Révision par la méthode des r-ensembles et ROBDD

Éric Würbel
Laboratoire S.I.S., Université de Toulon et du Var
Avenue de l'université
83957 La Garde cedex

Résumé *La mise en œuvre efficace d'opérateurs de révision de croyances reste un problème algorithmique ouvert, le problème de la révision étant difficile. Néanmoins on peut raisonnablement espérer trouver des représentations et des algorithmes adaptés qui soient efficaces en moyenne. Une solution pourrait passer par l'utilisation des techniques de compilation de bases de connaissances qui ont pour avantage d'essayer de se débarasser de la complexité du problème dès l'étape de représentation de la connaissance. S'appuyant sur de récents travaux permettant de représenter un préordre sur des interprétations à l'aide de ROBDDs, nous montrons qu'une famille d'opérations de révision, la révision par les r-ensembles, peut se représenter dans ce cadre. De plus, nous montrons qu'un majorant de la taille du ROBDD représentant le préordre sur les interprétations associé à cette opération de revision est en* $O\left(n^2\right)$.

1 Introduction

L'implantation en machine d'opérateurs de révision efficaces reste un problème ouvert. Le problème de la révision de croyances est un problème difficile [7, 14, 13, 16], mais on peut raisonnablement espérer trouver des représentations et des algorithmes efficaces dans le cas moyen ou pour certaines classes de problèmes.

Dans [19, 18, 21], les auteurs présentent une famille d'opérations de révision appelée « révision par la méthode des r-ensembles », permettant la révision d'un ensemble de clauses propositionnelles K par un autre ensemble de clauses propositionnelles A. Le principe de cette opération est de déterminer des sous ensembles de clauses de K à retirer de $K \cup A$ pour restaurer la cohérence. Cette approche généralise un opérateur précédemment défini dans [17]. Plusieurs solutions algorithmiques ont été proposées [21] pour la mise en œuvre de cet opérateur. Une de ces solutions utilisait les ROBDDs [4] en se basant sur un schéma de représentation de la connaissance inspiré par [11, 3, 12]. Malheureusement, cette approche s'est avérée très décevante du point de vue expérimental. Nous suspectons que ce mauvais comportement provenait du fait

que toute la connaissance contenue dans K et A était représentée dans un seul ROBDD, rendant rapidement celui-ci intractable.

Récemment, Gorogiannis and Ryan ont présenté un cadre de représentation d'opérations de changement de connaissance[1] utilisant les ROBDD [9]. Ce travail est directement inspiré des travaux de Katsuno and Mendelzon sur l'approche sémantique de la révision de croyances, puisqu'il propose une méthode permettant de représenter un préordre sur les interprétations du langage de la logique propositionnelle à l'aide d'un ROBDD.

Dans la section 2, nous exposons quelques rappels sur la logique propositionnelle, sur la révision de croyances et sur la révision par la méthode des r-ensembles en particulier. Nous introduisons notamment une caractérisation sémantique de la révision par les r-ensembles due à [2]. Nous rappelons aussi quelques propriétés utiles des ROBDD.

Puis dans la section 3 nous montrons que l'ordre sur les interprétations sous-jacent à la révision par les r-ensembles peut être représenté dans le contexte des ROBDD. Nous exposons ensuite dans la section 4 comment énumérer les r-ensembles à partir du ROBDD calculé dans la section précédente. Dans la section 5, nous donnons des estimations de la complexité dans le pire des cas, en temps et en espace, des opérations présentées. Notamment, nous montrons que, si on considère la révision d'un ensemble de clauses K par un ensemble de clauses A, le ROBDD représentant le préordre total sur les interprétations a une taille d'au plus $O\left(n^2\right)$, n étant le nombre de *clauses* de K. Enfin nous concluons et ouvrons des perspective au travaux présentés.

2 Prélambule

2.1 logique propositionnelle

Nous désignons par $\mathcal{L}$ le langage de la logique propositionnelle défini sur un ensemble de variables propositionnelles V et les connecteurs habituels ($\neg, \vee, \wedge, \rightarrow, \leftrightarrow$). On désigne par $Int(V)$ l'ensemble des interprétations. Nous écrivons $w \models \alpha$ pour dire que l'interprétation w est un modèle de la formule α. L'ensemble des modèles de α est noté $Mod(\alpha)$. La constante vrai (resp. faux) est notée $\top$ (resp. $\bot$). Un littéral est une variable $v \in V$ ou sa négation $\neg v$. Une clause est une disjonction ($\vee$) de littéraux. Un ensemble de clauses se comprend comme la conjonction ($\wedge$) des clauses qui le composent. On représente les interprétations comme des ensembles de littéraux non contradictoires, ces ensembles étant compris comme étant la conjonction de leurs éléments. Soit $D \subseteq V$, on désigne par $Lit(D)$ l'ensemble des littéraux qu'on peut construire à partir de D. Soit E un ensemble de formules. On désigne par $V(E)$ l'ensemble des variables apparaissant dans E.

Soit $D \subseteq V$. Une D-interprétation partielle w est un ensemble de littéraux non contradictoires tels que $w \subseteq Lit(D)$. Une D-interprétation par-

[1] « changement de connaissance » est ici pris dans un sens large, englobant révision, mise à jour, fusion...

tielle w telle que $\forall x \in D$, $x \in w$ ou $\neg x \in w$ est une D-interprétation. $Int_D(V)$ est l'ensemble des D-interprétations. Soit w une D-interprétation partielle. $Ext(w, D) = \{v \in Int_D(V) \mid w \subseteq v\}$ repésente l'ensemble de D-interprétations qui étendent w. Soient w une D-interprétation et ϕ une formule. w est un D-modèle de ϕ si et seulement si il existe un modèle v de ϕ tel que $w \subseteq v$. On dit que w D-satisfait ϕ et on note $w \models^D \phi$.

2.2 révision

Un état épistémique Ψ représente l'ensemble des croyances d'un agent concernant le monde qui l'entoure. Les états épistémiques servent à représenter la connaissance d'un agent intelligent. La plupart du temps, l'agent doit faire face à des informations incomplètes, incertaines ou imprécises, d'où la nécessité d'un mécanisme de changement lui permettant de prendre en compte l'évolution de ses croyances en présence de nouvelles informations. Dans le contexte de la révision de croyances, les états épistémiques ont été représentés par des ensembles de formules logiques déductivement clos [1], [8] ou par une formule [10] représentant les croyances courantes de l'agent. Mais l'état épistémique est quelque chose de plus complexe. Il ne se contente pas de représenter les croyances courantes de l'agent, mais aussi les stratégies utilisées par l'agent pour modifier ses croyances en présence d'une nouvelle information. De plus, si on considère la révision itérée, alors on se doit de trouver une repésentation des ces stratégies du changement [6].

