rightarrow g$ *deux formules propositionnelles t.q.* $\mathcal{R}$ *et* $\mathcal{R}'$ *encodent respectivement* r *et* r'*.* r' *est un impliquant de* r *modulo* K*. En effet,* $K \models g \Rightarrow c$ *donc* $K \cup \{a \Rightarrow g\} \models a \Rightarrow c$ *et* $K \cup \{a \Rightarrow g\} \models (a \wedge b) \Rightarrow c$.

Exemple 53. *Soit* $\Gamma = \{(g \wedge h, \emptyset, c), (\top, \emptyset, h)\}$ *un ensemble de règles PEC. Soient* $\mathcal{R} = (a \wedge b, \emptyset, c)$ *et* $\mathcal{R}' = (\top, \emptyset, g)$ *deux règles PEC.*

$$\dfrac{\dfrac{\dfrac{\dfrac{\vdash a \wedge b}{\vdash a} \quad \vdash a \Rightarrow g}{\vdash g} \quad \vdash h}{\vdash g \wedge h} \quad \vdash g \wedge h \Rightarrow c}{\vdash c}$$

FIGURE 5.11 – Arbre de l'exemple 52.

$\mathcal{R}'$ n'est pas un impliquant de $\mathcal{R}$ modulo Γ via le premier arbre de la figure 5.12. En effet, cet arbre n'est pas une $\mathcal{R}'$-dérivation de $\mathcal{R}$ à partir de Γ. Ici, on ne se place pas dans les conditions d'application de $\mathcal{R}$ (à savoir $a \wedge b$) pour montrer l'implication, ainsi $a \wedge b$ n'appartient pas aux nœuds positifs de l'arbre ce qui viole la première partie de l'item 7 de la définition 96. Notons tout de même que $\mathcal{R}'$ est un impliquant (strict) de $(\top, \emptyset, c)$ via cet arbre.

Ceci dit, $\mathcal{R}'$ est un impliquant de $\mathcal{R}$ modulo Γ via le deuxième arbre de la figure 5.12. En effet, cet arbre est une $\mathcal{R}'$-dérivation de $\mathcal{R}$ à partir de Γ. Notons que dans cet arbre on se place dans les conditions d'application de $\mathcal{R}$, bien que celles-ci ne jouent pas un rôle prépondérant dans la dérivation.

Soit $K = \{(g \wedge h) \Rightarrow c, h\}$ une théorie du cadre propositionnel standard t.q. Γ soit une traduction de K. Soient $r = (a \wedge b) \Rightarrow c$ et $r' = g$ deux formules propositionnelles t.q. $\mathcal{R}$ et $\mathcal{R}'$ encodent respectivement r et r'. r' est un impliquant de r modulo K. En effet, $K \cup \{r'\} \models c$ et donc $K \cup \{r'\} \models (a \wedge b) \Rightarrow c$.

$$\dfrac{\dfrac{\vdash g \quad \vdash h}{\vdash g \wedge h} \quad \vdash g \wedge h \Rightarrow c}{\vdash c} \qquad \dfrac{\dfrac{\dfrac{\dfrac{\vdash a \wedge b}{\vdash \top} \quad \vdash g}{\vdash g} \quad \vdash h}{\vdash g \wedge h} \quad \vdash g \wedge h \Rightarrow c}{\vdash c}$$

FIGURE 5.12 – Arbres de l'exemple 53.

Exemple 54. *Soient $\Gamma = \{(a, \emptyset, c)\}$ un ensemble de règles PEC. Soient $\mathcal{R} = (a \wedge b, \emptyset, c)$ et $\mathcal{R}' = (\top, \emptyset, x)$ deux règles PEC.*

$\mathcal{R}'$ est un impliquant de $\mathcal{R}$ modulo Γ. En effet, l'arbre de la figure 5.13 est une $\mathcal{R}'$-dérivation de $\mathcal{R}$ à partir de Γ.

Notons que cet arbre est aussi une dérivation de $\mathcal{R}$ à partir de Γ.

Soit $K = \{a \Rightarrow c\}$ une théorie du cadre propositionnel standard t.q. Γ soit une traduction de K. Soient $r = (a \wedge b) \Rightarrow c$ et $r' = x$ deux formules propositionnelles t.q. $\mathcal{R}$ et $\mathcal{R}'$ encodent respectivement r et r'. r' est un impliquant de r modulo K. En effet, $K \models a \Rightarrow c$ donc $K \models (a \wedge b) \Rightarrow c$ et $K \cup \{x\} \models (a \wedge b) \Rightarrow c$.

Intuitions et exemples dans le cadre général

Nous introduisons ici les clés principales permettant d'appréhender notre définition d'impliquant modulo un ensemble de règles PEC dans le cas général. En présence de règles PEC

$$\dfrac{\dfrac{\vdash a \wedge b}{\vdash a} \quad \vdash a \Rightarrow c}{\vdash c}$$

FIGURE 5.13 – Arbre de l'exemple 54.

qui possèdent un ensemble d'exceptions non vide, les items 1 et 2 de la définition 97 s'appliquent.

Intuitivement, il est nécessaire de tenir compte des liens pouvant exister entre les exceptions de $\mathcal{R}$ et celles de $\mathcal{R}'$. Ces liens doivent faire en sorte que si $\mathcal{R}'$ ne s'applique pas à cause de l'une de ses exceptions alors $\mathcal{R}$ ne s'applique pas non plus. Par contre, si $\mathcal{R}$ ne s'applique pas à cause de l'une des exceptions, il se pourrait quand même que $\mathcal{R}'$ s'applique. Ainsi, d'un côté toute exception de $\mathcal{R}'$ doit être au moins aussi forte qu'au moins une exception de $\mathcal{R}$ (voir $\neg d$, Exemple 55, $\neg d \wedge \neg f$, Exemples 56 et 57) et d'un autre côté, les exceptions apparaissant dans l'arbre de dérivation utilisé pour montrer l'implication et qui ne sont pas impliquées par une exception de $\mathcal{R}'$ doivent elles aussi être au moins aussi forte qu'au moins une exception de $\mathcal{R}$ (voir $\neg e \wedge \neg h$, Exemple 57). Aussi, il est nécessaire de tenir compte des liens pouvant opérer entre les exceptions de $\mathcal{R}$ et les nœuds de l'arbre de dérivation montrant l'implication. L'intuition veut que la démonstration de l'implication (via l'arbre de dérivation) n'entraîne pas un cas d'exception de $\mathcal{R}$ (voir $\neg e$, Exemple 58 et e, Exemple 59).

Ainsi, montrer que $\mathcal{R}'$ est un impliquant de $\mathcal{R} = (\rho, \{\epsilon_1, \ldots, \epsilon_m\}, \varsigma)$ modulo Γ, revient à montrer qu'il existe une $\mathcal{R}'$-dérivation de (ρ, E^*, ς) où E^* est constitué des nœuds négatifs de l'arbre (qu'ils appartiennent ou non aux exceptions de $\mathcal{R}'$), tels que chaque élément de E^* implique une exception de $\mathcal{R}$ (item 1) et et tels qu'aucune exception de $\mathcal{R}$ qui ne figure pas dans E^* soit une conséquence logique des nœuds positifs de l'arbre.

Notons que le contexte est pris en compte quand il s'agit d'évaluer si une règle PEC en subsume une autre dans le sens de la définition 97. En effet, l'impliquant et l'impliqué n'ont pas besoin nécessairement de comporter les mêmes prémisses (voir Exemple 60).

Notons que s'il existe une dérivation de $\mathcal{R}$ à partir de Γ (au sens de la définition 93), alors pour certaines règles PEC $\mathcal{R}'$, $\mathcal{R}'$ est un impliquant de $\mathcal{R}$ modulo Γ, sans pour autant que $\mathcal{R}'$ ne joue un rôle dans la dérivation (voir Exemple 61).

Exemple 55. *Soient Γ un ensemble vide de règles PEC. Soient $\mathcal{R} = (a \wedge b, \{\neg d, \neg e\}, c)$ et $\mathcal{R}' = (a, \{\neg d\}, c)$ deux règles PEC.*

$\mathcal{R}'$ est un impliquant de $\mathcal{R}$ modulo Γ via l'arbre de la figure 5.14. En effet, cet arbre est une $\mathcal{R}'$-dérivation de $(a \wedge b, \{\neg d\}, c)$ à partir de Γ. En accord avec l'item 1, $\neg d$ (exception de $\mathcal{R}'$) appartient aux exceptions de $\mathcal{R}$ et en accord avec l'item 2, $\neg e$ n'est pas déductible de $\{a, a \wedge b, c\}$.

$$\dfrac{\dfrac{\vdash a \wedge b}{\vdash a} \quad \nvdash \neg d}{\vdash c}$$

FIGURE 5.14 – Arbre de l'exemple 55.

Exemple 56. *Soit Γ un ensemble vide de règles PEC. Soient $\mathcal{R} = (a, \{\neg d, \neg e\}, c)$ et $\mathcal{R}' = (a, \{\neg d \wedge \neg f\}, c)$ deux règles PEC.*

$\mathcal{R}'$ est un impliquant de $\mathcal{R}$ modulo Γ via l'arbre de la figure 5.15. En effet, cet arbre est une $\mathcal{R}'$-dérivation de $(a, \{\neg d \wedge \neg f\}, c)$ à partir de Γ. En accord avec l'item 1, $\neg d \wedge \neg f$ (exception de $\mathcal{R}'$) implique $\neg d$ (une exception de $\mathcal{R}$) et en accord avec l'item 2, ni $\neg d$ ni $\neg e$ (les membres de $\{\epsilon_1, \ldots, \epsilon_m\}$) ne sont impliqués par $\{a, c\}$ (les formules des nœuds positifs de l'arbre).

$$\dfrac{\vdash a \quad \nvdash \neg d \wedge \neg f}{\vdash c}$$

FIGURE 5.15 – Arbre de l'exemple 56.

Exemple 57. *Soit $\Gamma = \{(a, \{\neg e \wedge \neg h\}, g)\}$ un ensemble de règles PEC. Soient $\mathcal{R} = (a, \{\neg d, \neg e\}, c)$ et $\mathcal{R}' = (g, \{\neg d \wedge \neg f\}, c)$ deux règles PEC.*

$\mathcal{R}'$ est un impliquant de $\mathcal{R}$ modulo Γ via l'arbre de la figure 5.16. En effet, cet arbre est une $\mathcal{R}'$-dérivation de $(a, \{\neg d \wedge \neg f, \neg e \wedge \neg h\}, c)$ à partir de Γ. En respect avec l'item 1, $\neg d \wedge \neg f$ (exception de $\mathcal{R}'$) implique $\neg d$ (une exception de $\mathcal{R}$) et $\neg e \wedge \neg h$ (exception de la règle PEC de Γ) implique $\neg e$ (une exception de $\mathcal{R}$). En respect avec l'item 2, ni $\neg d$ ni $\neg e$ (les membres de $\{\epsilon_1, \ldots, \epsilon_m\}$) ne sont impliqués par $\{a, c, g\}$ (les formules des nœuds positifs de l'arbre).

$$\dfrac{\dfrac{\vdash a \quad \nvdash \neg e \wedge \neg h}{\vdash g} \qquad \nvdash \neg d \wedge \neg f}{\vdash c}$$

FIGURE 5.16 – Arbre de l'exemple 57.

