HAL
open science

Application de la théorie de la révision des connaissances au raisonnement à partir de cas

Julien Cojan

To cite this version:

Julien Cojan. Application de la théorie de la révision des connaissances au raisonnement à partir de cas. Intelligence artificielle [cs.AI]. Université Henri Poincaré - Nancy I, 2011. Français. NNT : . tel-00646841v2

HAL Id: tel-00646841

https://theses.hal.science/tel-00646841v2

Submitted on 22 Jan 2012

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Nancy-Université
Université Henri Poincaré

Département de formation doctorale en informatique **École doctorale IAEM Lorraine**
UFR Sciences & Technologies

Application de la théorie de la révision des connaissances au raisonnement à partir de cas

THÈSE

pour l'obtention du

Doctorat de l'université Henri Poincaré – Nancy 1

(spécialité informatique)

par

Julien Cojan

Composition du jury

Rapporteurs :	Eyke Hüllermeier	Professeur à l'Université de Marbourg, Allemagne
	Pierre Marquis	Professeur à l'Université d'Artois
Examinateurs :	Adam Cichon	Professeur à l'Université Henri Poincaré, Nancy 1
	Jérôme Lang	D.R. CNRS, LAMSADE, Université Paris-Dauphine
	Jean Lieber	M.C à l'Université Henri Poincaré, Nancy 1
	Alain Mille	Professeur à l'Université Claude Bernard, Lyon 1
	Amedeo Napoli	D.R. CNRS, LORIA, Nancy
	Paul Zimmermann	D.R. INRIA, INRIA Nancy - Grand Est

Laboratoire Lorrain de Recherche en Informatique et ses Applications — UMR 7503

Loria
Laboratoire lorrain de recherche en informatique et ses applications

Remerciements

Je tiens à remercier les membres du jury pour avoir accepté d'examiner ma thèse, et particulièrement Eyke Hüllermeier et Pierre Marquis pour leur relecture de mon manuscrit et leurs remarques. Je suis aussi reconnaissant à Paul Zimmermann pour sont investissement dans son rôle de tuteur de thèse. Merci aussi à Adam Cichon d'avoir accepté de présider ce jury, à Jérôme Lang que j'ai eu plaisir de connaître à cette occasion, à Alain Mille pour les discussions que nous avons eues à propos du RÀPC, de la combinaison de cas et des traces, et à Amedeo pour ses conseils pendant la thèse.

Je remercie bien sûr énormément mon directeur de thèse, Jean Lieber, qui m'avait déjà encadré en stage de master et qui m'a donné envie de faire cette thèse. J'ai aimé la manière naturelle avec laquelle il m'a amené à aborder des sujets aussi variés sans perdre de vue les applications. Et je suis heureux qu'il prenne bien les contre-exemples qui démolissent nos idées (avec juste un grand soupir), apparemment je lui ai souvent fait ce coup là.

Un grand merci aussi aux personnes impliquées dans TAAABLE, en particulier Amélie et Emmanuel pour leur énergie et l'aide qu'ils m'ont apportée pour mes travaux. J'en profite pour m'excuser auprès d'Emmanuel pour avoir essayé plusieurs fois de donner son tablier aux fans de TAAABLE.

Je souhaite remercier les membres de l'équipe Orpailleur avec qui j'ai eu le plaisir de travailler et qui ont pris le temps de venir aux répétitions de présentations me faire part de leurs commentaires. J'ai aussi passé de bons moments de détente avec la bande de thésards de l'équipe et d'autres joyeux collègues, Fadi, Medhi, Stéphane, Pierre, Yesmine, Florent, Thomas, Rokia, Valmi, Emmanuelle, Julien et j'oublie sûrement d'autres personnes. De quoi faire de bonnes parties de Loup Garou.

I want to thank as well my fellow office mates Violeta, Laz and Vish that kindly gave my the title of "Dr. to be" so that I would not feel too overwhelmed among such excellence.

Je remercie aussi mes parents pour leur soutient durant ces années de thèse et leur intérêt pour mes travaux, mais aussi pour me faire partager leurs travaux les fins de semaine, ce qui m'a aidé à relativiser la difficulté à rédiger une thèse.

Et pour finir, je voudrait remercier ma très chère Susu qui a partagé avec moi ces années de thèse et qui m'a gentiment incité à m'arrêter à quatre ans. Je remercie aussi la clémence de Dr. Su, puisque qu'elle s'est retenue de poser la question qui tue à la fin de la soutenance.

