Prédominance de Connaissances Subsumées en Logique Classique

Philippe Besnard, Éric Grégoire, Sébastien Ramon

Prédominance de Connaissances Subsumées en Logique Classique *

Philippe Besnard[1] **Éric Grégoire**[2] **Sébastien Ramon**[2]
[1] IRIT CNRS UMR 5505, Toulouse, France
[2] Université d'Artois, CRIL CNRS UMR 8188, Lens, France
besnard@irit.fr {gregoire,ramon}@cril.fr

Résumé

Ce papier traite d'un problème fondamental en représentation des connaissances et des raisonnements qui, de manière surprenante, n'a reçu que très peu d'attention jusqu'ici. Celui-ci concerne l'insertion au sein d'une représentation en logique classique d'une nouvelle information devant prédominer quand celle-ci est logiquement plus faible (mais dans un sens plus précise) que l'information existante. En effet, elle ne peut ni prédominer par elle seule ni inhiber par elle-même l'information existante logiquement plus forte (mais moins précise) qui la subsume. Un cadre de travail permettant de résoudre ce problème est introduit et instancié afin de permettre à des règles de prédominer sur d'autres plus générales.

Abstract

This paper is concerned with a fundamental issue in knowledge representation and reasoning that, surprisingly, has received little attention so far. The point is that inserting some logically weaker (but in a sense more precise) information within a logic-based representation is not a straightforward process if the extra information must prevail. Indeed, it does neither prevail by itself nor disable the already existing logically stronger (but less precise) information that subsumes it. A general framework for solving this problem is introduced and instantiated to the task of making some rules prevail over more general ones.

1 Introduction

Depuis les prémisses de l'Intelligence Artificielle (I.A.), il est reconnu que la logique peut jouer différents rôles en représentation des connaissances et des raisonnements. Soit comme un modèle de compétence [13] au pseudo *niveau des connaissances* [14] par exemple, ou comme un véritable outil pour l'implantation. Récemment, un intérêt tout particulier pour les représentations à base de logique a vu le jour, motivé par la progression spectaculaire en la taille et en la véritable difficulté, des problèmes pouvant être manipulés par les prouveurs automatiques de théorèmes et les outils de vérification de cohérence basés sur des solveurs SAT [1].

Ce papier concerne donc la représentation des connaissances et des raisonnements en logique classique. Ces dernières années plusieurs écueils de la logique classique ont été traités avec succès par la communauté scientifique en I.A. [4]. Citons par exemple sa propriété de monotonie qui paraît être trop restrictive, ou la manière des systèmes déductifs complets de s'effondrer en présence d'informations contradictoires [17]. Pour autant, la question qui suit n'a pas reçu d'intérêt significatif en logique jusqu'ici, bien que cela soit ubiquitaire dans la dynamique du raisonnement et de l'apprentissage.

Considérons que nous souhaitions enrichir une base de connaissances Γ par une information qui est déjà subsumée par Γ (l'information additionnelle est donc dans un sens plus précise que la sous-base de Γ qui la subsume). Par exemple, Γ contient la règle "Si l'interrupteur est enclenché alors la pièce est éclairée" et nous souhaitons enrichir Γ de la règle plus précise "Si l'interrupteur est enclenché et si l'ampoule n'est pas cassée alors la pièce est éclairée". En se plaçant dans le cadre de la logique classique, cette dernière règle est déjà impliquée par la première. Cela signifie qu'à chaque fois que la condition de la première règle tient, à savoir, "Si l'interrupteur est enclenché" alors "La pièce est

*Les résultats mis en évidence dans ce papier ont fait l'objet d'une communication antérieure en langue anglaise [2].