Exemple 58. *Soit $\Gamma = \{(\top, \emptyset, a \Rightarrow \neg e), (\top, \emptyset, \neg e \Rightarrow f)\}$ un ensemble de règles PEC. Soient $\mathcal{R} = (a \wedge b, \{\neg d, \neg e\}, c)$ et $\mathcal{R}' = (f, \{\neg d\}, c)$ deux règles PEC.*

$\mathcal{R}'$ n'est pas un impliquant de $\mathcal{R}$ modulo Γ via l'arbre de la figure 5.17. En effet, bien que cet arbre soit une $\mathcal{R}'$-dérivation de $(a \wedge b, \{\neg d\}, c)$ à partir de Γ, l'item 2 de la définition 97 n'est pas respecté : $\vdash \neg e$ est un nœud de l'arbre bien que $\neg e$ soit une exception de $\mathcal{R}$. Intuitivement, $\mathcal{R}'$ ne subsume pas $\mathcal{R}$ parce que la manière dont $\mathcal{R}'$ est appliqué pour essayer d'inférer cette version "voisine" de $\mathcal{R}$ implique un cas qui se trouve être une exception de $\mathcal{R}$.

$$\dfrac{\dfrac{\dfrac{\vdash a \wedge b}{\vdash a} \quad \vdash a \Rightarrow \neg e}{\vdash \neg e} \qquad \vdash \neg e \Rightarrow f}{\vdash f} \qquad \nvdash \neg d}{\vdash c}$$

FIGURE 5.17 – Arbre de l'exemple 58.

Exemple 59. *Soit* $\Gamma = \{(a, \{e\}, f)\}$ *un ensemble de règles PEC. Soient* $\mathcal{R} = (a \wedge b, \{\neg d, \neg e\}, c)$ *et* $\mathcal{R}' = (f, \{\neg d\}, c)$ *deux règles PEC.*

$\mathcal{R}'$ n'est pas un impliquant de $\mathcal{R}$ modulo Γ via l'arbre de la figure 5.18. Bien que ce dernier soit une $\mathcal{R}'$-dérivation de $(a \wedge b, \{\neg d, e\}, c)$ à partir de Γ, l'item 1 de la définition d'impliquant n'est pas respecté : e est une formule de E^* à partir de laquelle aucune formule de $\{\neg d, \neg e\}$ ne peut être inférée. Intuitivement, $\mathcal{R}'$ ne subsume pas $\mathcal{R}$ parce que la manière dont il est appliqué pour essayer d'inférer cette version "voisine" de $\mathcal{R}$ introduit une exception de $\mathcal{R}$.

$$\dfrac{\dfrac{\dfrac{\vdash a \wedge b}{\vdash a} \quad \nvdash e}{\vdash f} \quad \nvdash \neg d}{\vdash c}$$

Figure 5.18 – Arbre de l'exemple 59.

Exemple 60. *Soit* Γ *un ensemble vide de règles PEC. Soient* $\mathcal{R} = (a, \{\neg b\}, b)$ *et* $\mathcal{R}' = (\top, \emptyset, a \Rightarrow b)$ *deux règles PEC.*

$\mathcal{R}'$ est un impliquant de $\mathcal{R}$ modulo Γ via l'arbre de la figure 5.19. En effet, cet arbre est une $\mathcal{R}'$-dérivation de $(a, \emptyset, b)$ à partir de Γ. L'item 1 est trivialement vérifié puisque E^* est vide. Et en accord avec l'item 2, $\neg b$ ne peut être déduit à partir de $\{a, b, a \Rightarrow b\}$ (c.-à-d. les formules des nœuds positifs de l'arbre).

$$\dfrac{\vdash a \Rightarrow b \quad \vdash a}{\vdash b}$$

Figure 5.19 – Arbre de l'exemple 60.

Exemple 61. *Soient* $\Gamma = \{(a, \{d, e\}, c)\}$ *un ensemble de règles PEC. Soient* $\mathcal{R} = (a \wedge b, \{d, e\}, c)$ *et* $\mathcal{R}' = (x, \{z\}, y)$ *deux règles PEC.*

$\mathcal{R}'$ est un impliquant de $\mathcal{R}$ modulo Γ. En effet, l'arbre de la figure 5.20 est une $\mathcal{R}'$-dérivation de $\mathcal{R}$ à partir de Γ. Notons que dans cette $\mathcal{R}'$ ne joue pourtant aucun rôle dans la dérivation.

Notons que cet arbre est aussi une dérivation de $\mathcal{R}$ à partir de Γ.

$$\dfrac{\dfrac{\vdash a \wedge b}{\vdash a} \quad \nvdash d \quad \nvdash e}{\vdash c}$$

Figure 5.20 – Arbre de l'exemple 61.

Équivalence de règles PEC

De faibles exigences sur *Cn* sont suffisantes pour montrer que la notion d'impliquant définit un pré-ordre. Le plus intéressant étant le cas de deux règles PEC qui sont impliquants

l'une de l'autre : elles sont sûrement équivalentes dans le sens fort du terme, lequel est fortement lié lié à la *Cn*-équivalence des exceptions.

Il est possible d'obtenir un tel résultat comme le montre la propriété immédiate suivante.

Propriété 11. *Soit* Γ *un ensemble vide de règles PEC. Soient* $\mathcal{R} = (\rho, \{\epsilon_1, \ldots, \epsilon_m\}, \varsigma)$ *et* $\mathcal{R}' = (\rho', \{\epsilon'_1, \ldots, \epsilon'_n\}, \varsigma')$ *deux règles PEC. Si* $\mathcal{R}$ *est un impliquant de* $\mathcal{R}'$ *modulo* Γ, $\mathcal{R}'$ *un impliquant de* $\mathcal{R}$ *modulo* Γ *alors :*

1. $\rho \Vdash \rho'$ *et* $\rho' \Vdash \rho$,
2. $\forall \epsilon_i \in \{\epsilon_1, \ldots, \epsilon_m\}, \exists \epsilon_j, \epsilon'_k$ *où* $\epsilon_j \in \{\epsilon_1, .., \epsilon_m\}$ *et* $\epsilon'_k \in \{\epsilon'_1, .., \epsilon'_n\}$, *t.q. :* $\epsilon_j \Vdash \epsilon_i$, $\epsilon_j \Vdash \epsilon'_k$ *et* $\epsilon'_k \Vdash \epsilon_j$.

L'idée de cette propriété est de vérifier (pour les règles PEC) que si deux objets sont en relation symétrique l'un par rapport à l'autre alors ils sont équivalents (en un sens déterminé par la relation qui les relie symétriquement). Ici donc, si $\mathcal{R}$ et $\mathcal{R}'$ sont impliquants mutuels, alors $\mathcal{R}$ et $\mathcal{R}'$ sont "équivalents" et la propriété précise les détails de cette "équivalence". Ainsi, l'item 2 dit essentiellement que les exceptions sont exactement les mêmes dans $\mathcal{R}$ et $\mathcal{R}'$, a l'équivalence logique près (sachant que par subsomption, il peut y avoir davantage d'exceptions dans $\mathcal{R}$ ou dans $\mathcal{R}'$).

5.3.2 Impliquants essentiels & impliquants premiers

Nous définissons maintenant le concept d'impliquant premier modulo un ensemble de règles PEC. Intuitivement, ce concept est à la notion d'impliquant un ensemble de règles PEC précédemment définie (voir Définition 97), ce que la notion d'impliquant premier modulo une théorie est à la notion d'impliquant modulo une théorie (voir Chapitre 1, Section 1.2). Dans cette définition, nous faisons appel à un autre concept que nous appelons *impliquant essentiel* modulo un ensemble de règles PEC. Intuitivement, celui-ci permet d'écarter les impliquants qui n'interviennent que de manière superflue dans la X-dérivation associée montrant l'implication.

Dans cette définition, nous considérons l'opérateur $\setminus$ comme disponible dans le cadre PEC. Intuitivement, $\setminus$ est un opérateur de retrait syntaxique modulo l'équivalence qui s'applique à un ensemble de règles PEC et à une règle PEC : $\Gamma \setminus \mathcal{R}$ retire $\mathcal{R}$ et toute règle PEC équivalente de Γ, en accord avec la notion d'équivalence discutée à la sous-section 5.3.1.

Définition 100 (impliquant essentiel)**.** *Soient* Γ *un ensemble de règles PEC,* $\mathcal{R}$ *et* $\mathcal{R}'$ *deux règles PEC.* $\mathcal{R}'$ *est un impliquant essentiel de* $\mathcal{R}$ *modulo* Γ *ssi il existe* $\Gamma' \subseteq \Gamma \setminus \{\mathcal{R}\}$ *t.q. :*

1. $\mathcal{R}'$ *est un impliquant de* $\mathcal{R}$ *modulo* Γ',
2. *il n'existe pas de dérivation d'un impliquant de* $\mathcal{R}$ *modulo* $\Gamma' \setminus \{\mathcal{R}'\}$.

Définition 101 (impliquant essentiel strict)**.** *Soit* Γ *un ensemble de règles PEC. Soient* $\mathcal{R}$ *et* $\mathcal{R}'$ *deux règles PEC.* $\mathcal{R}'$ *est un impliquant essentiel strict de* $\mathcal{R}$ *modulo* Γ *ssi* $\mathcal{R}'$ *est un impliquant essentiel de* $\mathcal{R}$ *modulo* Γ *et* $\mathcal{R}$ *n'est pas un impliquant essentiel de* $\mathcal{R}'$ *modulo* Γ.

Ensuite, pour tout $\mathcal{R}'$ impliquant essentiel strict de $\mathcal{R}$ modulo Γ, on dit que $\mathcal{R}'$ est un impliquant premier de $\mathcal{R}$ modulo Γ s'il n'existe pas de règle $\mathcal{R}''$ qui soit un impliquant essentiel strict de $\mathcal{R}$ modulo Γ, telle que $\mathcal{R}'$ en soit un impliquant essentiel strict modulo Γ.

Définition 102 (impliquant premier)**.** *Soient* Γ *un ensemble de règles PEC,* $\mathcal{R}$ *et* $\mathcal{R}'$ *deux règles PEC.* $\mathcal{R}'$ *est un impliquant premier de* $\mathcal{R}$ *modulo* Γ *ssi* $\mathcal{R}'$ *est un impliquant essentiel strict de* $\mathcal{R}$ *modulo* Γ *et il n'existe pas de règles PEC* $\mathcal{R}''$ *t.q. :*

1. $\mathcal{R}''$ *est un impliquant essentiel strict de* $\mathcal{R}$ *modulo* Γ*,*
2. $\mathcal{R}'$ *est un impliquant essentiel strict de* $\mathcal{R}''$ *modulo* Γ*.*

Nous donnons ici quelques clefs permettant d'appréhender au mieux ces définitions.

Dans un premier temps, considérons le cas particulier où Γ est vide. Dans ce cas la définition d'impliquant essentiel se simplifie : l'item 2 ne s'applique plus et un seul Γ' existe et il est vide. Intuitivement, lorsque Γ est vide, tout impliquant $\mathcal{R}'$ de $\mathcal{R}$ modulo Γ est un impliquant essentiel de $\mathcal{R}$ modulo Γ (voir Exemple 62). Notons que dans ce cas, un impliquant premier de $\mathcal{R}$ modulo Γ est un impliquant strict $\mathcal{R}'$ de $\mathcal{R}$ modulo Γ t.q. il n'existe pas un autre impliquant strict $\mathcal{R}''$ de $\mathcal{R}$ qui soit impliqué strictement par $\mathcal{R}'$ modulo Γ.

Maintenant, lorsque Γ n'est pas vide, pour qu'un impliquant $\mathcal{R}'$ de $\mathcal{R}$ modulo Γ soit un impliquant essentiel de $\mathcal{R}$ modulo Γ, il ne suffit plus qu'il soit un impliquant de $\mathcal{R}$ modulo un sous-ensemble de Γ privé de $\mathcal{R}$ et de ses équivalents logiques. En effet, il doit aussi ne pas exister de dérivation de $\mathcal{R}$ ou de l'un des impliquants modulo ce sous-ensemble privé de $\mathcal{R}'$ et de ses équivalents logiques. Ainsi, les impliquants $\mathcal{R}''$ de $\mathcal{R}$ modulo Γ qui n'interviennent que de manière superflue dans la $\mathcal{R}''$-dérivation associée montrant l'implication ne peuvent être considérés comme "essentiels" (voir Exemple 63 où il existe une dérivation de $\mathcal{R}$ modulo $\Gamma \setminus \{\mathcal{R}'\}$, et Exemple 64 où il existe une dérivation d'un impliquant de $\mathcal{R}$ modulo $\Gamma \setminus \{\mathcal{R}'\}$).