La révision de croyances a été caractérisée avec succès par Alchourón, Gärdenfors et Makinson [1] qui ont proposé un ensemble de postulats de rationalité pour des états épistémiques consistant en des ensembles de croyances représentant les croyances courantes de l'agent. Katsuno et Mendelzon [10] ont reformulé ces postulats pour des états épistémiques représentés par une seule formule propositionnelle ψ, toute formule déductible de ψ appartenant à l'ensemble des croyances de l'agent. Darwiche et Pearl [6] ont représenté un état épistémique par un préordre total sur les interprétations de $\mathcal{L}$. Soit Ψ un état épistémique dont l'ensemble de croyances associé est $Bel(\Psi)$, et soient ϕ et μ deux formules propositionnelles quelconques, les postulats AGM sont reformulés de la façon suivante :

- $(R1)$ $Bel(\Psi \circ \mu) \models \mu$,
- $(R2)$ Si $Bel(\Psi) \wedge \mu$ est satisfaisable, $\Psi \circ \mu \equiv \Psi \wedge \mu$.
- $(R3)$ Si μ est satisfaisable, alors $Bel(\Psi \circ \mu)$ est satisfaisable.
- $(R4)$ Si $\Psi_1 = \Psi_2$ et $\mu_1 \equiv \mu_2$, alors $\Psi_1 \circ \mu_1 = \Psi_2 \circ \mu_2$.
- $(R5)$ $Bel(\Psi \circ \mu) \wedge \phi \models Bel(\Psi \circ (\mu \wedge \phi))$.
- $(R6)$ Si $Bel(\Psi \circ \mu) \wedge \phi$ est satisfaisable, alors $Bel(\Psi \circ (\mu \wedge \phi)) \models Bel(\Psi \circ \mu) \wedge \phi$.

Une opération de révision satisfaisant les postulats AGM est équivalente à un ensemble de préordres totaux sur les interprétations de $\mathcal{L}$. La notion d'assi-

gnation fidèle introduite par KATSUNO et MENDELZON [10] pour les formules est étendue aux états épistémiques. Une fonction qui associe à un état épistémique Ψ un préordre total $\leq_\Psi$ sur les interprétations de $\mathcal{L}$ est une assignation fidèle ssi pour tout $\omega, \omega' \in Int(V)$ elle vérifie les conditions suivantes :

1. Si $\omega, \omega' \models Bel(\Psi)$ alors $\omega =_\Psi \omega'$;
2. Si $\omega \models Bel(\Psi)$ $\omega' \not\models Bel(\Psi)$ then $\omega <_\Psi \omega'$;
3. Si $\Psi = \Phi$ alors $\leq_\Psi = \leq_\Phi$.

DARWICHE et PEARL [6] ont reformulé le théorème de représentation de la façon suivante :

Théorème 2.1 *Un opérateur de révision $\circ$ vérifie les postulats $(R1)$–$(R6)$ si et seulement si il existe une assignation fidèle qui associe à chaque état épistémique Ψ un préordre total $\leq_\Psi$ tel que $Mod(Bel(\Psi \circ \mu)) = \min(Mod(\mu), \leq_\Psi)$*[2].

2.3 révision par les r-ensembles

La révision par la méthode des r-ensembles est présentée par WÜRBEL & AL dans [20, 19]. Cette méthode permet la révision d'un ensemble de clauses K par un ensemble de clauses A.

Définition 1 *Soient K et A deux ensembles de clauses. R est un r-ensemble de $K \cup A$ si et seulement si R est un sous-ensemble minimal (au sens de la cardinalité) de K tel que $(K \cup A) \setminus R$ est cohérent.*

La stratégie de révision est alors la suivante :
– si $K \cup A$ est cohérent, alors le résultat de la révision est $K \cup A$;
– si $K \cup A$ est incohérent, on cherche les r-ensembles de $K \cup A$.
– On choisit un des r-ensembles à retirer de $K \cup A$.

Ainsi la notion de r-ensemble définit une famille $\mathcal{F}$ d'opérations de révision. Chaque opérateur de $\mathcal{F}$ est est défini par $K \circ_R A = (K \cup A) \setminus R$ où R est un r-ensemble de $K \cup A$.

D'un point de vue sémantique, on a les résultats suivants, dont les preuves peuvent être trouvées dans [2].

Définition 2 *Pour toute interprétation $w \in Int(V)$ et pour tout ensemble de clauses K, $NS_K(w) \subseteq K$ est l'ensemble des clauses de K non satisfaites par w. On note $|NS_K(w)|$ le cardinal de cet ensemble.*

Définition 3 *Soient $w, w' \in Int(V)$ deux interprétations. On dit que w est préférée à w' et on note $w \prec_K w'$ si et seulement si $|NS_K(w)| < |NS_K(w')|$.*

Pour la proposition qui suit, on définit les notations suivantes : Pour tout ensemble de formules $F = \{\alpha_0, \ldots, \alpha_n\}$, on note $Fpos(F)$ la formule $\alpha_0 \wedge \ldots \wedge \alpha_n$ et $Fneg(F)$ la formule $\neg\alpha_0 \wedge \ldots \wedge \neg\alpha_n$.

[2] $min(Mod(\mu), \leq_\Psi)$ contient tous les modèles de $Mod(\mu)$ qui sont minimaux d'après selon le préordre total $\leq_\Psi$.