Table des matières

Introduction **1**

Chapitre 1 Préliminaires formels **5**

1.1 Représentation des connaissances 5

1.1.1 Syntaxe 5

1.1.2 Sémantique 6

1.2 Pré-ordres et « distances » 8

1.2.1 Pré-ordres 8

1.2.2 « Distance » 11

Chapitre 2 Révision et changement minimal des connaissances **15**

2.1 La révision et autres problèmes de changement minimal de connaissances 15

2.1.1 Révision de connaissances 15

2.1.2 Fusion des connaissances 16

2.1.3 Contraction 18

2.1.4 Mise à jour de connaissances 18

2.1.5 Débogage d'ontologies 19

2.2 Théorie AGM de la révision des connaissances 19

2.2.1 Postulats de la révision en logique propositionnelle 20

2.2.2 Assignation Fidèle 20

2.2.3 Opérateur de révision de Dalal et une généralisation 22

2.3 Révision dans d'autres formalismes 25

2.3.1 Généralisation des assignations fidèles 26

2.3.2 Opérateurs de révision définis à l'aide d'une « distance » 28

2.3.3 Révision sur l'adhérence 28

2.3.4 Révision floue 30

2.3.5 Autres travaux concernant la révision dans d'autres formalismes 33

2.4 Un opérateur de révision dans un formalisme attributs-valeurs simples 35

2.4.1 Formalisme attributs-valeurs simples 35

2.4.2 Réduction sous hypothèses à un problème d'optimisation linéaire 35
2.5 Un opérateur de révision dans la logique de descriptions $\mathcal{ALC}$ 39
2.5.1 Préliminaires concernant la logique de descriptions $\mathcal{ALC}$ 40
2.5.2 Une procédure de déduction classique pour $\mathcal{ALC}$: la méthode des tableaux 43
2.5.3 Algorithme de révision sur des ABox (modulo une TBox) 47
2.5.4 Propriétés de l'algorithme 55
2.5.5 Autres approches de la révision sur des logiques de descriptions 58
2.6 Perspectives concernant la révision en $\mathcal{ALC}$ 59
2.6.1 Exemples problématiques 60
2.6.2 Vers une définition sémantique 62
2.6.3 Vers la révision d'une BC quelconque (ABox et TBox non vides) 67
2.6.4 Exploitation d'un autre moteur d'inférences existant 69
2.7 Fusion contrainte 69
2.7.1 Définition 70
2.7.2 Exemple d'opérateur de fusion contrainte 72
2.7.3 Calcul de la fusion 75
2.7.4 Choix d'une fonction d'agrégation 77

Chapitre 3 L'adaptation par révision 79
3.1 Préliminaires 79
3.1.1 Le RÀPC : principes généraux 79
3.1.2 Connaissances exploitées par le RÀPC 80
3.1.3 Représentation des connaissances 82
3.1.4 L'adaptation en RÀPC 85
3.2 Définition de l'adaptation par révision 87
3.2.1 Propriétés de l'adaptation par révision 88
3.2.2 Adaptation conservatrice 89
3.2.3 Adaptation par révision et remémoration guidée par l'adaptation 91
3.2.4 Adaptation par mise à jour 92
3.3 Lien avec d'autres approches de l'adaptation 93
3.3.1 L'adaptation par règles et l'adaptation par révision 94
3.3.2 L'adaptation différentielle et l'adaptation par révision 98
3.4 Combinaison de cas par fusion de connaissances 104
3.4.1 Définition et propriétés 104
3.4.2 Exemple de combinaison de cas 105
3.4.3 Lien avec le CCBI 108
3.4.4 Lien avec d'autres approches de combinaison de cas en RÀPC 111

3.4.5 Perspectives 112

Chapitre 4 Application de l'adaptation par révision à TAAABLE **117**

4.1 Présentation de TAAABLE 117

4.1.1 Le *Computer Cooking Contest* 117

4.1.2 TAAABLE 1 118

4.1.3 TAAABLE 2 119

4.1.4 TAAABLE 3 122

4.2 Contribution à TAAABLE 3 : adaptation des quantités d'ingrédients 124

4.2.1 Formalisme de représentation 125

4.2.2 Formulation des connaissances du domaine et du cas source 126

4.2.3 Formulation du cas cible 129

4.2.4 Choix d'une distance et calcul 130

4.3 Contributions possibles à des versions futures de TAAABLE 130

Conclusion **133**

Index **137**

Bibliographie **141**

Introduction

La conception de systèmes à base de connaissances résolvant des problèmes posés par l'utilisateur fait partie des problématiques de l'intelligence artificielle.

Suivant les domaines d'application, différents modèles de raisonnements peuvent être utilisés. Plusieurs modèles, dont les systèmes à base de règles, supposent que les connaissances nécessaires à la résolution de problèmes soient entièrement formalisées. Cependant, tous les domaines d'application ne se prêtent pas à une formalisation complète des connaissances expertes, notamment lorsque l'ensemble des situations est complexe. C'est le cas lorsque de nombreux paramètres peuvent interférer avec une solution, par exemple en médecine, certaines caractéristiques d'un patient, pourtant étrangères à la maladie traitée, peuvent provoquer des contre-indications envers un traitement généralement recommandé pour cette maladie. C'est le cas aussi lorsqu'il y a une part de subjectivité dans l'évaluation de solutions, par exemple en cuisine, il est très difficile (sinon impossible) de formaliser des connaissances qui permettent de déterminer si une combinaison de goûts plaît.