Il est important de noter que pour montrer qu'une règle PEC $\mathcal{R}'$ est un impliquant essentiel de $\mathcal{R}$ modulo Γ, il est possible de ne considérer qu'un sous-ensemble Γ' de $\Gamma \setminus \{\mathcal{R}\}$. Il est donc possible de ne considérer dans Γ' que les règles PEC de $\Gamma \setminus \{\mathcal{R}\}$ qui participent à la $\mathcal{R}'$-dérivation de $\mathcal{R}$ modulo Γ. Cela permet ainsi de ne pas considérer dans Γ' les règles PEC de $\Gamma \setminus \{\mathcal{R}\}$ qui ne participent pas à la $\mathcal{R}'$-dérivation de $\mathcal{R}$ modulo Γ, mais qui peuvent potentiellement participer à la $\mathcal{R}''$-dérivation de $\mathcal{R}$ modulo Γ, de tout autre impliquant $\mathcal{R}''$ de $\mathcal{R}$ modulo Γ. En effet, la présence de ces règles dans Γ' pourraient faire en sorte qu'il existe une dérivation de $\mathcal{R}$ (ou de l'un de ses impliquants) modulo $\Gamma \setminus \{\mathcal{R}'\}$, empêchant $\mathcal{R}'$ d'être un impliquant essentiel de $\mathcal{R}$ à tort (voir Exemple 65).

Exemple 62. *Soient* Γ *un ensemble vide de règles PEC et* $\mathcal{R} = (a \wedge b, \{\neg d, \neg e\}, c)$ *une règle PEC.*

Considérons tout d'abord les impliquants essentiels de $\mathcal{R}$ *modulo* Γ*. Notons qu'ici* $\Gamma' = \Gamma \setminus \{\mathcal{R}\} = \Gamma = \emptyset$*.*

La règle PEC $\mathcal{R}' = (a, \{\neg d \wedge \neg f\}, c)$ *est un impliquant essentiel de* $\mathcal{R}$ *modulo* Γ*. D'un côté* $\mathcal{R}'$ *est un impliquant de* $\mathcal{R}$ *modulo* Γ'*. En effet, comme le montre le premier arbre de la figure 5.21, il existe une* $\mathcal{R}'$*-dérivation de* $(a \wedge b, \{\neg d \wedge \neg f\}, c)$ *à partir de* Γ'*. D'un autre côté, il ne peut exister de dérivation de* $\mathcal{R}$ *ou de l'un de ses impliquants à partir de* $\Gamma' \setminus \{\mathcal{R}'\}$ *puisque* $\Gamma = \emptyset$*.*

De la même manière, la règle PEC $\mathcal{R}'' = (a, \emptyset, c)$ *est un impliquant essentiel de* $\mathcal{R}$ *modulo* Γ*. D'un côté* $\mathcal{R}''$ *est un impliquant de* $\mathcal{R}$ *modulo* Γ'*. En effet, comme le montre le deuxième arbre de la figure 5.21, il existe une* $\mathcal{R}''$*-dérivation de* $(a \wedge b, \emptyset, c)$ *à partir de* Γ'*. D'un autre côté, ici aussi il ne peut exister de dérivation de* $\mathcal{R}$ *ou de l'un de ses impliquants à partir de* $\Gamma' \setminus \{\mathcal{R}''\}$*.*

Considérons maintenant les impliquants essentiels de $\mathcal{R}'$ modulo Γ. Notons qu'ici $\Gamma' = \Gamma \setminus \{\mathcal{R}'\} = \Gamma = \emptyset$.

Tout d'abord, il est trivial de voir que $\mathcal{R}$ n'est pas un impliquant essentiel de $\mathcal{R}'$ modulo Γ. $\mathcal{R}'$ est donc un impliquant essentiel strict de $\mathcal{R}$ modulo Γ.

Par contre $\mathcal{R}''$ est un impliquant essentiel de $\mathcal{R}'$ modulo Γ. D'un côté $\mathcal{R}''$ est un impliquant de $\mathcal{R}'$ modulo Γ'. En effet, il existe une $\mathcal{R}''$-dérivation de $(a, \emptyset, c)$ à partir de Γ', comme le montre le troisième arbre de la figure 5.21. D'un autre côté, il ne peut exister de dérivation de $\mathcal{R}'$ ou de l'un de ses impliquants à partir de $\Gamma' \setminus \{\mathcal{R}''\}$ puisque $\Gamma' \setminus \{\mathcal{R}''\}$ est vide.

Considérons maintenant les impliquants essentiels de $\mathcal{R}''$ modulo Γ. Notons qu'ici $\Gamma' = \Gamma \setminus \{\mathcal{R}''\} = \Gamma = \emptyset$.

Il est trivial de voir qu'à la fois, $\mathcal{R}'$ et $\mathcal{R}$ ne sont pas des impliquants essentiels de $\mathcal{R}''$ modulo Γ'. Ainsi, $\mathcal{R}''$ est donc à la fois un impliquant essentiel strict de $\mathcal{R}$ et de $\mathcal{R}'$ modulo Γ. Notons que par conséquent, $\mathcal{R}''$ n'est pas un impliquant premier de $\mathcal{R}$ modulo Γ.

Notons aussi que, bien que $\mathcal{R}'$ soit un impliquant essentiel strict de $\mathcal{R}$ modulo Γ et que $\mathcal{R}''$ n'en soit pas un, $\mathcal{R}'$ n'est pas nécessairement un impliquant premier de $\mathcal{R}$ modulo Γ. En effet, il peut exister d'autres impliquants essentiels stricts de $\mathcal{R}$ modulo Γ pour lesquels $\mathcal{R}'$ est un impliquant essentiel strict modulo Γ (trivialement, la règle PEC $\mathcal{R}''' = (a \wedge b, \{\neg d \wedge \neg f\}, c)$ est un impliquant essentiel strict de $\mathcal{R}$ modulo Γ et $\mathcal{R}'$ est un impliquant essentiel strict de $\mathcal{R}'''$ modulo Γ).

$$\dfrac{\dfrac{\vdash a \wedge b}{\vdash a} \quad \nvdash \neg d \wedge \neg f}{\vdash c} \qquad \dfrac{\dfrac{\vdash a \wedge b}{\vdash a} \quad \vdash a \Rightarrow c}{\vdash c} \qquad \dfrac{\vdash a \quad \vdash a \Rightarrow c}{\vdash c}$$

FIGURE 5.21 – Arbres de l'exemple 62.

Exemple 63. *Soient $\Gamma = \{(a, \{\neg d, \neg e\}, c)\}$ un ensemble de règles PEC et $\mathcal{R} = (a \wedge b, \{\neg d, \neg e\}, c)$ une règle PEC.*

Considérons tout d'abord les impliquants essentiels de $\mathcal{R}$ modulo Γ. Notons qu'ici $\Gamma \setminus \{\mathcal{R}\} = \Gamma$. Ainsi, il ne peut exister que deux Γ' différents, à savoir :

- $\Gamma'_1 = \Gamma$,
- $\Gamma'_2 = \emptyset$.

La règle PEC $\mathcal{R}' = (x, \{\neg z\}, y)$ n'est pas un impliquant essentiel de $\mathcal{R}$ modulo Γ. Considérons tout d'abord Γ'_1. D'un côté $\mathcal{R}'$ est bien un impliquant de $\mathcal{R}$ modulo Γ'_1. En effet, comme le montre l'arbre de la figure 5.22, il existe une $\mathcal{R}'$-dérivation de $\mathcal{R}$ à partir de Γ'_1. Mais d'un autre côté, comme le montre ce même arbre, il existe une dérivation de $\mathcal{R}$ à partir de $\Gamma'_1 \setminus \{\mathcal{R}'\}$ (notons que $\Gamma'_1 \setminus \{\mathcal{R}'\} = \Gamma$). Considérons maintenant Γ'_2. Trivialement, il n'existe pas de $\mathcal{R}'$-dérivation de $\mathcal{R}$ modulo Γ'_2. Ainsi, $\mathcal{R}'$ n'est pas un impliquant de $\mathcal{R}$ à partir de Γ'_2. Par conséquent, $\mathcal{R}'$ n'est pas un impliquant essentiel strict de $\mathcal{R}$ modulo Γ et n'en est donc pas non un impliquant premier.

Par contre, la règle PEC $\mathcal{R}'' = (a, \{\neg d, \neg e\}, c)$ est quant à elle un impliquant essentiel de $\mathcal{R}$ modulo Γ. D'un côté, $\mathcal{R}''$ est un impliquant de $\mathcal{R}$ modulo Γ'_1. En effet, l'arbre de la figure 5.22 est aussi une $\mathcal{R}''$-dérivation de $\mathcal{R}$ à partir de Γ'_1. Et d'un autre côté, il n'existe pas de

dérivation de $\mathcal{R}$ ou de l'un de ses impliquants à partir de $\Gamma'_1 \setminus \{\mathcal{R}''\}$. En effet, dans ce cas $\Gamma'_1 \setminus \{\mathcal{R}''\} = \emptyset$.

Considérons maintenant les impliquants essentiels de $\mathcal{R}''$ modulo Γ. Notons qu'ici $\Gamma \setminus \{\mathcal{R}''\} = \emptyset$. Ainsi, il n'existe qu'un seul Γ' et il est vide.

Il est trivial de voir que $\mathcal{R}$ n'est pas un impliquant essentiel de $\mathcal{R}''$ modulo Γ. $\mathcal{R}''$ est donc un impliquant essentiel strict de $\mathcal{R}$ modulo Γ.

Ainsi, au sein de $\{\mathcal{R}', \mathcal{R}''\}$, seuls $\mathcal{R}''$ et ses équivalents logiques peuvent prétendre à être des impliquants premiers de $\mathcal{R}$ modulo Γ.

$$\dfrac{\dfrac{\vdash a \wedge b}{\vdash a} \qquad \nvdash \neg d \qquad \nvdash \neg e}{\vdash c}$$

FIGURE 5.22 – Arbre de l'exemple 63.

Exemple 64. *Soient $\Gamma = \{(a, \{\neg d \wedge \neg f\}, c \wedge x)\}$ un ensemble de règles PEC et $\mathcal{R} = (a \wedge b, \{\neg d, \neg e\}, c)$ une règle PEC.*

Considérons tout d'abord les impliquants essentiels de $\mathcal{R}$ modulo Γ. Notons qu'ici $\Gamma \setminus \{\mathcal{R}\} = \Gamma$. Ainsi, il ne peut exister que deux Γ' différents, à savoir :

- *$\Gamma'_1 = \Gamma$,*
- *$\Gamma'_2 = \emptyset$.*

La règle PEC $\mathcal{R}' = (x, \{\neg d \wedge \neg f\}, c)$ n'est pas un impliquant essentiel de $\mathcal{R}$ modulo Γ. Considérons tout d'abord Γ'_1. D'un côté $\mathcal{R}'$ est bien un impliquant de $\mathcal{R}$ modulo Γ'_1. En effet, comme le montre le premier arbre de la figure 5.23, il existe une $\mathcal{R}'$-dérivation de $(a \wedge b, \{\neg d \wedge \neg f\}, c)$ à partir de Γ'_1. Mais d'un autre côté, comme le montre le deuxième arbre de la figure 5.23, il existe une dérivation de $(a \wedge b, \{\neg d \wedge \neg f\}, c)$, un impliquant de $\mathcal{R}$, à partir de $\Gamma'_1 \setminus \{\mathcal{R}'\}$ (notons que $\Gamma'_1 \setminus \{\mathcal{R}'\} = \Gamma$). Considérons maintenant Γ'_2. Trivialement, il n'existe pas de $\mathcal{R}'$-dérivation de $\mathcal{R}$ modulo Γ'_2. Ainsi, $\mathcal{R}'$ n'est pas un impliquant de $\mathcal{R}$ à partir de Γ'_2. Par conséquent, $\mathcal{R}'$ n'est pas un impliquant essentiel strict de $\mathcal{R}$ modulo Γ et n'en est donc pas non plus un impliquant premier.