Proposition 2.1 *Soit R un ensemble de clauses. R est un r-ensemble de $K \cup A$ si et seulement si $R \subseteq K$ et $Mod((K \cup A) \setminus R) \neq \emptyset$ et*

$$Mod((K \cup A) \setminus R) = min(Mod(A), \prec_K) \cap Mod(Fpos(K \setminus R) \wedge Fneg(R)).$$

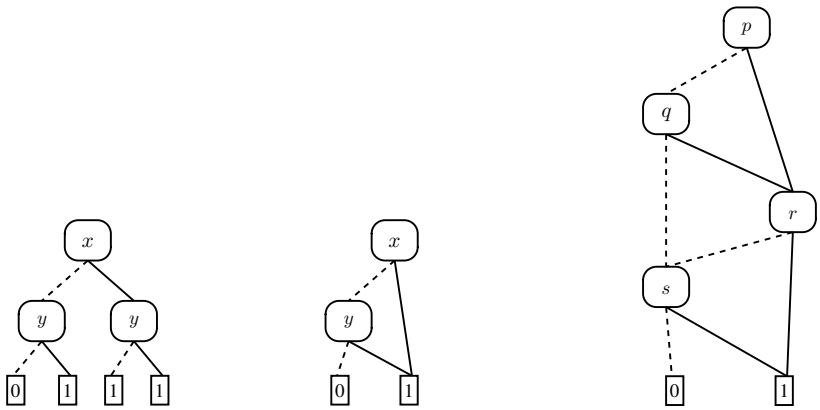
Cette proposition nous fournit donc une caractérisation sémantique d'un opérateur de révision $\circ_R$. En effet, pour tout r-ensemble R de $K \cup A$ on a $K \circ_R A = (K \cup A) \setminus R$, donc

$$\begin{aligned} Mod(K \circ_R A) &= Mod((K \cup A) \setminus R) \\ &= min(Mod(A), \prec_K) \cap Mod(Fpos(K \setminus R) \wedge Fneg(R)) \end{aligned}$$

Notons que cette stratégie de révision se rapproche de la « safe base revision » de Nebel [16]. La stratégie de cette opération est d'enlever un élément dans chaque « entailment set » de $\neg\phi$. Un ensemble $B \subseteq K$ est un entailment set pour $\neg\phi$ ssi $B \models \neg\phi$ et $\forall C \subset B\ C \not\models B$. Cela revient à enlever un élément de K dans chaque sous-ensemble minimal-incohérent de $K \cup \{\phi\}$, ce qui est une autre façon d'exprimer la révision par les r-ensemble [19]. En revanche, la « safe base revision » permet de choisir les éléments à retirer, plûtot que de proposer tous les retraits possibles, en s'appuyant sur un ordre partiel strict défini sur les formules de K et respectant l'équivalence. De plus, la notion de r-ensemble ne repose pas que sur la minimalité de l'inclusion mais aussi sur la cardinalité des ensembles. Nous ne nous étendrons pas plus sur la comparaison entre r-ensemble et « safe base revision » qui n'est pas l'objet de cet article mais qui en revanche mériterait un étude complète.

2.4 BDDs, RBDDs, ROBDDs

Un diagramme de décision binaire (BDD, Binary Decision Diagam) est une structure de données compacte et expérimentalement efficace permettant de représenter des formules de la logique propositionnelle [4]. L'arbre de décision de la formule $x \vee y$ est représenté à la figure1(a). Les lignes pointillées (resp. pleines) représentent le chemin à emprunter quand un nœud est faux (resp. vrai). Les arbres de décision codent la table de vérité des formules qu'ils représentent. De ce fait ils sont inefficaces en espace puisqu'ils contiennent $2^{n+1} - 1$ nœuds si la formule contient n variables propositionnelles. Un BDD représentant la même formule apparait à la figure 1(b). Il est obtenu en fusionnant les sous-arbres identiques de l'arbre de décision et en éliminant les nœuds redondants. Pour prendre la mesure du gain de place qu'il est possible d'obtenir avec un BDD, il suffit d'observer le BDD représentant $((p \vee q) \wedge r) \vee s$ présenté à la figure 1(c). Celui-ci contient 6 nœuds alors que l'arbre de décision correspondant en comprend 31. Dans le pire des cas, le BDD d'une formule peut contenir $O(2^n)$ nœuds, mais en pratique beaucoup de formules peuvent être représentées de façon compacte. La taille finale d'un BDD est très dépendante de l'ordre d'apparition des variables dans le BDD, mais cette discussion dépasse le cadre de cet article (voir [5]).



(a) Arbre de décision de la formule $x \vee y$

(b) BDD de la formule $x \vee y$

(c) BDD de la formule $((p \wedge q) \vee r) \wedge s$.

FIG. 1: Arbres de décision et diagrammes de décision

Quand un BDD n'a plus de nœud redondant ni de sous-diagrammes isomorphes, on dit qu'il est *réduit*. Les BDDs réduits (RBDD) sont canoniques, i.e. pour une formule donnée et un ordre sur les variables, il existe un unique RBDD corespondant. Le fait que l'ordre soit fixé fait qu'on parle de BDD *ordonné* et *réduit* (ROBDD, *Reduced Ordered Binary Decision Diagram*).

Par la suite, nous notons B_α le ROBDD représentant une formule popositionnelle α. Après avoir converti une formule en ROBDD, il est possible de manipuler et de transformer ce ROBDD en utilisant un ensemble d'opérations dont les principales sont :

- vérification de tautologie : si un ROBDD représente une tautologie, il est réduit au puit 1. Donc l'opération de vérification s'effectue en temps constant.
- satisfaisabilité : Si un ROBDD ne se réduit pas au puit 0, alors il représente une formule satisfaisable, donc le test de satisfaisabilité s'effectue en temps constant.

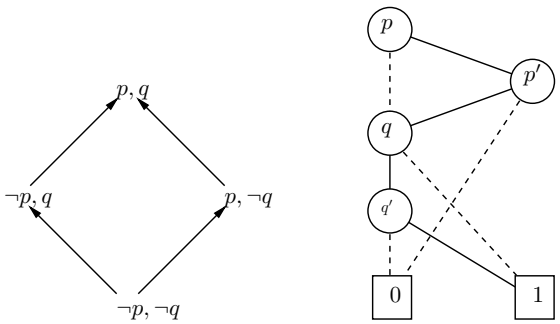
Ormis ces opérations de test, on trouve les opérations de construction et de transformation résumées dans le tableau ci-après. Les algorithmes sont notés à l'aide des symboles logiques en caractères gras. $\mathbf{p}$ et $\mathbf{p'}$ représentent des vecteurs de variables propositionnelles :