1. Voir par exemple http://www.satlive.org.

éclairée" (conclusion de cette règle) est dérivé, même si une information additionnelle comme par exemple "L'ampoule est cassée" est ajoutée. Ce problème est lié à la propriété de monotonie de la logique classique, qui impose que toute conclusion dérivable le reste quelles que soient les nouvelles informations ajoutées ensuite. Les théories de la révision de croyances et les logiques non monotones ont été mises au point afin de traiter le problème de la manipulation de nouvelles informations qui contredisent des informations existantes. Mais, la nouvelle information de l'exemple précédent n'est pas contradictoire avec Γ. Ainsi, la théorie de la révision et les logiques non monotones ne sont pas une réponse à notre problème. Ce dont nous avons besoin est une approche qui permette à la nouvelle information de prédominer : "La pièce est éclairée" ne doit pouvoir être dérivable uniquement lorsque les deux conditions "L'interrupteur est enclenché" et "L'ampoule n'est pas cassée" sont toutes deux dérivables.

Ce problème peut sembler à première vue trivial : pour le résoudre, nous pourrions simplement retirer l'ancienne règle et toute autre règle étant logiquement "plus forte" que la nouvelle. Nous pourrions aussi "casser" les cheminements de raisonnement (impliquant potentiellement plusieurs règles) permettant de déduire l'ancienne règle. Seulement, cela n'est pas suffisant puisque l'insertion de la nouvelle règle devant prédominer pourrait "restaurer" ces cheminements à peine éliminés. Dans ce papier, nous examinons ce problème dans le cadre de la logique classique et fournissons une solution détaillée.

Le papier est organisé comme suit. Dans la section 2, quelques préliminaires formels de la logique classique sont rappelés. Dans la section 3, le problème énoncé ci-dessus est exprimé formellement et il est montré que la solution triviale discutée n'est pas adaptée. La section 4 présente alors une solution à ce problème et la section 5 l'étudie d'un point de vue algorithmique. Après avoir discuté de travaux similaires dans la section 6, des perspectives de recherches futures sont présentées dans la section 7.

2 Préliminaires formels

Dans ce papier, les notations suivantes sont utilisées : $\neg$, $\vee$, $\wedge$ et $\rightarrow$ représentent les connecteurs classiques de négation, de disjonction, de conjonction et d'implication matérielle, respectivement. $\bot$ et $\top$ représentent la contradiction et la tautologie, respectivement.

Soit Γ un ensemble fini de formules Booléennes, construites de manière classique à partir des connecteurs et de variables Booléennes. Une interprétation de Γ assigne la valeur de vérité $vrai$ ou $faux$ à tout littéral apparaissant dans Γ et spécifie donc la valeur de vérité de toute formule de Γ par composition. Un modèle m de Γ est une interprétation qui satisfait toutes les formules de Γ, c.-à-d. leur valeur de vérité a pour valeur $vrai$. On représente une interprétation par l'ensemble des ces littéraux dont la valeur est $vrai$. L'ensemble des modèles de Γ est noté $\mathcal{M}(\Gamma)$. Une formule f peut être déduite de Γ quand f est $vraie$ dans tous les modèles de Γ, on note cela $\Gamma \models f$. $Cn(\Gamma)$ dénote la fermeture déductive de Γ, i.e. l'ensemble de toutes les formules qui peuvent être déduites de Γ. Γ est incohérent quand il ne possède aucun modèle : Γ est alors équivalent à $\bot$. Nous notons $f \models g$ et disons qu'une formule g est une conséquence logique d'une formule f lorsque $\mathcal{M}(g) \subseteq \mathcal{M}(f)$. Lorsque $f \models g$ et que $g \models f$, g et f sont dits équivalents, on note cela $f \equiv g$. Nous notons $f \models_\Gamma g$ et disons que g est une conséquence logique de f modulo un ensemble de formules Γ lorsque $\Gamma \cup \{f\} \models g$. Lorsque $f \models_\Gamma g$ et $g \models_\Gamma f$, f et g sont dits équivalents modulo Γ, noté $f \equiv_\Gamma g$.

Toute formule Booléenne peut être réécrite sous une forme clausale équivalente, i.e. comme une conjonction de clauses, où une clause est une disjonction de littéraux et où un littéral est une variable Booléenne signée, autrement dit une variable x ou sa forme négative $\neg x$. Il est courant de représenter une clause par l'ensemble de ses littéraux. En respect avec cela, les sous-clauses d'une clause f sont représentées par les sous-ensembles de f.