Par contre, la règle PEC $\mathcal{R}'' = (a, \{\neg d \wedge \neg f\}, c \wedge x)$ est quant à elle un impliquant essentiel de $\mathcal{R}$ modulo Γ. D'un côté, $\mathcal{R}''$ est un impliquant de $\mathcal{R}$ modulo Γ'_1. En effet, le deuxième arbre de la figure 5.23 est aussi une $\mathcal{R}''$-dérivation de $(a \wedge b, \{\neg d \wedge \neg f\}, c)$ à partir de Γ'_1. Et d'un autre côté, il n'existe pas de dérivation de $\mathcal{R}$ ou de l'un de ses impliquants à partir de $\Gamma'_1 \setminus \{\mathcal{R}''\}$. En effet, dans ce cas $\Gamma'_1 \setminus \{\mathcal{R}''\} = \emptyset$.

Considérons maintenant les impliquants essentiels de $\mathcal{R}''$ modulo Γ. Notons qu'ici $\Gamma \setminus \{\mathcal{R}''\} = \emptyset$. Ainsi, il n'existe qu'un seul Γ' et il est vide.

Il est trivial de voir que $\mathcal{R}$ n'est pas un impliquant essentiel de $\mathcal{R}''$ modulo Γ. $\mathcal{R}''$ est donc un impliquant essentiel strict de $\mathcal{R}$ modulo Γ.

Ainsi, au sein de $\{\mathcal{R}', \mathcal{R}''\}$, seuls $\mathcal{R}''$ et ses équivalents logiques peuvent prétendre à être des impliquants premiers de $\mathcal{R}$ modulo Γ.

Exemple 65. *Soient $\Gamma = \{(a, \{\neg d \wedge \neg f\}, c), (a \vee h, \emptyset, c)\}$ un ensemble de règles PEC et $\mathcal{R} = (a \wedge b, \{\neg d, \neg e\}, c)$ une règle PEC.*

$$\dfrac{\dfrac{\dfrac{\vdash a \wedge b}{\vdash a} \quad \nvdash \neg d \wedge \neg f}{\vdash c \wedge x} \quad \nvdash \neg d \wedge \neg f}{\vdash c} \qquad \dfrac{\dfrac{\vdash a \wedge b}{\vdash a} \quad \nvdash \neg d \wedge \neg f}{\vdash c \wedge x}$$
$$\vdash c$$

Figure 5.23 – Arbres de l'exemple 64.

Considérons tout d'abord les impliquants essentiels de $\mathcal{R}$ *modulo* Γ. *Notons qu'ici* $\Gamma \setminus \{\mathcal{R}\} = \Gamma$. *Ainsi, il existe quatre* Γ' *différents, à savoir :*

- $\Gamma'_1 = \Gamma$,
- $\Gamma'_2 = \{(a, \{\neg d \wedge \neg f\}, c)\}$,
- $\Gamma'_3 = \{(a \vee h, \emptyset, c)\}$,
- $\Gamma'_4 = \emptyset$.

La règle PEC $\mathcal{R}' = (a, \{\neg d \wedge \neg f\}, c)$ *est un impliquant essentiel de* $\mathcal{R}$ *modulo* Γ. *D'un côté* $\mathcal{R}'$ *est un impliquant de* $\mathcal{R}$ *modulo* Γ'_2 *(ou* Γ'_4*). En effet, comme le montre le premier arbre de la figure 5.24, il existe une* $\mathcal{R}'$*-dérivation de* $(a \wedge b, \{\neg d \wedge \neg f\}, c)$ *à partir de* Γ'_2 *(ou* Γ'_4*). D'un autre côté, il n'existe pas de dérivation de* $\mathcal{R}$ *ou de l'un de ses impliquants modulo* $\Gamma'_2 \setminus \{\mathcal{R}'\}$ *puisque* $\Gamma'_2 \setminus \{\mathcal{R}'\} = \Gamma'_4 = \emptyset$.

De la même manière, la règle PEC $\mathcal{R}'' = (a \vee h, \emptyset, c)$ *est un impliquant essentiel de* $\mathcal{R}$ *modulo* Γ. *D'un côté* $\mathcal{R}''$ *est un impliquant de* $\mathcal{R}$ *modulo* Γ'_3 *(ou* Γ'_4*). En effet, comme le montre le deuxième arbre de la figure 5.24, il existe une* $\mathcal{R}''$*-dérivation de* $(a \wedge b, \emptyset, c)$ *à partir de* Γ'_3 *(ou* Γ'_4*). D'un autre côté, il n'existe pas de dérivation de* $\mathcal{R}$ *ou de l'un de ses impliquants modulo* $\Gamma'_3 \setminus \{\mathcal{R}''\}$ *puisque* $\Gamma'_3 \setminus \{\mathcal{R}''\} = \Gamma'_4 = \emptyset$.

Considérons maintenant les impliquants essentiels de $\mathcal{R}'$ *modulo* Γ. *Notons qu'ici* $\Gamma \setminus \{\mathcal{R}'\} = \{(a \vee h, \emptyset, c)\}$. *Ainsi, il n'existe que deux* Γ' *différents, à savoir :*

- $\Gamma'_1 = \{(a \vee h, \emptyset, c)\}$,
- $\Gamma'_2 = \emptyset$.

Tout d'abord, il est trivial de voir que $\mathcal{R}$ *n'est pas un impliquant essentiel de* $\mathcal{R}'$ *modulo* Γ. $\mathcal{R}'$ *est donc un impliquant essentiel strict de* $\mathcal{R}$ *modulo* Γ.

Par contre, $\mathcal{R}''$ *est un impliquant essentiel de* $\mathcal{R}'$ *modulo* Γ. *D'un côté,* $\mathcal{R}''$ *est un impliquant de* $\mathcal{R}'$ *modulo* Γ'_1. *En effet, comme le montre le troisième arbre de la figure 5.24, il existe une* $\mathcal{R}''$*-dérivation de* $(a, \emptyset, c)$ *à partir de* Γ'_1. *D'un autre côté, il n'existe pas de dérivation de* $\mathcal{R}'$ *ou de l'un de ses impliquants modulo* $\Gamma'_1 \setminus \{\mathcal{R}''\}$ *puisque* $\Gamma'_1 \setminus \{\mathcal{R}''\} = \emptyset$.

Considérons maintenant les impliquants essentiels de $\mathcal{R}''$ *modulo* Γ. *Notons qu'ici* $\Gamma \setminus \{\mathcal{R}''\} = \{(a, \{\neg d \wedge \neg f\}, c)\}$. *Ainsi, il n'existe que deux* Γ' *différents, à savoir :*

- $\Gamma'_1 = \{(a, \{\neg d \wedge \neg f\}, c)\}$,
- $\Gamma'_2 = \emptyset$.

Là aussi, il est trivial de voir que $\mathcal{R}$ *n'est pas un impliquant essentiel de* $\mathcal{R}''$ *modulo* Γ. $\mathcal{R}''$ *est donc un impliquant essentiel strict de* $\mathcal{R}$ *modulo* Γ.

De la même manière, $\mathcal{R}'$ *n'est pas un impliquant essentiel de* $\mathcal{R}''$ *modulo* Γ. *Pour montrer cela, il est nécessaire de considérer les différents sous-ensembles* Γ' *de* $\Gamma \setminus \{\mathcal{R}''\}$ *:*

- *Considérons* $\Gamma'_1 = \{(a, \{\neg d \wedge \neg f\}, c)\}$. *Trivialement,* $\mathcal{R}'$ *n'est pas un impliquant de* $\mathcal{R}''$ *modulo* Γ'_1.

– *Considérons alors* $\Gamma'_2 = \emptyset$. *Trivialement là aussi,* $\mathcal{R}'$ *n'est pas un impliquant de* $\mathcal{R}''$ *modulo* Γ'_2.

Ainsi, $\mathcal{R}''$ *est un impliquant essentiel strict de* $\mathcal{R}'$ *modulo* Γ.

Par conséquent, $\mathcal{R}''$ *ne peut prétendre à être un impliquant premier de* $\mathcal{R}$ *modulo* Γ, *seuls* $\mathcal{R}'$ *et ses équivalences logiques le peuvent.*

$$\frac{\dfrac{\vdash a \wedge b}{\vdash a} \quad \nvdash \neg d \wedge \neg f}{\vdash c} \qquad \frac{\dfrac{\dfrac{\vdash a \wedge b}{\vdash a}}{\vdash a \vee h} \quad \vdash (a \vee h) \Rightarrow c}{\vdash c} \qquad \frac{\dfrac{\vdash a}{\vdash a \vee h} \quad \vdash (a \vee h) \Rightarrow c}{\vdash c}$$

FIGURE 5.24 – Arbres de l'exemple 65.

5.4 Solution générale

Dans la section 5.1, nous avons introduit le formalisme général de *règle PEC* permettant la représentation de règles avec exceptions, indépendamment de toute logique particulière. Dans la section 5.2, nous avons défini les concepts de *dérivation* et de *X-dérivation* afin de pouvoir raisonner à partir de ces règles, en ayant la possibilité de considérer des hypothèses additionnelles. Pour finir, dans la section précédente nous avons défini pour des règles PEC, les concepts d'impliquant, d'impliquant essentiel et d'impliquant premier, modulo un ensemble de règles PEC. Ici nous proposons une solution générale à notre problème technique, lequel, en se basant sur ces différentes notions, s'écrit maintenant comme suit. Soit Γ un ensemble de règles PEC et $\mathcal{R}$ une règle PEC devant prédominer dans Γ. Comment transformer Γ en Γ' de telle manière que $\Gamma' \vdash^{\mathcal{E}} \mathcal{R}$, mais que $\Gamma' \nvdash^{\mathcal{E}'} \mathcal{R}'$, pour tout impliquant essentiel strict $\mathcal{R}'$ de $\mathcal{R}$ modulo Γ ?

Rappelons que, comme nous l'avons vu dans la section précédente, il est essentiel de considérer les impliquants essentiels stricts de $\mathcal{R}$ modulo Γ pour éviter de retirer de Γ des règles PEC qui n'interviennent que de manière superflue dans la dérivation de $\mathcal{R}$. De plus, en accord avec notre solution dans le cadre classique, nous pouvons dans notre solution nous limiter à ne considérer que les impliquants premiers de $\mathcal{R}$ modulo Γ (voir Définition 102), lesquels se basent sur la définition d'impliquant essentiel strict. Ainsi, si Γ' n'est pas en capacité de déduire le moindre impliquant premier de $\mathcal{R}$ modulo Γ, il est alors nécessairement dans l'incapacité de déduire le moindre impliquant essentiel strict de $\mathcal{R}$ modulo Γ.

Chose importante, pour préempter les règles qui subsument $\mathcal{R}$, l'empêchant de jouer le rôle qu'il devrait jouer, il n'est *a priori* pas suffisant de "retirer" tous les impliquants premiers de $\mathcal{R}$ modulo Γ et d'insérer $\mathcal{R}$ ensuite (ou de réviser par $\mathcal{R}$ dans le cas incohérent). En effet, comme vu au chapitre précédent pour le cas classique, il pourrait rester dans l'ensemble Γ résultant, certaines informations d'un changement auto-conflictuel, e.g. tel que si $\mathcal{R}$ est dérivable, un de ses impliquants essentiels stricts (qui n'est pas nécessairement premier) l'est aussi. En accord avec cela, le processus sera un peu plus élaboré.