algorithme	notation	complexité	remarques
$\texttt{apply}(B_\alpha, B_\beta, \bullet)$	$B_\alpha \bullet B_\beta$	$O(\lvert B_\alpha\rvert . \lvert B_\beta\rvert)$	[4]$\bullet$ est un des connecteurs $\vee$, $\wedge$, $\rightarrow$... Calcul du ROBDD correspondant à $\alpha \bullet \beta$.
$\texttt{negate}(B_\alpha)$	$\neg B_\alpha$	$O(\lvert B_\alpha\rvert)$	Calcule le ROBDD correspondant à $\neg\alpha$. Utilisation de `apply` avec $\neg\alpha \equiv \alpha \rightarrow \bot$.
$\texttt{restrict}(C, p, B_\alpha)$	$B_\alpha[C/p]$	$O(\lvert B_\alpha\rvert)$	[4]. Substitution d'une variable propositionnelle p par une constante C ($\bot$ ou $\top$).
$\texttt{exists}(\mathbf{p}, B_\alpha)$	$\exists \mathbf{p}.B_\alpha$	$O\left(\lvert B_\alpha\rvert^{2^k}\right)$	[5]. Quantification existentielle. Cet algorithme utilise la définition suivante : $\exists p.\alpha = \alpha[\top/p] \vee \alpha[\bot/p]$.
$\texttt{forall}(\mathbf{p}, B_\alpha)$	$\forall \mathbf{p}.B_\alpha$	idem $\exists$.	[5]. Quantification universelle. Cet algorithme utilise la définition suivante : $\forall p.\alpha = \alpha[\top/p] \wedge \alpha[\bot/p]$.
$\texttt{andExists}(\mathbf{p}, B_\alpha, B_\beta)$	$\exists \mathbf{p}.(B_\alpha \wedge B_\beta)$	$O\left(\lvert B_\alpha\rvert \cdot \lvert B_\beta\rvert \cdot 2^{2n}\right)$	[15]. n est le nombre de variables apparaissant dans α et β. La conjonction n'est pas calculée explicitement.
$\texttt{impliesForall}(\mathbf{p}, B_\alpha, B_\beta)$	$\forall \mathbf{p}.(B_\alpha \rightarrow B_\beta)$	idem `andExists`	[15].
$\texttt{replace}(\mathbf{p}, \mathbf{p}', B)$	$B[\mathbf{p}'/\mathbf{p}]$	voir ci-contre	Substitution de variables. Complexité linéaire si la substitution respecte l'ordre des variables, exponentielle sinon.

Nous avons vu que les ROBDD représentent des formules en représentant l'ensemble des modèles qui les satisfont[3]. Pour construire une opération de révision, nous avons besoin de représenter des relations sur des modèles à l'aide de BDDs. Une idée présentée dans [9] constiste à voir une relation comme une fonction qui à deux interprétations associe une valeur booléenne. Cette fonction est donc représentable à l'aide d'un ROBDD sur *deux copies des variables propositionnelles*, que nous appelons primées et non primées, et que nous notons $\mathbf{p}$ et $\mathbf{p}'$ (ici encore $\mathbf{p}$ et $\mathbf{p}'$ sont des vecteurs de variables).

Exemple Considérons le préordre $\leq$ représenté à la figure 2(a) sur les 4 interprétations du langage $\{p, q\}$.

[3]Plus exactement, un ROBDD représente des modèles partiels.



(a) un préordre partiel (b) BDD représentant un préordre partiel

FIG. 2: Un préordre partiel et le ROBDD le représentant

Le ROBDD correspondant est représenté à la figure 2(b). Il s'utilise de la façon suivante : soient deux interprétations i et i', pour déterminer si $i \leq i'$, on fournit les valeurs de vérité $\mathbf{p}$ pour i et $\mathbf{p}'$ pour i' au ROBDD et on obtient une valeur booléenne en résultat.

Nous présentons rapidement quelques opérations utiles sur des ROBDD représentant des relations de préordre.

Si B_R est un ROBDD représentant une relation R sur des variables propositionnelles primées et non primées, alors le ROBDD $B_{R^{-1}}$ représentant la relation inverse R^{-1} est obtenu en renomant simultanément les variables primées en non primées, et les variables non primées en primées. Nous notons cette opération par $B_R[\mathbf{p}/\mathbf{p}', \mathbf{p}'/\mathbf{p}]$. La contrepartie stricte d'une relation R est définie par $R^< = R \cap \overline{R^{-1}}$[4]. Le ROBDD calculant cette contrepartie stricte est donné par : $\texttt{strict}(B_R) = B_R \wedge \neg(B_R[\mathbf{p}/\mathbf{p}', \mathbf{p}'/\mathbf{p}])$. Les éléments R-minimaux d'un ensemble X sont définis par $\min_R(X) = \left\{w \in X \mid \forall v \in X \; v\overline{R^<}w\right\}$ où $R^<$ est la contrepartie stricte de R. L'algorithme calculant le ROBDD $\min_R(X)$ représentant les éléments R-minimaux de X à l'aide de B_R et B_X peut s'écrire

$$\texttt{min}(B_R, B_X) = ((\forall \mathbf{p}.(B_X \rightarrow \neg \texttt{strict}(B_R)))[\mathbf{p}/\mathbf{p}']) \wedge B_X$$

Si la relation R est totale, alors l'expression des éléments R-minimaux est simplifiée en $\min_R(X) = \{w \in X \mid \forall v \in X \; wRv\}$, et l'algorithme devient

$$\texttt{min}(B_R, B_X) = (\forall \mathbf{p}'.(B_X[\mathbf{p}'/\mathbf{p}] \rightarrow B_R)) \wedge B_X.$$

3 R-ensemble, préordre et ROBDD

Notre problème est de réviser un état épistémique Ψ représenté par une formule $Bel(\Psi) = K$, K étant un ensemble de clauses, par un ensemble de

[4]La notation $\overline{R}$ désigne le complément de la relation R. Cela signifie que si R est définie sur le produit cartésien $E \times E$, on peut voir R comme un sous ensemble de $E \times E$ et dans ce cas $\overline{R} = (E \times E) \setminus R$.

clauses A. Nous avons vu dans la section 2.3 que chaque r-ensemble R de $K \cup A$ définissait une opération de révision $\circ_R$ telle que

$$Mod\left(K \circ_R A\right) \quad = \quad \min\left(Mod\left(A\right), \prec_K\right) \cap Mod\left(Fpos\left(K \setminus R\right) \wedge Fneg\left(R\right)\right).$$

Le problème de cette définition est que l'opération de révision ainsi définie nécessite la connaissance a-priori de R, ce qui est assez contre-intuitif ! Nous allons donc calculer une variante de $\min\left(Mod\left(A\right), \prec_K\right)$ s'appuyant sur un préordre différent de $\prec_K$, qui va nous permettre ensuite de déterminer tous les r-ensembles.