Nous faisons une différence entre les représentations basées sur la syntaxe qui prennent Γ comme un ensemble de formules et les approches basées sur la sémantique qui voient Γ comme un ensemble de modèles. Dans ce dernier cas, l'ensemble des formules correspondant à Γ est déductivement clos alors que ce n'est pas le cas dans les approches basées sur la syntaxe.

Le problème consistant à vérifier si un ensemble de clauses est satisfaisable et d'en exhiber un modèle dans le cas positif est appelé problème SAT. Ce problème est NP-complet. Un sous-ensemble minimalement incohérent (en abrégé MUS) Σ d'un ensemble Γ de formules est un sous-ensemble de Γ qui est incohérent et qui est tel que tout sous-ensemble propre de Σ est satisfaisable.

3 Caractérisation logique

Soit une formule g représentant une règle que l'on veut voir prédominer 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 codage 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 codage 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").

Définition 1. *f est un impliquant strict de g ssi $f \models g$ mais $g \not\models f$.*

Clairement, dans le cadre clausal, f est un impliquant strict de g ssi f est une sous-clause stricte de g.

Définition 2. *Γ' subsume g ssi $\Gamma' \models f$ pour au moins un impliquant strict f de g.*

Avant d'aller plus loin, il est nécessaire de prendre certaines décisions d'ordre épistémologique 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 ou des circonstances additionnelles incontestables (c.-à-d. le fait "La pièce est éclairée et l'interrupteur est enclenché"), 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 1. *Soient $\Gamma = \{a, b, g = \neg a \vee \neg b \vee c\}$ un ensemble de formules et g une clause. Clairement, g représente la règle $(a \wedge b) \rightarrow c$: supposons que nous voulons la voir prédominer dans Γ. Ici, g se trouve donc déjà dans Γ. Notons que c est un impliquant strict de g. Γ' ne doit pouvoir contenir à la fois a et b car sinon $\Gamma' \models c$ et donc $\Gamma' \models a \rightarrow c$, ce signifiant que g serait subsumé par Γ'.*

Exemple 2. *Soient $\Gamma = \{\neg a, g = \neg a \vee \neg b \vee c\}$ un ensemble de formules et g une clause. g représente, ici aussi, la règle $(a \wedge b) \rightarrow c$ que nous voulons voir prédominer dans Γ. Γ' ne pourra pas contenir $\neg a$, dans le cas contraire $\Gamma' \models f = \neg a \vee c$, f représentant la règle $a \rightarrow c$ subsumant g.*

Ainsi, par rapport à ces considérations épistémologiques, nous considérons, dans 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 prédominer.

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 3. *Un opérateur de contraction $\ominus$, à partir d'un ensemble de formules Γ et d'une formule non tautologique 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. 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 impose quant à elle que le résultat d'une contraction soit un ensemble fermé déductivement.

Différents opérateurs concrets de contraction peuvent ainsi être envisagés. 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.

Lorsque Γ est cohérent, une approche naturelle pour transformer Γ en Γ' tel que $\Gamma' \models g$ mais Γ' ne subsume pas 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. Considérons l'exemple suivant.

Exemple 3. *Soit $\Gamma = \{c \rightarrow a \vee b\}$. Supposons que nous voulions transformer Γ en Γ' tel que $\Gamma' \models a \vee b \vee c$ et que Γ' ne le subsume pas. Γ ne permet de déduire aucun impliquant strict de $a \vee b \vee c$. 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 $a \vee b \vee c$. Clairement, $\Gamma' \cup \{a \vee b \vee c\}$ permet cependant de déduire $a \vee b$, qui est impliquant strict de $a \vee b \vee c$.*

4 Solution générale

Nous avons donc besoin d'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, se basant sur une utilisation un peu plus élaborée d'un opérateur $\ominus$ de contraction.

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 la forme clausale 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 $\Gamma \ominus \{h\}$ parfois 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.

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.

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

Définition 4. *Soit f un impliquant strict de g.*

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

Théorème 1.
(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 3, 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 $\Gamma \ominus \{g \rightarrow f\} \not\models \neg g$. 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. □

Il reste maintenant à étudier comment cette approche peut être étendue pour considérer tous les impliquants stricts de g. Notons 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 stricts 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).