Dans la suite, nous considérons un opérateur $\setminus$ comme disponible dans le cadre PEC. Intuitivement, $\setminus$ est une forme d'opérateur de contraction qui s'applique à un ensemble de

règles PEC et à une paire de règles PEC : $\Gamma \setminus (\mathcal{R}, \mathcal{R}')$ est attendu comme contractant Γ de $\mathcal{R}'$ en la présence de $\mathcal{R}$. Intuitivement. Plus formellement, les propriétés suivantes sont exigées pour cet opérateur :

1. $\Gamma \setminus (\mathcal{R}, \mathcal{R}') \not\vdash^{\mathcal{E}}_{\mathcal{R}} \mathcal{R}'$,
2. $\Gamma \setminus (\mathcal{R}, \mathcal{R}') \vdash^{\mathcal{E}} \mathcal{R}'' \Rightarrow \Gamma \vdash^{\mathcal{E}} \mathcal{R}''$,
3. $\Gamma \setminus (\mathcal{R}, \mathcal{R}') = Cn(\Gamma \setminus (\mathcal{R}, \mathcal{R}'))$,
4. Pour $\mathcal{R} = (\rho, \{\epsilon_1, \ldots, \epsilon_n\}, \varsigma)$ où $n \geq 1$, si $\epsilon_i \notin Cn(\{\rho, \varsigma\})$ pour $i = 1..n$ alors $\Gamma \setminus (\mathcal{R}, \mathcal{R}') \not\vdash (\rho \wedge \varsigma, \emptyset, \epsilon_i)$ pour $i = 1..n$.

où $\mathcal{R}$, $\mathcal{R}'$, $\mathcal{R}''$ sont des règles PEC et Γ est un ensemble de règles PEC qui n'a pas nécessairement besoin d'être cohérent. La dernière contrainte n'est pas naturelle pour un pur opérateur d'élimination, elle est essentiellement optionnelle mais est présente ici afin de permettre d'utiliser un tel opérateur $\setminus$ pour la *définition* de l'opérateur $\oplus$ d'insertion comme dans la définition 103 que voici.

Définition 103 (opérateur $\oplus_{\rangle\mathcal{R}'\langle}$)**.** *Soit $\mathcal{R}'$ un impliquant premier de $\mathcal{R}$ modulo Γ.*
$\Gamma \oplus_{\rangle\mathcal{R}'\langle} \mathcal{R} =_{def} \Gamma \setminus (\mathcal{R}, \mathcal{R}') \cup \{\mathcal{R}\}$.

Théorème 9. *Soit Γ un ensemble de règles PEC. Soit $\mathcal{R} = (\rho, \{\epsilon_1, \ldots, \epsilon_n\}, \varsigma)$ une règle PEC où $n \geq 1$ et telle que $\epsilon_i \notin Cn(\{\rho, \varsigma\})$ pour $i = 1..n$. Soit $\mathcal{R}'$ un impliquant premier de $\mathcal{R}$ modulo Γ.*

- *$\Gamma \oplus_{\rangle\mathcal{R}'\langle} \mathcal{R}$ est cohérent,*
- *$\Gamma \oplus_{\rangle\mathcal{R}'\langle} \mathcal{R} \vdash^{\epsilon} \mathcal{R}$,*
- *$\Gamma \oplus_{\rangle\mathcal{R}'\langle} \mathcal{R} \not\vdash^{\epsilon} \mathcal{R}'$.*

Démonstration. Nous allons d'abord montrer $\Gamma \oplus_{\rangle\mathcal{R}'\langle} \mathcal{R} \vdash^{\epsilon} \mathcal{R}$. Par la définition 103, $\mathcal{R} \in \Gamma \oplus_{\rangle\mathcal{R}'\langle} \mathcal{R}$. Donc, on peut construire un arbre ayant pour racine $\vdash \varsigma$ et pour feuilles ses parents $\vdash \rho$ et $\nvdash \epsilon_1, \ldots, \nvdash \epsilon_n$. Cet arbre visiblement respecte toutes les conditions de la définition 93 sauf peut-être l'item 2. Il faut donc montrer $\epsilon_i \notin Cn(\{\gamma \text{ t.q. } (\top, \emptyset, \gamma) \in \Gamma \setminus (\mathcal{R}, \mathcal{R}')\} \cup \{\gamma_1 \supset \gamma_2 \text{ t.q. } (\gamma_1, \emptyset, \gamma_2) \in \Gamma \setminus (\mathcal{R}, \mathcal{R}')\} \cup \{\rho, \varsigma\})$. Ce qui équivaut à $\epsilon_i \notin Cn(\{\gamma_1 \supset \gamma_2 \text{ t.q. } (\gamma_1, \emptyset, \gamma_2) \in \Gamma \setminus (\mathcal{R}, \mathcal{R}')\} \cup \{\rho, \varsigma\})$ et à $\{\gamma_1 \supset \gamma_2 \text{ t.q. } (\gamma_1, \emptyset, \gamma_2) \in \Gamma \setminus (\mathcal{R}, \mathcal{R}')\} \not\models \rho \wedge \varsigma \supset \epsilon_i$. Cette dernière condition est satisfaite, via la contrainte que doit respecter $\setminus(\mathcal{R}, \mathcal{R}')$, car $\not\models \rho \wedge \varsigma \supset \epsilon_i$.
Considérons maintenant la cohérence de $\Gamma \oplus_{\rangle\mathcal{R}'\langle} \mathcal{R}$. Nous venons de montrer $\{\gamma_1 \supset \gamma_2 \text{ t.q. } (\gamma_1, \emptyset, \gamma_2) \in \Gamma \setminus (\mathcal{R}, \mathcal{R}')\} \not\models \rho \wedge \varsigma \supset \epsilon_i$. Donc, $\{\gamma_1 \supset \gamma_2 \text{ t.q. } (\gamma_1, \emptyset, \gamma_2) \in \Gamma \setminus (\mathcal{R}, \mathcal{R}')\} \not\models \bot$. Or, s'il existait une dérivation de $(\top, \emptyset, \bot)$ à partir de $\Gamma \setminus (\mathcal{R}, \mathcal{R}')$, alors elle ne contiendrait que des nœuds positifs (item 7 de la définition 93), ce qui, par les items 4 et 5 de ladite définition, entraînerait $\{\gamma_1 \supset \gamma_2 \text{ t.q. } (\gamma_1, \emptyset, \gamma_2) \in \Gamma \setminus (\mathcal{R}, \mathcal{R}')\} \models \bot$ dont nous venons de voir l'impossibilité. Donc, il n'existe pas de dérivation de $(\top, \emptyset, \bot)$ à partir de $\Gamma \setminus (\mathcal{R}, \mathcal{R}')$. Et donc, il n'existe pas de dérivation de $(\top, \emptyset, \bot)$ à partir de $\Gamma \setminus (\mathcal{R}, \mathcal{R}') \cup \{\mathcal{R}\}$ car une telle dérivation aurait $\vdash \bot$ comme racine, ce qui violerait l'item 2 de la définition 93 pour chaque ϵ_i de $\mathcal{R}$ (il en existe au moins un car $n \geq 1$).
Nous terminons par la démonstration de $\Gamma \oplus_{\rangle\mathcal{R}'\langle} \mathcal{R} \not\vdash^{\epsilon} \mathcal{R}'$. Puisque $\mathcal{R}'$ est un impliquant premier de $\mathcal{R}$ modulo Γ, la définition 103 peut s'appliquer et $\Gamma \oplus_{\rangle\mathcal{R}'\langle} \mathcal{R} = \Gamma \setminus (\mathcal{R}, \mathcal{R}') \cup \{\mathcal{R}\}$. Or, $\setminus(\mathcal{R}, \mathcal{R}')$ satisfait la contrainte $\Gamma \setminus (\mathcal{R}, \mathcal{R}') \not\vdash^{\epsilon}_{\mathcal{R}} \mathcal{R}'$. Par la définition 96, il n'existe donc pas de $\mathcal{R}$-dérivation de $\mathcal{R}'$ à partir de $\Gamma \setminus (\mathcal{R}, \mathcal{R}')$. Et donc il n'existe pas de dérivation de $\mathcal{R}'$ à partir de $\Gamma \setminus (\mathcal{R}, \mathcal{R}') \cup \{\mathcal{R}\}$. C'est-à-dire $\Gamma \oplus_{\rangle\mathcal{R}'\langle} \mathcal{R} \not\vdash^{\epsilon} \mathcal{R}'$. □

L'étape suivante consiste en la généralisation du processus précédent à l'ensemble des impliquants premiers de $\mathcal{R}$. Considérons que l'opérateur $\setminus$ soit étendu de telle sorte qu'il puisse s'appliquer à tous les éléments de son second argument lequel est maintenant un ensemble de règles PEC, nous n'avons besoin que d'une définition supplémentaire.

Définition 104 (opérateur $\oplus_{\langle \mathcal{Y} \rangle}$)**.** *Soit $\mathcal{Y}$ l'ensemble fini des impliquants premiers de $\mathcal{R}$ modulo* Γ.

$\Gamma \oplus_{\langle \mathcal{Y} \rangle} \mathcal{R} =_{def} \Gamma \setminus (\mathcal{R}, \mathcal{Y}) \cup \{\mathcal{R}\}$.

Il est aisé de montrer que cette définition jouit des propriétés prouvées pour l'opérateur $\oplus_{\langle \mathcal{R}' \rangle}$ (voir Théorème 9). Ainsi, le résultat de $\Gamma \oplus_{\langle \mathcal{Y} \rangle} \mathcal{R}$ est cohérent, permet bien de déduire $\mathcal{R}$ et de ne pas le subsumer.

Propriété 12. *Soit Γ un ensemble de règles PEC. Soit $\mathcal{R} = (\rho, \{\epsilon_1, \ldots, \epsilon_n\}, \varsigma)$ une règle PEC où $n \geq 1$ et telle que $\epsilon_i \notin Cn(\{\rho, \varsigma\})$ pour $i = 1..n$. Soit $\mathcal{Y}$ l'ensemble des impliquants premiers de $\mathcal{R}$ modulo* Γ.

- $\Gamma \oplus_{\rangle \mathcal{Y} \langle} \mathcal{R}$ *est cohérent,*
- $\Gamma \oplus_{\rangle \mathcal{Y} \langle} \mathcal{R} \vdash^{\epsilon} \mathcal{R}$,
- $\Gamma \oplus_{\rangle \mathcal{Y} \langle} \mathcal{R} \nvdash^{\epsilon} \mathcal{R}', \forall \mathcal{R}' \in \mathcal{Y}$.

5.4.1 Instanciation au cadre propositionnel standard

Clairement, dans le cadre de la logique propositionnelle, les concepts de dérivation et de X-dérivation se réduisent aux concepts d'arbre de déduction et d'arbre de déduction modulo la prise en compte d'une hypothèse supplémentaire X. La condition de non-existence d'une g-dérivation de f à partir de $\Gamma \setminus \{g \Rightarrow f\}$ s'écrit aussi en $\Gamma \setminus \{g \Rightarrow f\} \cup \{g\} \nvdash f$. Les conditions supplémentaires imposées sur l'opérateur $\oplus$ sont directement satisfaites de par les propriétés imposées sur $\setminus$ dans le cadre classique.

5.5 Conclusion & perspectives

Dans un premier temps, dans ce chapitre, nous avons présenté un cadre de travail permettant de représenter de manière uniforme, à la fois des connaissances monotones et des règles révisables. Des outils de dérivation permettant de raisonner et d'inférer à partir, indifféremment, de ces deux types de connaissances, ont été définis. Aussi, nous avons défini un concept de X-dérivation qui permet d'établir des règles avec exceptions comme des hypothèses additionnelles de raisonnement, ajoutant donc un caractère révisable aux règles avec exceptions. Nous croyons que cette forme de raisonnement hypothétique à double niveau devrait être explorée et raffinée dans le futur. De plus, différents types d'impliquants pour des règles révisables pourraient être développés, en accord avec la forme de raisonnement qui est effectivement modélisée et des rôles épistémologiques effectivement attendus des prémisses, exceptions et conclusions.