La comparaison entre deux interprétations dans le préordre $\prec_K$ nécessite un comptage de *clauses falsifiées*, qui est une opération *dépendant de la syntaxe* et *extérieure au langage.* Pour minimiser ce problème, au lieu de compter des clauses falsifiées, nous proposons de compter des variables propositionnelles assignées à une valeur particulière. Informellement, nous transformons chaque clause c de K en une formule $\Phi_c = (\neg c \rightarrow \phi_c) \wedge \neg\left(c \wedge \phi_c\right) = \phi_c \oplus c$[5], ϕ_c étant une variable n'apparaissant pas dans K ni dans A. Ainsi, si une interprétation w est telle que $w \models c$ (resp. $w \not\models c$), alors la satisfaction de Φ_c impose que $w \not\models \phi_c$ (resp. $w \models \phi_c$). Nous proposons la définition formelle de cette transformation ainsi qu'un réultat qui va nous servir par la suite.

Définition 4 *Soient deux ensembles de clauses $K = \{c_0, \ldots, c_n\}$ et A. On note $Tr\left(K\right)$ l'ensemble de formules construit de la façon suivante :*

$$Tr\left(K\right) = \left\{\phi_i \oplus c_i \mid c_i \in K,\, \forall i \in \{1, \ldots, n\}\right\},$$

avec $\forall i \in \{1, \ldots, n\}$, ϕ_i n'apparaît ni dans K, ni dans A. On note $\Phi_K = \{\phi_i \mid i \in \{1, \ldots, n\},\, i \in Tr\left(K\right)\}$

On a alors le résultat suivant :

Proposition 3.1 *Soient $K = \{c_0, \ldots, c_n\}$ un ensemble de clauses et sa transformation $Tr\left(K\right)$. $\forall w \in Mod\left(Tr\left(K\right)\right)$, w est un modèle de K si et seulement si $\forall i \in \{1, \ldots, n\}$, $w \not\models \phi_i$.*

Preuve: Si $w \models K$ alors $\forall i \in \{1, \ldots, n\} w \models c_i$, or par hypothèse $w \models \phi_i \oplus c_i$, donc $w \not\models \phi_i$ d'après la définition de la table de vérité du connecteur $\oplus$. De même, si $w \not\models K$, alors $\exists i \in \{1, \ldots n\}$ tel que $w \not\models c_i$, or par hypothèse $w \models \phi_i \oplus c_i$, donc $w \models \phi_i$ d'après la définition de la table de vérité du connecteur $\oplus$. ■

L'idée sous-jacente à cette proposition est que, sous réserve qu'aucun ϕ_i n'apparaisse dans K, l'ensemble des modèles de $Tr\left(K\right)$ restreints aux variables de K (i.e. sans les ϕ_i) est égal à l'union des modèles et des contre-modèles de K. C'est cette conséquence qui justifie dans la suite que nous ne travaillions que sur les modèles de $Tr\left(K\right)$.

[5] le symbole $\oplus$ désigne le connecteur « ou exclusif ».

Proposition 3.2 *Soient $K = \{c_0, \ldots, c_n\}$ un ensemble de clauses et sa transformation $Tr(K)$. Soit v un $V(K)$-modèle de K. $m \in Ext(v, Tr(K))$ est un modèle de $Tr(K)$ si et seulement si $\forall i \in \{1, \ldots, n\}$ $m \not\models \phi_i$.*

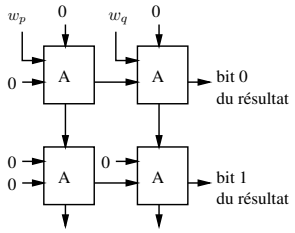
Preuve:($\Rightarrow$) : Soit $m \in Ext(v, Tr(K))$ tel que $m \models Tr(K)$. Supposons qu'il existe $i \in \{1, \ldots n\}$ tel que $m \models \phi_i$. Mais alors $m \not\models c_i$ ce qui est contradictoire avec le fait que $v \models c_i$, car $v \subseteq m$.($\Leftarrow$) : Soit $m \in Ext(v, Tr(K))$ tel que $\forall i \in \{1, \ldots, n\}$ $m \not\models \phi_i$. Il en découle que $\forall i \in \{1, \ldots, n\}$, $m \models c_i$, donc $m \models Tr(K)$. ■

Autrement dit, les modèles de $Tr(K)$ sont aussi des modèles de K si et seulement si il assignent la valeur $\bot$ aux variables ϕ_i. Par la suite, on note $\phi^-(w) = \{\phi_i \in Tr(K) \mid w \not\models \phi_i\}$ et $\phi^+(w) = \{\phi_i \in Tr(K) \mid w \models \phi_i\}$. Nous allons maintenant définir un préordre total basé sur le cardinal des ensembles $\phi^+(w)$. C'est à partir de cet ordre que nous construirons les r-ensembles.

Définition 5 *Soient $w, w' \in Mod(Tr(K))$. On définit l'ordre $\preceq_{Tr}$ par : $w \preceq_{Tr}$ w' si et seulement si $|\phi^+(w)| \leq |\phi^+(w')|$.*

On voit tout de suite l'analogie entre le comptage des clauses falsifiées par une interprétation w et le comptage des variables ϕ_i tq $w \models \phi_i$. Pour représenter cet ordre sous forme d'une fonction booléenne, dans l'esprit des travaux de Gorogiannis et Ryan [9], nous construisons un circuit constitué d'additionneurs et de soustracteurs permettant de déterminer si, étant donné deux interprétations v et w, on a $v \prec_{Tr} w$. Une première partie du circuit est chargée de compter le nombre de variables ϕ_f telles que $v \models \phi_f$. Cette première partie de circuit produit donc un résultat sous forme de nombre binaire.

Exemple Pour fixer les idées, considérons le vocabulaire constitué des variables $\{p, q\}$. Le circuit suivant prend en entrée une interpétation sur ces deux variables et produit en sortie le nombre de variables affectées à $\top$ par cette interprétation. Il est constitué de 4 additionneurs à 1 bit (les carrés marqués « A »). Chaque additionneur comprend deux opérandes en entrée (à gauche des carrés), une retenue en entrée (au dessus du carré), une retenue en sortie (en bas) et un résultat en sortie (à droite). les entrées w_p et w_q sur le schéma représentent les valeurs données à p et q par l'interprétation w.



Ensuite une deuxième partie du circuit permet de décompter, i.e. de soustraire du nombre binaire précédent le nombre de variables ϕ_f telles que $w \models \phi_f$. Si le résultat est négatif ou nul, alors $|\phi^+(v)| \leq |\phi^+(w)|$, et donc $v \preceq_{Tr} w$.