Comme g est une clause formée de n littéraux différents, il suffit de considérer les n *impliquants premiers* de g, i.e. 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. Nous proposons ici, afin de garantir l'unicité de notre processus de transformation de Γ en Γ', d'avoir recours à la *contraction multiple* [6]. 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. La contraction multiple est caractérisée par les quatre conditions suivantes :

(**1**) $\Gamma \ominus \Lambda \subseteq \Gamma$,
(**2**) Si $\Lambda \cap Cn(\emptyset) = \emptyset$ alors $\Lambda \cap \Gamma \ominus \Lambda = \emptyset$,
(**3**) Si $\Lambda \equiv_\Gamma \Theta$ alors $\Gamma \ominus \Lambda = \Gamma \ominus \Theta$,
(**4**) Si $\varphi \in \Gamma \ominus \Lambda$ alors $\Gamma \ominus \Lambda \subseteq \Omega \subseteq \Gamma$ pour tout Ω t.q. $\Lambda \cap \Omega = \emptyset$ et $\Lambda \cap Cn(\Omega \cup \{\varphi\}) \neq \emptyset$.

Alors que ce qui précède est une caractérisation, une méthode permettant de définir une opération de contraction multiple est la suivante. Pour cela, une notation supplémentaire est nécessaire, $\Gamma \bot \Lambda$ représente les plus grands sous-ensembles maximaux de Γ qui n'impliquent aucune information de Λ. Formellement,

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

L'outil suivant est une fonction de sélection μ (pour tout Γ et Λ, chaque sous-ensemble de $\Gamma \bot \Lambda$ est considéré) qui correspond à toute fonction satisfaisant les deux conditions ci-dessous :

- $\emptyset \neq \mu(\Gamma \bot \Lambda) \subseteq \Gamma \bot \Lambda$ si $\Gamma \bot \Lambda \neq \emptyset$,
- $\mu(\Gamma \bot \Lambda) = \Gamma$ si $\Gamma \bot \Lambda = \emptyset$.

Notons bien que la spécification d'un opérateur de contraction multiple peut être faite en déterminant une fonction de sélection μ. En effet, $\ominus$ est un opérateur de contraction multiple ssi il existe une fonction de sélection μ dans $\Gamma \bot \Lambda$ tel que :

$$\Gamma \ominus \Lambda = \bigcap \mu(\Gamma \bot \Lambda).$$

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

Définition 5. *Soient $f_1, \ldots, f_n$ les 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 1).

Propriété 1.
(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.

Lorsque 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 [1], 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.

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.

Clairement, si l'opérateur $\ominus$ choisi permet de manipuler des ensembles incohérents de formules alors la nécessité de différencier le cas où Γ est cohérent de celui où il ne l'est pas pourrait disparaître puisque le traitement proposé dans le premier cas pourrait implicitement impliquer une étape de révision si nécessaire, renforçant alors la cohérence et la possibilité de dériver g.

Contre-partie sémantique

Il est aisé de fournir un opérateur sémantique alternatif à notre opérateur $\oplus'$ dans le cas général.

Considérons $g = a_1 \vee \ldots \vee a_n$ où $a_1 \ldots a_n$ sont des littéraux.

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 1 (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.

5 Étude algorithmique

L'approche proposée dans ce papier repose sur des opérateurs de révision AGM et sur des tests de satisfaisabilité Booléenne qui souffrent tous 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 croyance AGM appartiennent au second niveau de la hiérarchie polynomiale [5]. 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 à moins que $P = NP$.

L'objectif de cette étude n'est pas de développer une nouvelle plateforme expérimentale pour les problèmes relatifs à la révision Booléenne. Au lieu de cela, nous avons implanté un outil dont le but est simplement d'aider un utilisateur à la compréhension étape par étape du processus proposé dans ce papier.

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 (qui 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 $\Gamma \cup \{\neg(g \rightarrow f_i)\}$. Lorsqu'une incohérence est détectée, il est alors précisé à l'utilisateur 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, qui se focalisent principalement sur une politique de changement minimal, sont alors proposées à l'utilisateur. 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.