Dans un second temps, dans ce chapitre, ce cadre de travail a été mis en œuvre afin de proposer une solution à notre problème spécifique de représentation des connaissances et du raisonnement, lequel n'a reçu que très peu d'intérêt jusqu'alors dans la littérature. À savoir,

comment une information peut préempter les informations qui la subsument lors de son insertion dans un ensemble de connaissances ? Nous affirmons qu'un tel problème ne doit pas être considéré comme trivial. En effet, dans la vie de tous les jours nous devons régulièrement incorporer des informations qui sont logiquement plus faibles mais qui apparaissent pourtant plus précises que celles déjà établies et qui, d'une certaine manière, doivent les inhiber.

Conclusion générale

Dans ce travail de thèse, nous nous sommes intéressés à la résolution d'un problème fondamental de la représentation des connaissances et des raisonnements en logique : comment faire prédominer une nouvelle information lors de son insertion dans un ensemble de connaissances qui la subsument ? En particulier, comment occulter des connaissances par des informations logiquement plus faibles, mais au contenu épistémologiquement plus spécifique ou plus précis ? Conceptuellement très simple dans sa formulation, ce problème nécessite pourtant des considérations non triviales pour le résoudre de façon satisfaisante. De manière surprenante, celui-ci n'avait reçu que très peu d'intérêt de la part de la communauté de la recherche en I.A. jusque-là. S'agissant d'une nouvelle information non logiquement contradictoire avec les connaissances pré-existantes, ni les différentes logiques non monotones ni les approches standard de révision de croyances n'adressent ce problème de manière appropriée.

Au terme de ce travail de thèse, dressons le résumé de nos résultats ainsi que les perspectives qui nous semblent les plus prometteuses. Notons que ces travaux ont fait l'objet de communications en langue anglaise (voir [Besnard *et al.* 2011a] et [Besnard *et al.* 2011b]).

Résumé des contributions

Au sein du chapitre 4, nous avons montré comment imposer la prédominance d'informations logiquement plus faibles dans le cadre élémentaire de la logique booléenne, et plus particulièrement dans son fragment clausal. Nous avons vu que la solution triviale qui consiste à simplement éliminer de l'ensemble de connaissances existant les impliquants de l'information devant prédominer, puis à ajouter celle-ci à l'ensemble de connaissances résultant, n'est pas satisfaisante. En effet, l'introduction de la nouvelle information peut créer de nouveaux cheminements de raisonnement (ou en rétablir d'autres que nous viendrions à peine de retirer), lesquels permettent de déduire certains de ces impliquants. La solution que nous avons proposée est donc plus élaborée. Dans le cadre booléen clausal, nous avons ainsi caractérisé une famille d'opérateurs permettant de forcer la prédominance d'informations logiquement plus faibles. Nous avons effectué cette analyse d'abord sur le plan de la logique et ensuite sur le plan algorithmique pratique, jusqu'à l'implantation et la conduite d'études expérimentales.

Au sein du chapitre 5, nous avons étudié ce problème de prédominance dans le cadre plus expressif de logiques permettant de représenter à la fois des formules classiques booléennes et des règles avec exceptions lorsque celles-ci reposent sur des tests de cohérence. Nous avons proposé une solution à ce problème indépendamment d'un formalisme d'une logique non monotone en particulier. À cet effet, nous avons défini le formalisme des règles PEC (pour Prémisses-Exceptions-Conclusion) permettant la représentation uniforme de ces deux types d'objets. Nous avons ensuite défini des outils de dérivation permettant de manipuler ces règles et étendu différents concepts d'impliquants pour ce formalisme. De manière intéressante, nos outils de dérivation permettent donc de dériver aussi bien des formules classiques que des règles avec exceptions soumises à des tests de cohérence. Ils permettent aussi de raisonner modulo des hypothèses supplémentaires offrant ainsi la possibilité d'un raisonnement hypothé-

tique à plusieurs niveaux. Ces outils ouvrent eux-mêmes, nous le pensons, des perspectives intéressantes au-delà de l'utilisation spécifique que nous en avons faite.

De fait, nous percevons de nombreuses perspectives et poursuites possibles de ces travaux.

Perspectives de travaux futurs

Tout d'abord, la méthode ainsi définie pour imposer la prédominance d'informations plus faibles repose sur des familles d'opérateurs de contraction et de révision qu'il faudrait instancier tout en les généralisant au cadre PEC pour obtenir des outils spécifiques précis. En effet, nous avons défini notre technique modulo des contraintes minimales sur ces opérateurs et avons montré que les opérateurs *à la* AGM correspondant qui reposent sur les principes fondamentaux de cohérence, de primauté de la nouvelle information et de changement minimal, appartiennent à ces familles d'opérateurs. L'étude d'opérateurs adaptés de contraction et de révision dans le formalisme PEC est, selon nous, déjà en elle-même une piste prometteuse pour de futures recherches.

Une des contributions de cette thèse a été de donner et de motiver différents concepts d'impliquants dans le cadre générique PEC. Il reste à concevoir et à étudier des algorithmes qui permettent de calculer pareils impliquants, modulo bien évidemment des limites de complexité. À cet égard, nous pourrions envisager de mettre en évidence des fragments du formalisme PEC, comme peut-être par exemple un fragment clausal, permettant un calcul efficace de ces impliquants et la résolution en temps "raisonnable" du problème de prédominance. Par ailleurs, le concept d'impliquant est présent dans de nombreux domaines de l'I.A. comme le *diagnostic* où il apparaît comme une étape fondamentale dans les systèmes de diagnostic pour le *traitement de fautes multiples* (initié par [Reiter 1987] et [de Kleer & Williams 1987]), la *démonstration automatique* où il est utilisé pour la *preuve automatique de théorèmes* (initié par [Slagle *et al.* 1969] et [Slagle *et al.* 1970]), ou encore la *compilation logique* où de nombreuses méthodes sont fondées sur des mécanismes permettant de construire une représentation des connaissances à base d'impliqués et d'impliquants (voir [Schrag 1996] ou encore [Marquis 1995], pour ne citer que quelques travaux récents). Nous souhaitons aussi explorer comment la définition que nous en avons donnée dans un cadre plus expressif que celui de la logique standard, permet d'étendre son utilisation dans ces autres domaines.

Instancier notre travail à des cadres de représentation reposant sur des logiques non monotones spécifiques est aussi une perspective naturelle de poursuite de notre travail. Que ce soit des instanciations et des adaptations à différentes variantes, par exemple, de la logique de défauts, de la circonscription, ou encore à la *programmation logique* et ses versions dédiées à l'*Answer Set Programming* (voir par exemple [Baral & Gelfond 1994] et [Baral 2003] pour une synthèse des travaux récents dans ces deux domaines).

Une autre poursuite de cette thèse pourrait consister à étudier le caractère itératif possible de nos opérateurs de prédominance et à analyser les éventuelles contraintes à leur apporter pour en garantir certaines propriétés (associativité, commutativité, etc.).

Enfin, une perspective très ambitieuse et très ouverte pourrait être la suivante. Différents domaines de l'I.A. ne reconnaissent un caractère conflictuel aux raisonnements que si la confrontation de ceux-ci produit une incohérence logique. Revoir ces différents domaines et les outils logiques sur lesquels ils se fondent (notamment les logiques non monotones et les tech-

niques de révision de croyances et de mise à jour) pour prendre en compte des formes de conflits comme celle étudiée dans cette thèse qui ne se traduisent pas par des incohérences logiques serait un objectif ambitieux, bien évidemment difficile, mais qui selon nous vaudrait la peine d'être exploré avec attention. Dans ce contexte, nous envisageons notamment de jeter les bases d'une nouvelle théorie de traitement des exceptions, en particulier au sens informatique du terme où un cas est expressément traité de manière spécifique sans que ce cas se présente comme une contradiction logique par rapport aux autres cas.

Bibliographie

[Alchourrón *et al.* 1985] Carlos Alchourrón, Peter Gärdenfors et David Makinson. On the logic of theory change : partial meet contraction and revision functions. Journal of Symbolic Logic, vol. 50, no. 2, pages 510–530, 1985. (Cité en pages 2, 4, 51, 52, 53, 56, 69, 74 et 88.)

[Alliot & Schiex 1993] Jean-Marc Alliot et Thomas Schiex. Intelligence artificielle & informatique théorique. Cepadues, 1993. (Cité en page 20.)

[Baral & Gelfond 1994] Chitta Baral et Michael Gelfond. Logic programming and knowledge representation. Journal of Logic Programming, vol. 19, pages 73–148, 1994. (Cité en page 132.)

[Baral 2003] Chitta Baral. Knowledge representation, reasoning and declarative problem solving. Cambridge University Press, 2003. (Cité en page 132.)

[Benferhat *et al.* 1999] Salem Benferhat, Didier Dubois et Henri Prade. A computational model for belief change and fusing ordered belief bases. Dans Frontiers in Belief revision, pages 109–134. Kluwer Academic Publishers, 1999. (Cité en page 52.)

[Benferhat *et al.* 2000] Salem Benferhat, Didier Dubois, Souhila Kaci et Henri Prade. Encoding information fusion in possibilistic logic : a general framework for rational syntactic merging. Dans 14^{th} European Conference on Artificial Intelligence (ECAI'00), pages 3–7. IOS Press, 2000. (Cité en page 2.)

[Benferhat *et al.* 2010] Salem Benferhat, Jonathan Ben-Naim, Odile Papini et Éric Würbel. An answer set programming encoding of prioritized removed sets revision : application to GIS. Journal of Applied Intelligence, vol. 32, no. 1, pages 60–87, 2010. (Cité en page 52.)

[Bertossi *et al.* 2005] Leopoldo Bertossi, Anthony Hunter et Torsten Schaub. Inconsistency tolerance. Springer Verlag, 2005. (Cité en page 43.)

[Besnard & Hunter 1995] Philippe Besnard et Anthony Hunter. Quasi-classical logic : non-trivializable classical reasoning from inconsistent information. Dans 3^{rd} European Conference on Symbolic and Quantitative Approaches to Reasoning and Uncertainty (ECSQARU'95), pages 44–51. Springer Verlag, 1995. (Cité en page 43.)

[Besnard *et al.* 1988] Philippe Besnard, Jean Houdebine et Raymond Rolland. A formula circumscriptively both valid and unprovable. Dans 8^{th} European Conference on Artificial Intelligence (ECAI'88), pages 516–518. Morgan Kaufmann, 1988. (Cité en page 48.)

[Besnard *et al.* 2009a] Philippe Besnard, Éric Grégoire et Sébastien Ramon. Corriger la logique des défauts par la logique des défauts. Dans Journées Nationales de l'Intelligence Artificielle Fondamentale (IAF'09), 2009. (Cité en page 43.)

[Besnard *et al.* 2009b] Philippe Besnard, Éric Grégoire et Sébastien Ramon. A default logic patch for default logic. Dans Claudio Sossai et Gaetano Chemello, editeurs, 10^{th}

European Conference on Symbolic and Quantitative Approaches to Reasoning with Uncertainty (ECSQARU'09), pages 578–589. LNAI 5590, Springer, 2009. (Cité en page 43.)

[Besnard *et al.* 2011a] Philippe Besnard, Éric Grégoire et Sébastien Ramon. Enforcing logically weaker knowledge in classical logic. Dans 5^{th} International Conference on Knowledge Science Engineering and Management (KSEM'11). LNAI 7091, Springer (à paraître), décembre 2011. (Cité en pages 66 et 131.)

[Besnard *et al.* 2011b] Philippe Besnard, Éric Grégoire et Sébastien Ramon. Overriding subsuming rules. Dans 11^{th} European Conference on Symbolic and Quantitative Approaches to Reasoning with Uncertainty (ECSQARU'11), pages 532–544. LNAI 6717, Springer, 2011. (Cité en pages 102 et 131.)