Pour détailler, le premier étage du circuit (partie gauche de la figure 3) utilise un ensemble d'additionneurs pour compter le nombre de variables ϕ_f telles que $v \models \phi_f$. Le résultat de ce premier circuit est un nombre binaire contenant au plus $k = \lceil log_2(|\Phi_K|) \rceil$, où $|\Phi_K|$ est le nombre de variables ϕ_i introduites dans $Tr(K)$. Ce calcul nécessite donc l'utilisation de $|\Phi_K| \cdot \lceil log_2(|\Phi_K|) \rceil$ additionneurs 1 bit, chaque additionneur étant un circuit comprenant en entrée deux opérandes a et b et une retenue c, et produisant en sortie une somme o et une retenue c'. Les sorties s'expriment en fonction des entrées par

$$\begin{aligned} o &= \neg(\neg(a \leftrightarrow b) \leftrightarrow c), \\ c' &= (a \wedge b) \vee (c \wedge \neg(a \leftrightarrow b)). \end{aligned}$$

On utilise ensuite (partie droite de la figure 3) un ensemble de soustracteurs permettant le décompte des variables ϕ_f telles que $w \models \phi_f$. Comme précédemment, on utilise $|\Phi_K| \cdot \lceil log_2(|\Phi_K|) \rceil$ soustracteurs 1 bit, chaque soustracteur contenant en entrée deux opérandes a et b et une retenue c et produisant en sortie un résultat o et une retenue de sortie c' :

$$\begin{aligned} o &= \neg(\neg(a \leftrightarrow b) \leftrightarrow c), \\ c' &= (b \wedge c) \vee (\neg a \wedge \neg(b \leftrightarrow c)). \end{aligned}$$

D'après cette formule, le résultat de la soustraction est négatif si et seulement si un des soustracteurs produisant le bit k génère une retenue de 1. Le résultat de la soustraction est nul si et seulement si toutes les sorties du dernier étage de soustracteurs est nul. Il suffit donc de collecter les retenues des circuits produisant le bit k, de les combiner à l'aide de portes *ou* pour obtenir le résultat de la comparaison $v \prec_{Tr} w$. Pour ce qui est de l'égalité, il suffit de vérifier que le résultat n'est pas égal à 0. Le circuit complet est représenté à la figure 3. Un tel circuit peut se représenter à l'aide d'un ROBDD noté $B_{\preceq_{Tr}}$.

4 ROBDDs et révision par les r-ensembles

Il nous reste à définir comment, à partir du ROBDD représentant le préordre $\preceq_{Tr}$, identifier les modèles correspondant à un r-ensemble. Dans le même esprit que la révision définie par une assignation fidèle, on calcule les modèles minimaux de A (l'ensemble de clauses par lequel on révise) d'après le préordre $\preceq_{Tr}$, soit $\min(Mod(A), \preceq_{Tr})$. Ces modèles peuvent être représentés par un ROBDD de la façon suivante : soit $B_{\preceq_{Tr}}$ le ROBDD défini sur deux vecteurs de variables $\mathbf{p}$ et $\mathbf{p}'$, et B_A le ROBDD représentant A. Le ROBDD représentant $\min(Mod(A), \preceq_{Tr})$ s'écrit en terme d'opérations sur les ROBDD de la façon suivante :

$$B_{min} = min(B_{\preceq_{Tr}}, B_A) = \forall \mathbf{p}'.(B_A[\mathbf{p}'/\mathbf{p}] \rightarrow B_{\preceq_{Tr}}) \wedge B_A$$

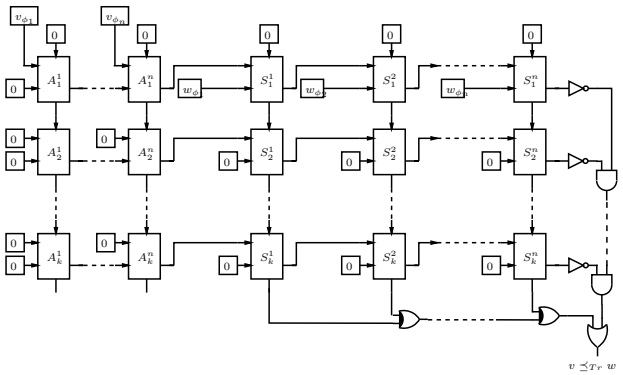


FIG. 3: Circuit calculant le résultat de $v \preceq_{Tr} w$. Dans la figure, on a, $n = |\Phi_K|$et $k = \lceil log_2(|\Phi_K|)\rceil$.

Observons que le ROBDD $B_{\preceq_{Tr}}$ représentant l'ordre $\preceq_{Tr}$ ne dépend que des variables ϕ_i. Les autres variables (celles qui viennent de K) ne sont pas concernées par le comptage. Parmi les modèles de ce ROBDD, seuls ceux qui sont des modèles de $Tr(K)$ nous intéressent, en vertu de la proposition 3.1. Plus formellement nous déterminons l'ensemble de modèles $\min(Mod(A), \leq_{Tr}) \cap Mod(Tr(K))$. En terme de ROBDD, cela revient à réaliser la conjonction de B_{min} et $B_{Tr(K)}$:

$$\begin{aligned} B_{\mathcal{R}} &= B_{min} \wedge B_{Tr(K)} \\ &= \forall \mathbf{p}'.(B_A[\mathbf{p}'/\mathbf{p}] \rightarrow B_{\preceq_{Tr}}) \wedge B_A \wedge B_{Tr(K)} \end{aligned}$$

Cette opération est nécessaire car $\preceq_{Tr}$ n'est pas une assignation fidèle. Le ROBDD résultant $B_{\mathcal{R}}$ contient tous les R-ensembles. En effet, B_{min} représente les modèles minimaux de A d'après le préordre $\preceq_{Tr}$. D'après la définition de cet ordre, chaque modèle minimal m est tel qu'il minimise le nombre de variables ϕ_i tel que $m \not\models \phi_i$. Pour produire un r-ensemble, il suffit de partitionner l'ensemble des modèles de $B_{\mathcal{R}}$ en $B_{\mathcal{R}_1}, \ldots, B_{\mathcal{R}_r}$, tel que $\forall i \in \{1, \ldots r\}$, $\forall m, m' \in Mods(B_{\mathcal{R}_i})$, $m' \neq m$, $m \models \phi_j$ ssi $m' \models \phi_j$ pour toutes les variables ϕ_j. Chaque $B_{\mathcal{R}_i}$ correspond alors à un r-ensemble.