Concentrons nous sur cette partie de la plateforme et illustrons la en présentant nos résultats expérimentaux. La plateforme permet de détecter des MUSes ("Minimally Unsatisfaisable 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. Les politiques de révision qui éliminent des formules doivent retirer au moins une clause par MUS. Seulement, la détection et l'énumération exhaustive de tous les MUSes d'une formule CNF est intraitable dans le pire des cas. Rap-

pelons qu'une formule composée de n clauses peut contenir un nombre exponentiel de MUSes, à savoir $C_n^{n/2}$, en la taille de la formule dans le pire des cas et que vérifier si une formule appartient à l'ensemble des MUSes d'une formule incohérente est Σ_2^p-complet [5].

Cependant, quand le nombre de MUSes et leur taille restent limités, les progrès récents dans le domaine de la résolution pratique du problème SAT permettent d'obtenir un résultat complet. En accord avec cela, nous revendiquons que dans des situations de la vie courante et tout particulièrement dans des systèmes d'aide à la décision, la taille minimale des chaînes de raisonnement menant à une incohérence faisant suite à l'insertion de la nouvelle clause reste souvent limitée par rapport à la taille de la base de connaissances et ne concerne qu'un nombre de clauses restreint. De plus, dans ces systèmes, une nouvelle information ne peut souvent mener qu'à un nombre limité de MUSes. Ainsi, avec de telles hypothèses de calculabilité, pour chaque f_i la plateforme a pour but de délivrer l'ensemble complet des MUSes, qui doit être réparé pour assurer la cohérence de $\Gamma \cup \{\neg(g \rightarrow f_i)\}$.

La plateforme a été implantée en langage `C++` et les expérimentations ont toutes été conduites sur un processeur `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, la plateforme fait appel à l'algorithme `HYCAM` de Grégoire, Mazure et Piette [11]. 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 de méthodes de recherche locale et de techniques de recherche complètes. L'efficacité de l'algorithme `HYCAM` sur des instances variées et particulièrement des instances issues des dernières compétitions SAT a été démontrée à plusieurs reprises [11] [12]. Nous avons aussi expérimenté une approche alternative qui délivre une couverture incohérente stricte des instances, soit un nombre suffisant de MUSes qui puisse permettre de retrouver la cohérence si retirés. Principalement, cette approche répète les étapes suivantes jusqu'à ce que la cohérence soit retrouvée : à chaque fois qu'un MUS est trouvé, il est retiré de l'ensemble de clauses et ajouté à la couverture que l'on calcule. Très clairement, cette seconde approche [10] est censée se montrer plus efficace dans le cas général puisqu'elle ne nécessite pas de calculer exhaustivement les MUSes, ces derniers pouvant partager des intersections non-vides.

Premièrement, nous avons concentré notre attention sur des instances de test structurées issues des dernières compétitions SAT [2] dans le but de vérifier la faisabilité de notre approche qui doit fournir à l'utilisateur les ensembles minimaux de clauses menant à l'incohérence et mettre en avant les clauses devant en être retirées afin que g puisse prédominer. Pour cela nous avons considéré le cas particulier où Γ est incohérent, qui nécessite d'extraire les différentes sources d'incohérence de $\Gamma \cup \{g\}$, en considérant que g n'interagit pas nécessairement avec ces MUSes. Bien sûr, l'approche est montrée faisable pour des instances impliquant un nombre minimal de MUSes ayant une taille elle aussi limitée. La table 1 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 (#MUSes) de l'instance. Les deux dernières colonnes principales fournissent les résultats expérimentaux pour les approches calculant exhaustivement les MUSes et celles ne calculant qu'une couverture incohérente stricte, respectivement. Pour chaque approche, le temps d'exécution des méthodes (#sec) est fourni, ainsi que le nombre de clauses présentées à l'utilisateur pour chaque méthode (#Cl-dans-MUSes). Pour les approches ne calculant qu'une simple couverture incohérente stricte, le nombre de MUSes de la dite couverture est aussi fourni (#MUSes). Un *time out* a été fixé à 250 secondes.