[Besnard 1988] Philippe Besnard. Axiomatisations in the metatheory of nonmonotonic inference systems. Dans 7^{th} biennal conference of the Canadian Society for Computational Studies of Intelligence (CSCSI'88), pages 117–124. Morgan Kaufmann, 1988. (Cité en page 51.)

[Besnard 1989] Philippe Besnard. An introduction to default logic. Symbolic computation. Springer, 1989. (Cité en pages 40 et 44.)

[Boole 1847] George Boole. The mathematical analysis of logic : being an essay towards a calculus of deductive reasoning. Cambridge, Philosophical Library, 1847. (Cité en pages 1 et 19.)

[Boole 1854] George Boole. An investigation of the laws of thought : on which are founded the mathematical theories of logic and probabilities. Walton and Maberly, 1854. (Cité en pages 1 et 19.)

[Borgida 1985] Alexander Borgida. Language features for flexible handling of exceptions in information systems. ACM Transactions on DataBase Systems (TODS), vol. 10, 1985. (Cité en page 55.)

[Brass 1993] Stefan Brass. On the semantics of supernormal defaults. Dans 13^{th} International Joint Conference on Artificial Intelligence (IJCAI'93), pages 578–583. Morgan Kaufmann, 1993. (Cité en page 105.)

[Brewka & Eiter 2000] Gerhard Brewka et Thomas Eiter. Prioritizing default logic. Dans Intellectics and Computational Logic, pages 27–45. Kluwer Academic, 2000. (Cité en page 44.)

[Brewka 1991] Gerhard Brewka. Cumulative default logic : in defense of nonmonotonic inference rules. Artificial Intelligence, vol. 50, no. 2, pages 183–205, 1991. (Cité en page 39.)

[Cantor & Jourdain 1911] Georg Cantor et Philip Jourdain. Contributions to the founding of the theory of transfinite numbers. The Open Court Publishing Company, 1911. (Cité en page 9.)

[Cantor 1895] Georg Cantor. Beiträge zur Begründung der transfiniten Mengenlehre. Mathematische Annalen, vol. 46, pages 481–512, 1895. (Cité en page 9.)

[Cayrol *et al.* 1998] Claudette Cayrol, Marie-Christine Lagasquie-Schiex et Thomas Schiex. Nonmonotonic reasoning : from complexity to algorithms. Annals of Mathematics and Artificial Intelligence, vol. 22, no. 3–4, pages 207–236, 1998. (Cité en page 1.)

[Church 1936] Alonzo Church. An unsolvable problem of elementary number theory. American Journal of Mathematics, vol. 58, no. 2, pages 345–363, 1936. (Cité en page 14.)

[Clark 1978] Keith L Clark. Negation as failure. Dans Logic and Data Bases, volume 1, pages 293–322. Plenum Press, 1978. (Cité en page 38.)

[Cook 1971] Stephen Cook. The complexity of theorem-proving procedures. Dans 3^{rd} annual ACM symposium on Theory of computing, pages 151–158. Association for Computing Machinery, 1971. (Cité en pages 15 et 31.)

[Curry & Feys 1958] Haskell Curry et Robert Feys. Combinatory logic, volume 1. North-Holland, 1958. (Cité en page 28.)

[Curry *et al.* 1972] Haskell Curry, J Roger Hindley et Jonathan Seldin. Combinatory logic, volume 2. North-Holland, 1972. (Cité en page 28.)

[Dalal 1988] Mukesh Dalal. Investigations into a theory of knowledge base revision : preliminary report. Dans 7^{th} National Conference on Artificial Intelligence (AAAI'88), pages 475–479. AAAI Press, 1988. (Cité en page 55.)

[Darwiche & Marquis 2002] Adnan Darwiche et Pierre Marquis. A knowledge compilation map. Journal of Artificial Intelligence Research, vol. 17, pages 229–264, 2002. (Cité en page 19.)

[de Kleer & Williams 1987] J de Kleer et B.C Williams. Diagnosing multiple faults. Artificial Intelligence, vol. 32, pages 97–130, 1987. (Cité en page 132.)

[Delgrande *et al.* 2007] James P. Delgrande, Daphne H. Liu, Torsten Schaub et Sven Thiele. COBA 2.0 : a consistency-based belief change system. Dans 9^{th} European Conference on Symbolic and Quantitative Approaches to Reasoning with Uncertainty (ECSQARU'07), pages 78–90. Springer, 2007. (Cité en page 52.)

[Eiter & Gottlob 1992] Thomas Eiter et Georg Gottlob. On the complexity of propositional knowledge base revision, updates, and counterfactuals. Artificial Intelligence, vol. 57, no. 2–3, pages 227–270, 1992. (Cité en pages 32, 77 et 81.)

[Fuhrmann & Hansson 1994] André Fuhrmann et Sven Ove Hansson. A survey of multiple contractions. Journal of Logic, Language and Information, vol. 3, no. 1, pages 39–76, 1994. (Cité en pages 56, 58, 59 et 73.)

[Gabbay & Hunter 1991] Dov M. Gabbay et Anthony Hunter. Making inconsistency respectable : a logical framework for inconsistency in reasoning. Dans 1^{st} International Workshop on Fundamentals of Artificial Intelligence Research, pages 19–32. Springer Verlag, 1991. (Cité en page 43.)

[Gabbay & Robinson 1994] C. J. Gabbay Dov M.and Hogger et J. A. Robinson, editeurs. Handbook of logic in artificial intelligence and logic programming : nonmonotonic reasoning and uncertain reasoning, volume 3. Oxford University Press, 1994. (Cité en page 38.)

[Gabbay 1985] Dov Gabbay. Theoretical foundations for non-monotonic reasoning in expert systems. Springer Verlag, 1985. (Cité en page 51.)

[Gärdenfors 1990] Peter Gärdenfors. Belief revision and nonmonotonic logic : two sides of the same coin ? Dans 9th European Conference on Artificial Intelligence (ECAI'90), pages 768–773. Pitman, 1990. (Cité en page 51.)

[Garey & Johnson 1979] Michae Garey et David Johnson. Computers and intractability : a guide to the theory of np-completeness. W. H. Freeman & Co., 1979. (Cité en page 11.)

[Gentzen *et al.* 1955] Gerhard Gentzen, Robert Feys et Jean Ladrière. Recherches sur la déduction logique (traduction de Untersuchungen über das logische Schließen II). Presses Universitaires de France, 1955. (Cité en page 27.)

[Gentzen 1935a] Gerhard Gentzen. Untersuchungen über das logische Schließen I. Mathematische Zeitschrift, vol. 39, pages 176–210, 1935. (Cité en page 27.)

[Gentzen 1935b] Gerhard Gentzen. Untersuchungen über das logische Schließen II. Mathematische Zeitschrift, vol. 39, pages 405–431, 1935. (Cité en page 27.)

[Gottlob 1992] Georg Gottlob. Complexity results for nonmonotonic logics. Journal of Logic of Computation, vol. 2, no. 3, pages 397–425, 1992. (Cité en page 1.)

[Grégoire *et al.* 2006a] Éric Grégoire, Bertrand Mazure et Cédric Piette. Extracting MUSes. Dans 17th European Conference on Artificial Intelligence (ECAI'06), pages 387–391. IOS Press, 2006. (Cité en page 90.)

[Grégoire *et al.* 2006b] Éric Grégoire, Bertrand Mazure et Cédric Piette. Tracking MUSes and strict inconsistent covers. Dans 6th International Conference on Formal Methods in Computer Aided Design (FMCAD'06), pages 39–46. IEEE Computer Society, 2006. (Cité en pages 33 et 90.)

[Grégoire *et al.* 2007a] Éric Grégoire, Bertrand Mazure et Cédric Piette. Boosting a complete technique to find MSS and MUS thanks to a local search oracle. Dans 20th International Joint Conference on Artificial Intelligence (IJCAI'07), pages 2300–2305. Morgan Kaufmann, 2007. (Cité en page 90.)

[Grégoire *et al.* 2007b] Éric Grégoire, Bertrand Mazure et Cédric Piette. Local-search extraction of MUSes. Constraints, vol. 12, no. 3, pages 325–344, 2007. (Cité en page 90.)

[Grégoire *et al.* 2009a] Éric Grégoire, Bertrand Mazure et Cédric Piette. Does this set of clauses overlap with at least one MUS ? Dans 22nd International Conference on Automated Deduction (CADE'09), pages 100–115. LNCS, 2009. (Cité en page 90.)

[Grégoire *et al.* 2009b] Éric Grégoire, Bertrand Mazure et Cédric Piette. Using local search to find MSSes and MUSes. European Journal of Operational Research, vol. 199, no. 3, pages 640–646, 2009. (Cité en page 90.)

[Grégoire 1990] Éric Grégoire. Logiques non monotones et intelligence artificielle. Langue, raisonnement, calcul. Hermes, 1990. (Cité en page 38.)

[Grégoire 2002] Éric Grégoire. Fusing cooperative technical-specification knowledge components. Dans 14th IEEE International Conference on Tools with Artificial Intelligence (ICTAI'02), pages 535–542. IEEE Computer Society, 2002. (Cité en page 97.)

[Grégoire 2003] Éric Grégoire. Fusing cooperative technical-specification knowledge components. International Journal on Artificial Intelligence Tools, vol. 12, no. 3, pages 265–278, 2003. (Cité en page 97.)

[Grégoire 2007] Éric Grégoire. Knowledge refinement through revision. Dans 8^{th} IEEE International Conference on Information Reuse and Integration (IRI'07), pages 285–290. IEEE Press, 2007. (Cité en page 97.)

[Grove 1988] Adam Grove. Two modellings for theory change. Journal of Philosophical Logic, vol. 17, pages 157–170, 1988. (Cité en page 55.)

[Gärdenfors 1988] Peter Gärdenfors. Knowledge in flux : modeling the dynamics of epistemic states, volume 103. MIT Press, 1988. (Cité en pages 51, 52, 57, 69 et 76.)

[Hansson 1991] Sven Ove Hansson. Belief contraction without recovery. Studia Logica, vol. 50, pages 251–260, 1991. (Cité en pages 56 et 71.)

[Katsuno & Mendelzon 1991a] Hirofumi Katsuno et Alberto Mendelzon. On the difference between updating a knowledge base and revising it. Dans 2^{nd} International Conference on Principles of Knowledge Representation and Reasoning (KR'91), pages 387–394. Morgan Kaufmann, 1991. (Cité en pages 52, 55, 59, 60, 69, 70, 71 et 76.)

[Katsuno & Mendelzon 1991b] Hirofumi Katsuno et Alberto Mendelzon. Propositional knowledge base revision and minimal change. Artificial Intelligence, vol. 52, pages 263–294, 1991. (Cité en pages 52, 54, 55 et 57.)

[Keller & Winslett 1985] Arthur Keller et Marianne Winslett. On the use of an extended relational model to handle changing incomplete information. IEEE Transactions on Software Engineering, vol. 11, no. 7, pages 620–633, 1985. (Cité en page 59.)

[Kleene 1967] Stephen Cole Kleene. Mathematical logic. Wiley, 1967. (Cité en pages 1 et 20.)

[Konieczny & Grégoire 2006] Sébastien Konieczny et Éric Grégoire. Logic-based information fusion in artificial intelligence. Information Fusion, vol. 7, no. 1, pages 4–18, 2006. (Cité en page 43.)

[Konieczny & Pino Pérez 1998] Sébastien Konieczny et Ramón Pino Pérez. On the logic of merging. Dans 6^{th} International Conference on Principles of Knowledge Representation and Reasoning (KR'98), pages 488–498. Morgan Kaufmann, 1998. (Cité en page 2.)

[Konieczny 1999] Sébastien Konieczny. Sur la logique du changement : révision et fusion de bases de connaissance. Thèse de doctorat, Université des Sciences et Technologies de Lille, Laboratoire d'Informatique Fondamentale de Lille, 1999. (Cité en page 52.)