5 Résultats de complexité

Pour déterminer un majorant de la complexité dans le pire des cas de l'opération de révision précédemment décrite, nous commençons par calculer ce majorant pour la construction du ROBDD $B_{\preceq_{Tr}}$ représentant $\preceq_{Tr}$. Pour cela, nous avons besoin de quelques résultats concernant la complexité des circuits.

Soit ϕ une fonction booléenne n-aire. Supposons qu'on ait un circuit permettant le calcul de ϕ. Ce circuit contient m blocs, chaque bloc étant une porte

(binaire ou autre) ou une entrée (les entrées sont des blocs ne possédant pas d'entrée et une sortie). On appelle ordre linéaire sur un circuit une numérotation des blocs de 1 à m pour laquelle le bloc produisant la sortie est numéroté en dernier. On appelle *section avant* au bloc i le nombre total de connexions allant de la sortie d'un bloc j, avec $j < i$, à l'entrée d'un bloc k avec $i \leq k$. La *largeur avant* w_f *du circuit* (en respectant l'ordre linéaire choisi) est définie comme étant la plus grande section avant de tous les blocs. De façon similaire, la *section arrière* au bloc i est le nombre total de connexions allant de la sortie d'un bloc j, avec $j > i$, à l'entrée d'un bloc k avec $i \geq k$. La *largeur arrière* w_r *du circuit* (en respectant l'ordre linéaire choisi) est définie comme étant la plus grande section arrière de tous les blocs. On a alors le théorème suivant :

Théorème 5.1 ([15]) *Si un circuit calculant une fonction booléenne ϕ possède une section avant w_f et une section arrière w_r pour un ordre linéaire L, alors il existe un ROBDD représentant la fonction booléenne ϕ dont la taille est bornée par $n2^{w_f 2^{w_r}}$, où n est le nombre d'entrées du circuit.*

Dans notre cas, le circuit servant à représenter l'ordre $\preceq_{Tr}$ admet un ordre topologique sur ses blocs, i.e. un ordre tel que $w_r = 0$. Ainsi, la borne donnée par le théorème 5.1 devient $n2^{w_f}$.

En utilisant ces résultats, nous sommes maintenant capables de calculer une borne supérieure de la complexité dans le pire des cas du circuit représentant le préordre total $\preceq_{Tr}$.

Proposition 5.1 *Il existe un ROBDD de taille $O\left(|\Phi_K|^2\right)$ représentant le préordre total $\preceq_{Tr}$.*

Preuve: En premier lieu nous choisissons un ordre linéaire sur les blocs du circuit :

$$
\begin{array}{cccccc}
v_{\phi_1}, & A_1^1, & \ldots, & A_k^1, & & \\
v_{\phi_2}, & A_1^2, & \ldots, & A_k^2, & & \\
\vdots & \vdots & & \vdots & & \\
v_{\phi_n}, & A_1^n, & \ldots, & A_k^n, & & \\
w_{\phi_1}, & S_1^1, & \ldots, & S_k^1, & & \\
w_{\phi_2}, & S_1^2, & \ldots, & S_k^2, & \vee, & \\
\vdots & \vdots & & \vdots & \vdots & \\
w_{\phi_n}, & S_1^n, & \ldots & S_k^n, & \vee, & \\
 & \neg, & & \neg, & & \\
 & \wedge, & & \wedge, & \vee &
\end{array}
$$

D'après cet ordre, pour tout bloc, la section avant est au plus de $k + C$, et nous avans vu précédemment que $k = \lceil log_2(|\Phi_K|)\rceil$. Le théorème 5.1 nous donne alors un majorant égal à $n2^{w_f}$, soit dans notre cas $2\,|\Phi_K|\,2^{\lceil log_2(|\Phi_K|)\rceil} =$

$O\left(|\Phi_K|^2\right)$. ■

Si on appelle $\mathbf{assign}_{\preceq_{Tr}}$ la fonction qui construit $B_{\preceq_{Tr}}$ et qu'on note $\left|\mathbf{assign}_{\preceq_{Tr}}\right|$ la complexité temporelle de l'opération $\mathbf{assign}_{\preceq_{Tr}}\left(B_{Tr(K)}\right)$, elle n'utilise que l'algorithme `apply` et donc elle est polynomiale en $\left|B_{Tr(K)}\right|$ (nombre de nœuds du ROBDD $B_{Tr(K)}$).

Concernant le ROBDD

$$B_{\mathcal{R}} \quad = \quad \forall \mathbf{p}'.\,(B_A\,[\mathbf{p}'/\mathbf{p}] \longrightarrow B_{\preceq_{Tr}}) \wedge B_A \wedge B_{Tr(K)},$$

la complexité temporelle de construction et la complexité en espace du ROBDD résultant se calculent ainsi :

opération	complexité temporelle	taille du ROBDD résultant
$B_A\,[\mathbf{p}'/\mathbf{p}]$	$O\left(\lvert B_A\rvert\right)$	$O\left(\lvert B_A\rvert\right)$
$B_{\preceq_{Tr}}$	$O\left(\left\lvert B_{Tr(K)}\right\rvert\right)$	$O\left(\lvert\Phi_K\rvert^2\right)$
$\forall \mathbf{p}'.\,(\cdot \longrightarrow \cdot)$	$O\left(\lvert B_A\rvert.\lvert\Phi_K\rvert^2.2^{4n}\right)$	$O\left(2^n\right)$
$B_A \wedge \cdot$	$O\left(\lvert B_A\rvert.2^n\right)$	$O\left(2^n\right)$
$B_{Tr(K)} \wedge \cdot$	$O\left(\left\lvert B_{Tr(K)}\right\rvert.2^n\right)$	$O\left(2^n\right)$

Malgré tout, ce majorant ne donne pas une idée précise de la complexité dans une situation réelle de révision car :

- Cette borne décrit le pire des cas, et le cas moyen est très complexe à formuler ;
- La taille du ROBDD représentant le préordre est très petite, donc si la taille du ROBDD B_A est acceptable dans le cas moyen, la complexité de l'opération devrait être aussi acceptable ;
- $|\Phi_K|$ est petit par rapport à la taille de toute la connaissance manipulée.