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 et la plupart d'entre elles contiennent un nombre important de MUSes de très grandes tailles. Pourtant, nous supposons qu'une nouvelle information ne contredit souvent qu'une partie minime de l'information déjà présente dans un système d'aide à la décision, menant alors à un nombre limité de chaînes de raisonnement menant à l'incohérence, qui sont souvent de taille modeste. De manière évidente, les instances de test des compétitions SAT ne suivent que très rarement cette intuition. De plus, nous exprimons aussi le besoin d'étudier l'influence de certains paramètres clés des instances en les faisant varier, ce qui n'est pas chose aisée via des instances de test SAT. Ce sont ces diverses raisons qui nous ont amenés à signer notre propre *générateur d'instances*, qui génère des ensembles de clauses incohérents qui exhibent des MUSes en nombre réduit et ayant une taille modeste par rapport à certains paramètres.

La table 2 présente les résultats les plus pertinents de cette expérimentation, montrant qu'il est possible de fournir à l'utilisateur tous les MUSes rencontrés pour les différents $\Gamma \cup \{\neg(g \rightarrow f_i)\}$ testés, sous les hypothèses suivantes. Dans la première colonne prin-

2. Voir `http://www.satcompetition.org/2011/` pou l'édition 2011.

instances	#cla	#var	#MUSes	Calcul de tous les MUSes		Calcul d'une couverture de MUSes		
				#sec	#Cl-dans-MUSes	#MUSes	#sec	#Cl-dans-MUSes
barrel2	159	50	27	0.098	99	1	0.191	77
barrel3	942	275	67765	127	546	1	6.707	456
aim-100-1_6-no-4	160	100	1	0.084	48	1	0.153	48
aim-200-1_6-no-2	320	200	2	0.082	81	1	0.281	80
aim-200-2_0-no-4	400	200	2	0.084	43	1	0.432	42
dubois29	232	87	1	0.102	232	1	0.302	232
C168_FW_UT_851	6758	1909	102	1.798	30	1	47.12	8
C170_FR_RZ_32	4067	1659	32768	18.64	243	1	9.871	227
C202_FW_RZ_57	7434	1799	1	1.414	213	1	14.955	213
C220_FV_RZ_12	4017	1728	80272	5.057	56	1	21.607	11
C220_FV_SZ_65	4014	1728	103442	8.953	103	1	8.758	23

TABLE 1 – Extrait des résultats expérimentaux sur des instances structurées.

cipale, les différents paramètres de l'instance générée sont fournis, à savoir le nombre total de MUSes (nm), la taille des MUSes (tm), la taille de la clause g en nombre de littéraux (tg) et le pourcentage de clauses de Γ qui n'appartiennent à aucun MUS (po). Le générateur fournit une instance Γ et une clause g en accord avec ces paramètres : 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 les résultats expérimentaux pour les approches calculant exhaustivement les MUSes et celles ne calculant qu'une couverture incohérente stricte, respectivement. Pour chaque approche, le temps d'exécution des méthodes (#sec) est fourni, ainsi que le nombre de clauses présentées à l'utilisateur pour chaque méthode (#cl-dans-MUSes). Pour les approches ne calculant qu'une simple couverture incohérente stricte, le nombre de MUSes de la dite couverture est aussi fourni (#MUSes). Ici aussi, un *time out* a été fixé à 250 secondes [3].