[Konieczny 2010] Sébastien Konieczny. Raisonnement et Incohérence. Habilitation à Diriger les Recherches, Université d'Artois, Centre de Recherche en Informatique de Lens, 2010. (Cité en page 52.)

[Kraus *et al.* 1990] Sarit Kraus, Daniel J. Lehmann et Menachem Magidor. Nonmonotonic eeasoning, preferential models and cumulative logics. Artificial Intelligence, vol. 44, no. 1-2, pages 167–207, 1990. (Cité en page 51.)

[Lang & Marquis 2002] Jérome. Lang et Pierre Marquis. Resolving inconsistencies by variable forgetting. Dans 8^{th} International Conference on Principles of Knowledge Representation and Reasoning (KR'02), pages 239–250. Morgan Kaufmann, 2002. (Cité en page 61.)

[Lehmann 1995] Daniel Lehmann. Belief revision, revised. Dans 14^{th} International Joint Conference on Artificial Intelligence (IJCAI'95), pages 1534–1540. Morgan Kaufmann, 1995. (Cité en page 55.)

[Liffiton & Sakallah 2005] Mark Liffiton et Karem Sakallah. On finding all minimally unsatisfiable subformulas. Dans 8^{th} International Conference on Theory and Applications of Satisfiability Testing (SAT'05), pages 173–186. Springer Verlag, 2005. (Cité en page 90.)

[Lin & Reiter 1994] Fangzhen Lin et Ray Reiter. Forget it ! Dans AAAI Fall Symposium on Relevance, pages 154–159, 1994. (Cité en page 61.)

[Lukaszewicz 1988] Witold Lukaszewicz. Considerations on default logic : an alternative approach. Computational intelligence, vol. 4, no. 1, pages 1–16, 1988. (Cité en pages 39 et 43.)

[Makinson 1987] David Makinson. On the status of the postulate of recovery in the logic of theory Change. Journal of Philosophical Logic, vol. 16, no. 4, pages 383–394, 1987. (Cité en pages 56 et 71.)

[Makinson 1989] David Makinson. General theory of cumulative inference. Dans 5^{th} International Workshop on Logic Programming and Non-Monotonic Reasoning (LPNMR'89), pages 1–18. Springer Verlag, 1989. (Cité en page 51.)

[Makinson 1994] David Makinson. General pattern in nonmonotonic reasoning. Oxford University Press, 1994. (Cité en page 51.)

[Manor & Rescher 1970] Ruth Manor et Nicholas Rescher. On inferences from inconsistent information. Theory and Decision, vol. 1, pages 179–219, 1970. (Cité en page 1.)

[Marquis 1995] Pierre Marquis. Knowledge compilation using theory prime implicates. Dans 14^{th} International Joint Conference on Artificial Intelligence (IJCAI'95), pages 837–845. Morgan Kaufmann, 1995. (Cité en page 132.)

[McCarthy 1980] John McCarthy. Circumscription - a form of non-monotonic reasoning. Artificial Intelligence, vol. 13, no. 1-2, pages 27–39, 1980. (Cité en pages 38, 39, 44 et 103.)

[McCarthy 1986] John McCarthy. Applications of circumscription to formalizing common sense knowledge. Artificial Intelligence, vol. 59, pages 23–26, 1986. (Cité en pages 38, 39, 44 et 103.)

[McDermott 1982] Drew McDermott. Nonmonotonic logic II : nonmonotonic modal theories. Journal of the ACM, vol. 29, pages 33–57, 1982. (Cité en pages 38 et 39.)

[Mikitiuk & Truszczynski 1993] Artur Mikitiuk et Miroslaw Truszczynski. Rational default logic and disjunctive logic programming. Dans 2^{nd} International Workshop on Logic Programming and Non-Monotonic Reasoning (LPNMR'93), pages 283–299. MIT Press, 1993. (Cité en page 39.)

[Moore 1985] Robert C. Moore. Semantical considerations on nonmonotonic logic. Artificial Intelligence, vol. 25, pages 75–94, 1985. (Cité en pages 38 et 39.)

[Nebel 1991] Bernhard Nebel. Belief revision and default reasoning : syntax-based approaches. Dans 2^{nd} International Conference on Principles of Knowledge Representation and Reasoning (KR'91), pages 417–428. Morgan Kaufmann, 1991. (Cité en page 55.)

[Nebel 1992] Bernhard Nebel. Syntax based approaches to belief revision. Dans Belief Revision, pages 52–88. Cambridge University Press, 1992. (Cité en page 52.)

[Nebel 1998] Bernhard Nebel. How Hard is it to Revise a Belief Base ? Dans Handbook of Defeasible Reasoning and Uncertainty Management Systems : Belief Change, volume 3, pages 77–145. Kluwer Academic Publishers, 1998. (Cité en page 52.)

[Papadimitriou & Wolfe 1988] Christos Papadimitriou et David Wolfe. The complexity of facets resolved. Journal of Computer and System Sciences, vol. 37, no. 1, pages 2–13, 1988. (Cité en pages 32 et 81.)

[Papadimitriou 1994] Christos Papadimitriou. Computational complexity. Addison-Wesley, 1994. (Cité en page 11.)

[Pequeno & Buchsbaum 1991] Tarcisio Pequeno et Arthur Buchsbaum. The logic of epistemic inconsistency. Dans 2^{nd} International Conference on Principles of Knowledge Representation and Reasoning (KR'91), pages 453–460. Morgan Kaufmann, 1991. (Cité en page 103.)

[Piette 2007] Cédric Piette. Techniques algorithmiques pour l'extraction de formules minimales inconsistantes. Thèse de doctorat, Université d'Artois, Centre de Recherche en Informatique de Lens, 2007. (Cité en pages 19 et 31.)

[Reiter 1977] Raymond Reiter. On closed world data bases. Dans Logic and Data Bases, pages 55–76, 1977. (Cité en pages 38 et 44.)

[Reiter 1980] Ray Reiter. A logic for default reasoning. Artificial Intelligence, vol. 13, pages 81–132, 1980. (Cité en pages 2, 4, 37, 38, 39, 40, 41, 42, 43, 44, 102, 103 et 110.)

[Reiter 1987] Raymond Reiter. A theory of diagnosis from first principles, pages 352–371. Morgan Kaufmann, 1987. (Cité en pages 86 et 132.)

[Richards & Mooney 1995] Bradley L. Richards et Raymond J. Mooney. Automated refinement of first-order Horn-clause domain theories. Machine Learning, vol. 19, no. 2, pages 95–131, 1995. (Cité en page 97.)

[Robinson 1965] J. A. Robinson. A machine-oriented logic based on the resolution principle. Journal of ACM, vol. 12, pages 23–41, Janvier 1965. (Cité en page 27.)

[Saïs 2008] Lakhdar Saïs. Problème sat : progrès et défis. Hermes Publishing Ltd, 2008. (Cité en pages 4, 19 et 30.)

[Schaub 1992] Torsten Schaub. On constrained default theories. Dans 10^{th} European Conference on Artificial Intelligence (ECAI'92), pages 304–308. John Wiley & Sons, 1992. (Cité en page 39.)

[Schrag 1996] Robert Schrag. Compilation for critically constrained knowledge bases. Dans 13^{th} National Conference on Artificial Intelligence (AAAI'96), pages 510–515. AAAI Press, 1996. (Cité en page 132.)

[Slagle *et al.* 1969] J.R. Slagle, Chin-Liang Chang et R.C.T Lee. Completeness theorems for semantic resolution in consequence-finding. Dans 1^{st} International Joint Conference on Artificial Intelligence (IJCAI'69), pages 281–285. Morgan Kaufmann, 1969. (Cité en page 132.)

[Slagle *et al.* 1970] J.R Slagle, Chin-Liang Chang et R.C.T Lee. A new algorithm for generating prime implicants. IEEE Transactions on Computers, vol. 19, pages 304–310, 1970. (Cité en page 132.)

[Turing 1936] Alan Turing. On computable numbers, with an application to the Entscheidungsproblem. London Mathematical Society, vol. 2, no. 42, pages 230–265, 1936. (Cité en page 13.)

[van Heijenoort 2002] Jean van Heijenoort. From Frege to Gödel : a source book in mathematical logic, 1879-1931. Harvard University Press, 2002. (Cité en page 9.)

[Wassermann 2000] Renata Wassermann. An algorithm for belief revision. Dans 7^{th} International Conference on Principles of Knowledge Representation and Reasoning (KR'00), pages 345–352. Morgan Kaufmann, 2000. (Cité en page 86.)

[Winslett 1989] Marianne Winslett. Theory revision semantics for use in reasoning about action. Unpublished manuscript, 1989. (Cité en pages 60 et 71.)

[Wrobel 1996] Stefan Wrobel. First order theory refinement. Dans Advances in Inductive Logic Programming, pages 14–33. IOS Press, 1996. (Cité en page 97.)

[Wurbel *et al.* 2000] Éric Wurbel, Robert Jeansoulin et Odile Papini. Revision : an application in the framework of GIS. Dans 7^{th} International Conference on Principles of Knowledge Representation and Reasoning (KR'00), pages 505–518. Morgan Kaufmann, 2000. (Cité en page 55.)

Méthodes Permettant la Prédominance de Connaissances Subsumées

Résumé : Cette thèse s'inscrit dans le domaine de l'Intelligence Artificielle symbolique. Elle y traite d'une question fondamentale liée à la représentation des connaissances et des raisonnements à base de logique. Plus précisément, elle s'intéresse au problème pouvant se produire lors de l'insertion dans un ensemble de connaissances d'une information qui peut déjà en être déduite. Comment faire en sorte que cette nouvelle information vienne préempter les informations qui permettent son inférence ? Supposons par exemple qu'un ensemble de prémisses contienne l'information "Si l'interrupteur est enclenché alors la pièce est éclairée". Il est naturel d'espérer que l'ajout d'une règle additionnelle, en un sens plus précise que la première, et qui exprime que "Si l'interrupteur est enclenché et si l'ampoule n'est pas cassée alors la pièce est éclairée", puisse venir la préempter. En effet, il ne doit plus être suffisant de savoir que "L'interrupteur soit enclenché" pour en conclure que "La pièce est éclairée" : il faut aussi que "L'ampoule ne soit pas cassée". Remarquons que la seconde règle est cohérente avec la première et que les cadres de logiques non monotones et de révision de croyances ou de mise à jour ne traitent pas a priori de ce problème. Nous adressons d'abord cette question dans le cadre de la logique classique et ensuite dans un cadre plus général de représentation à base de logiques non monotones, et particulièrement de celles permettant la représentation de règles avec exceptions reposant sur des tests de cohérence.

Mots clés : Intelligence Artificielle symbolique, représentation des connaissances et des raisonnements, logique.

Methods Allowing the Overriding of Subsumed Knowledge

Abstract : This thesis is in line with the symbolic Artificial Intelligence domain. It deals with a fondamental issue of the logic-based knowledge and reasoning representation. Most particulary, this thesis is interested in the issue occurring when a piece of information is added to a knowledge set which already entails it. How to make sure that this new piece of information prevails the ones that allow its inference ? Suppose for instance that a premisses set contains the piece of information "If the switch is on then the room is lighted". It is natural to expect that adding an additional rule, in a way more precise than the first one, and which assert that "If the switch is on and if the lamp bulb is not broken then the room is lighted", could prevail it. Indeed, it will not be sufficient to know that "The switch is on" to conclude that "The room is lighted" : it is now necessary that "The lamp bulb is not broken". Let us note that the second rule is consistent with the first one and that the non monotonic logics and beliefs revision or update frameworks do not handle this issue in principle. First of all, we adress this issue in the classical logic framework, and secondly in a most general framework of non monotonic logic based representation, and particulary the ones that allow the representation of rules with exceptions subject of consistency test.

Keywords : symbolic Artificial Intelligence, knowledge representation and reasoning, logic