6 Conclusion

Dans cet article, nous avons montré que la famille d'opérateurs de révision par les r-ensembles, qui se base sur un critère syntaxique de cardinalité, est implantable en utilisant les ROBDD. De plus, nous avons montré que la complexité de l'opération transformant le préordre total sur les interprétations sous-jacent à l'opération de révision par les r-ensembles en un ROBDD est faible en terme de temps, et que le ROBDD ainsi construit possède une taille qui est une fonction quadratique d'un nombre de sous-formules de la base de croyances initiale.

Remarquons que dans notre approche, la transformation de chaque clause $c \in K$ en une formule $\phi_c \oplus c$ permet d'envisager un cadre plus général d'utilisation. En effet, rien n'empêche à partir d'un ensemble $F = \{\alpha_1, \ldots, \alpha_n\}$ de formules α_i quelconques d'appliquer la même transformation et d'utiliser la même stratégie de révision.

Le travail futur ce sujet portera dans un premier temps sur une expérimentation sur une application. Cette phase est nécessaire pour valider l'intéret d'utiliser une technique de compilation de base de connaissances, puisque les seuls résultats de majoration de la complexité dans le pire des cas ne permettent pas de conclure de façon optimiste. De plus, nos travaux sur la révision prennent place dans le projet européen REVIGIS (REVIsion in Geographical Information Systems) dont la thématique est l'utilisation de la révision dans le cadre de l'information géographique. Ce projet comprend plusieurs applications qui servent de base de test pour les opérations de révision. Ces applications manipulent des sources de données de taille importante (plusieurs dizaines de milliers de clauses) et sont donc de bonnes candidates pour effectuer des tests de performance.

Nous envisageons aussi de comparer plus finement la révision par les r-ensembles à la « safe base revision » de Nebel [16], ainsi qu'il a été dit à la fin de la section 2.3.

Dans un deuxième temps, nous souhaitons considérer le cas où la connaissance contenue dans A — la formule par laquelle on révise — est ordonnée elle aussi. Cela permettrait ensuite d'appliquer le cadre de représentation des préordres à l'aide de ROBDD à la fusion.

Remerciements

Ce travail rentre dans le cadre du projet européen REVIGIS IST 1999-14189.

Références

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

[2] Jonathan Bennaim. Révision d'une base de croyances dans le cadre de la programmation logique avec négation par l'échec : application à l'information géographique. Mémoire de DEA, Université de Toulon et du Var, 2003.

[3] Fabrice Bouquet and Philippe Jégou. Solving over-constrained CSP using weighted OBDDs. In Michael Jampel, Eugene Freuder, and Michael Maher, editors, *Over-Constrained Systems*, volume 1106 of *Lecture Notes in Computer Science*, pages 293–308. Springer-Verlag, 1996.

[4] Randal E. Bryant. Graph-based algorithms for boolean function manipulation. *IEEE Transactions on computers*, C-35(8) :677–691, Aout 1986.

[5] Randal E. Bryant. Symbolic boolean manipulation with ordered binary decision diagrams. *ACM Computing Surveys*, 24(3) :293–318, 1992.

[6] Adnan Darwiche and Judea Pearl. On the logic of iterated revision. *Artificial Intelligence*, 89 :1–29, 1997.

[7] T. Eiter and G. Gottlob. On the complexity of propositional knowledge base revision, updates and counterfactual. *Artificial Intelligence*, 57 :227–270, 1992.

[8] Peter Gärdenfors. *Knowledge in Flux : Modelling the Dynamics of Epistemic States*. MIT Press, 1988.

[9] Nikos Gorogiannis and Mark D. Ryan. Implementation of belief change operators using bdds. *Studia Logica*, 70 :131–156, 2002.

[10] H. Katsuno and A. Mendelzon. Propositional knowledge base revision and minimal change. *Artificial Intelligence*, 52 :263–294, 1991.

[11] J. De Kleer. An assumption-based tms. *Artificial Intelligence*, 28 :127–162, 1986.

[12] Marie-Christine Lagasquie-Schiex. *Contribution à l'étude des relations d'inférence non-monotone combinant inférence classique et préférences*. PhD thesis, Université Paul Sabatier, Toulouse, IRIT, Université Paul Sabatier, 118 route de Narbonne, 31062 Toulouse Cedex, Décembre 1995.

[13] Paolo Liberatore. The complexity of iterated belief revision. In *Proceedings of the Sixth International Conference on Database Theory (ICDT'97)*, pages 276–290, 1997.

[14] Paolo Liberatore and Marco Schaerf. The complexity of model checking for belief revision and update. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence (AAAI'96)*, pages 556–561. AAAI Press/The MIT Press, 1996.

[15] K. L. Mc Millan. *Symbolic Model Checking*. Kluwer Academic, 1993.

[16] Bernhard Nebel. *Handbook of Defeasible Reasoning and Uncertainty Management Systems*, volume 3 : Belief Change, chapter How hard is it to revise a belief base ?, pages 77–145. Kluwer Academic, Derdrecht, The Netherlands, 1998.

[17] Odile Papini. A complete revision function in propositionnal calculus. In B. Neumann, editor, *Proceedings of ECAI92*, pages 339–343. John Wiley and Sons. Ltd, 1992.

[18] Eric Würbel. *Révision de connaissances géographiques*. PhD thesis, Université de Provence, LIM-CMI, 39 avenue Joliot-Curie 13453 Marseille cedex 13, Décembre 2000.

[19] Eric Würbel, Robert Jeansoulin, and Odile Papini. Revision : An application in the framework of gis. In Anthony G. Cohn, Fausto Giunchiglia, and Bart Selman, editors, *Proceedings of the Seventh Internationnal Conference about Principles of Knowledge Representation and Reasoning, KR2000*, pages 505–516, Breckenridge, Colorado, USA, April 2000. KR, inc., Morgan Kaufmann.

[20] Eric Würbel, Robert Jeansoulin, and Odile Papini. Révision : une application dans le cadre des systèmes d'information géographiques. In Département Traitement du Signal et des Inamges de Télécom Paris CNRS

URA 820, editor, *Actes du douxième congrès francophone de Reconnaissance des Formes et Intelligence artificielle RFIA2000*, volume 1, pages 195–204, Paris, France, Février 2000. AFRIF et AFIA.

[21] Eric Würbel, Robert Jeansoulion, and Odile Papini. Spatial information revision : A comparision between 3 approaches. In Salem Benferhat and Philippe Besnard, editors, *Symbolic and Quantitative Approaches to Reasoning with Uncertainty, 6th European Conference, ECSQARU 2001*, number 2143 in Lecture Notes in Artificial Intelligence, pages 454–465, Toulouse, France, September 2001. Springer.