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, ces méthodes traitent de nombreuses instances pour lesquelles les méthodes qui extraient tous les MUSes atteignent le *time out*. Par exemple, le premier *time out* de la table est lié à une instance où Γ contient 300 clauses et 217 variables et pour qui la préemption de g (constituée de 3 littéraux) dans Γ nécessite de traiter 20 MUSes (constitués de 5 clauses chacun). Les méthodes calculant une couverture de MUSes détectent 14 MUSes contenant 48 clauses en 4 secondes approximativement. Dans un autre exemple, nous avons généré une instance impliquant 5 MUSes possédant chacun 20 clauses, g étant constitué de 19 littéraux et Γ de 210 clauses et 281 variables, 80% de ces clauses n'appartenant à aucun MUS. La première approche, qui calcule tous les MUSes, extraient 42 clauses en 19 secondes. La seconde approche calcule une couverture constituée de 4 MUSes contenant au total 41 clauses en 5 secondes. Notons qu'un tel exemple où g est constitué de 19 littéraux représente une situation inhabituelle puisque dans celle-ci nous souhaitons ajouter une nouvelle information de taille très importante dans une base de laquelle elle serait déjà déductible. La dernière partie de la table montre des résultats impliquant de larges bases de connaissances (eg. 4000 clauses impliquant 3801 variables), où le nombre de MUSes est pourtant limité (5) ainsi que leur taille (5 clauses). La taille de g étant très modeste (3), les deux approches délivre un résultat dans des temps très courts. Notons que dans cette situation les méthodes qui calculent exhaustivement les MUSes 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.

6 Travaux similaires

De manière assez surprenante, ce problème n'a pas reçu qu'une très faible attention dans la littérature. Citons tout de même l'un des auteurs, 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 [7] [8], ne considérant qu'un cas très particulier de préemption uniquement ; et une formalisation de la préemption de formules logiquement plus faibles en utilisant un schéma double de révision de croyances

3. Le générateur ainsi que les algorithmes permettant de faire prédominer g dans Γ selon différentes politiques sont disponibles à l'adresse `http://www.cril.univ-artois.fr/~ramon/preempte`.

instances				Γ		Calcul de tous les MUSes		Calcul d'une couverture de MUSes		
nm	*tm*	*tg*	*po*	#cla	#var	#sec	#Cl-dans-MUSes	#MUSes	#sec	#Cl-in-MUSes
1	5	3	80	15	24	<1	3	1	<1	3
2				30	37	<1	6	2	<1	6
3				45	46	<1	9	3	<1	9
4				60	57	<1	12	3	<1	10
5				75	66	<1	15	4	<1	13
6				90	77	<1	18	5	<1	16
7				105	86	<1	21	6	<1	19
8				120	97	3	24	6	<1	20
9				135	106	7	27	7	<1	23
10				150	117	36	30	7	2	24
20				300	217	*time out*		14	4	48
30				450	317	*time out*		21	11	72
40				600	417	*time out*		27	24	94
50				750	517	*time out*		34	45	118
5	10	3	80	200	191	<1	40	4	<1	33
	20			450	441	4	90	4	2	73
	30			700	691	25	140	4	3	113
	40			950	941	*time out*		4	6	153
	50			1200	1191	*time out*		4	8	193
	100			2450	2441	*time out*		4	43	393
	150			3700	3691	*time out*		4	112	593
	200			4950	4941	*time out*		4	231	793
5	10	3	80	200	191	<1	40	4	<1	33
		4		185	181	<1	37	4	<1	31
		5		170	171	<1	34	4	<1	29
		6		155	161	<1	31	4	<1	27
		7		140	151	<1	28	4	<1	25
		8		125	141	<1	25	4	<1	23
		9		110	131	<1	22	4	<1	21
	20	19		210	281	19	42	4	5	41
	30	29		310	431	*time out*		4	12	61
	40	39		410	581	*time out*		4	24	81
	50	49		510	731	*time out*		4	41	101
5	10	3	81	200	191	<1	40	4	<1	33
			83	200	191	<1	40	4	<1	33
			85	240	229	<1	40	4	<1	33
			87	280	267	<1	40	4	<1	33
			89	360	343	<1	40	4	<1	33
			91	440	419	<1	40	4	<1	33
			93	560	533	<1	40	4	1	33
			95	800	761	<1	40	4	2	33
			97	1320	1255	<1	40	4	4	33
			99	4000	3801	<1	40	4	22	33

TABLE 2 – Extrait des résultats expérimentaux sur des instances générées.

[9]. Ce papier traite de travaux totalement différents et qui couvrent le fragment clausal entier de la logique Booléenne.

Sur certains points, le problème qui est abordé dans ce papier 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* [15] [16]. 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 traitons ici 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 code 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 par rapport à 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 papier 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.

7 Conclusions et perspectives

Ce travail peut être considéré comme un moyen d'étendre le modèle de compétence de la logique classique à une capacité de raisonnement et d'apprentissage par-

ticulière : comment une connaissance logique faible (mais possiblement plus précise) peut préempter des connaissances existantes plus fortes. Nous entrevoyons plusieurs extensions possibles de ce travail. Premièrement, ce papier se focalise sur le fragment clausal de la logique classique, qui permet d'obtenir un traitement aisé qui se base sur la manipulation de sous-clauses maximales et nous permet donc de bénéficier des techniques récentes de la communauté SAT en ce qui concerne le calcul des MUSes. 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. Deuxièmement, il serait naturel d'étendre ce cadre de travail au cas de la logique du premier ordre, tenant compte alors de quantification. Aussi, nous pourrions étendre notre cadre représentatif 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 en utilisant des tests de cohérence. Un premier travail dans ce sens a été réalisé [3].

Références

[1] Carlos Alchourrón, Peter Gärdenfors, and David Makinson. On the logic of theory change : partial meet contraction and revision functions. *Journal of Symbolic Logic*, 50(2) :510–530, 1985.

[2] Philippe Besnard, Éric Grégoire, and Sébastien Ramon. Enforcing logically weaker knowledge in classical logic. In *5th International Conference on Knowledge Science Engineering and Management (KSEM'11)*, pages 44–55. LNAI 7091, Springer, 2011.

[3] Philippe Besnard, Éric Grégoire, and Sébastien Ramon. Overriding subsuming rules. In *11th European Conference on Symbolic and Quantitative Approaches to Reasoning with Uncertainty (ECSQARU'11)*, pages 532–544. LNAI 6717, Springer, 2011.

[4] James P. Delgrande and Wolfgang Faber, editors. *11th International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR'11)*, volume 6645. Springer, 2011.

[5] Thomas Eiter and Georg Gottlob. On the complexity of propositional knowledge base revision, updates, and counterfactuals. *Artificial Intelligence*, 57(2–3) :227–270, 1992.

[6] André Fuhrmann and Sven Ove Hansson. A survey of multiple contractions. *Journal of Logic, Language and Information*, 3(1) :39–76, 1994.

[7] Éric Grégoire. Fusing cooperative technical-specification knowledge components. In *14th IEEE International Conference on Tools with Artificial Intelligence (ICTAI'02)*, pages 535–542. IEEE Computer Society, 2002.

[8] Éric Grégoire. Fusing cooperative technical-specification knowledge components. *International Journal on Artificial Intelligence Tools*, 12(3) :265–278, 2003.

[9] Éric Grégoire. Knowledge refinement through revision. In *8th IEEE International Conference on Information Reuse and Integration (IRI'07)*, pages 285–290. IEEE Press, 2007.

[10] Éric Grégoire, Bertrand Mazure, and Cédric Piette. Local-search extraction of muses. *Constraints*, 12(3) :325–344, 2007.

[11] Éric Grégoire, Bertrand Mazure, and Cédric Piette. Does this set of clauses overlap with at least one mus ? In *22nd International Conference on Automated Deduction (CADE'09)*, pages 100–115. LNCS, 2009.

[12] Éric Grégoire, Bertrand Mazure, and Cédric Piette. Using local search to find msses and muses. *European Journal of Operational Research*, 199(3) :640–646, 2009.

[13] Robert C. Moore. The role of logic in knowledge representation and commonsense reasoning. In *2nd National Conference on Artificial Intelligence (AAAI'82)*, pages 428–433. AAAI Press, 1982.

[14] Allen Newell. The knowledge level. *Artificial Intelligence*, 18(1) :87–127, 1982.

[15] Bradley L. Richards and Raymond J. Mooney. Automated refinement of first-order horn-clause domain theories. *Machine Learning*, 19(2) :95–131, 1995.

[16] Stefan Wrobel. First order theory refinement. In *Advances in Inductive Logic Programming*, pages 14–33. IOS Press, 1996.

[17] Du Zhang and Éric Grégoire. The landscape of inconsistency : a perspective. *International Journal of Semantic Computing*, 5(3) :235–256, 2011.