Le raisonnement à partir de cas (RÀPC) est un modèle de raisonnement exploitant l'expérience de résolution de problèmes pour en résoudre de nouveaux. Ces expériences sont des connaissances à propos d'une situation particulière, et il n'est pas fait d'effort *a priori* pour généraliser ces connaissances à d'autres situations. Cet effort est effectué lors de la résolution d'un nouveau problème, une expérience jugée pertinente par rapport à ce problème est sélectionnée puis elle est adaptée afin de tenir compte des différences entre les situations.

Un parallèle peut être fait entre le processus d'adaptation d'expériences en RÀPC et la théorie de la révision des connaissances. Celle-ci traite des modifications à apporter à des connaissances afin de les rendre cohérentes avec d'autres connaissances dans le but de les intégrer en un tout cohérent. Dans les deux cas, des connaissances sont amenées à être modifiées afin de satisfaire de nouvelles contraintes. Cette thèse étudie les liens entre le RÀPC et la théorie de la révision des connaissances.

L'adaptation en raisonnement à partir de cas

Le raisonnement à partir de cas (RÀPC) est un type de raisonnement par analogie appliqué à la résolution de problèmes. Des expériences de résolution de problèmes, les cas sources, sont utilisées pour en résoudre de nouveaux, les cas cibles.

> Par exemple, dans le domaine culinaire, une recette de cuisine représente une expérience de confection de plats. Supposons que l'on veuille cuisiner une mousse au chocolat sans œuf mais que l'on ne trouve pas de recette satisfaisant cette requête. Avec quelques connaissances culinaires on peut s'aventurer à en créer une en s'inspirant d'une recette traditionnelle qui utilise du blanc d'œuf monté en neige pour faire la mousse, et les jaunes comme liant. On peut par exemple remplacer les blancs d'œufs par du tofu soyeux et les jaunes par une autre matière grasse, par exemple du beurre.

Dans cet exemple, il a fallu modifier un cas source, la recette traditionnelle de mousse au chocolat, afin de la rendre cohérente avec le problème à résoudre et obtenir une solution satisfaisante. Cette opération correspond à l'étape d'adaptation du RÀPC tel que décrit dans [Aamodt et Plaza, 1994].

La révision de connaissances

Dans plusieurs domaines, les connaissances disponibles sont imparfaites et peuvent être amenées à être corrigées par des connaissances jugées plus fiables. C'est le cas lorsque ces connaissances sont obtenues par extrapolations à partir d'observations, et où de nouvelles observations peuvent remettre en cause les connaissances établies précédemment. C'est le cas aussi lorsque des connaissances établies dans un contexte sont exploitées dans un autre contexte. Ce processus de correction de connaissances par d'autres connaissances jugées plus fiables est appelé révision : les premières connaissances sont révisées par les secondes, lorsqu'il y a conflit, les premières connaissances sont modifiées afin de restaurer la cohérence de l'ensemble des connaissances. Il y a souvent plusieurs résultats possibles lors d'une révision.

> On considère l'exemple classique des oiseaux, où les connaissances de départ établissent que : « Tous les oiseaux peuvent voler, et les manchots, les rapaces et les autruches sont des oiseaux. » Une nouvelle information nous apprend que : « Max est un manchot qui ne peut pas voler. » Une révision possible consiste à renoncer au fait que les manchots sont des oiseaux, une autre révision consiste à renoncer au fait que tous les oiseaux peuvent voler. Il peut aussi y avoir différents niveaux de finesse dans les corrections : dans le cas où on remet en cause le fait que tous les oiseaux peuvent voler, conserve-t-on le fait que les rapaces et les autruches peuvent voler, puisque ces connaissances ne sont pas remises en cause directement ? On retiendrait alors que « À part les manchots qui sont des oiseaux ne pouvant pas voler, tous les oiseaux peuvent voler. » Mais on pourrait aussi retenir que « À part Max qui est un manchot ne pouvant pas voler, tous les oiseaux peuvent voler. »

La théorie de la révision traite de la caractérisation des opérations de révision à travers des propriétés attendues de ces opérations et la représentation des choix qui leur sont associés.

L'adaptation par révision

L'exemple de l'adaptation de la recette de mousse au chocolat peut être mis sous la forme d'un scénario de révision :

> Garuda peut voler, mais il est allergique à l'œuf. Il y a de la mousse au chocolat en dessert, il s'imagine alors qu'elle a été faite suivant une recette classique et qu'elle contient de l'œuf, donc pas de dessert pour lui. Mais Horus, qui a préparé cette mousse, le rassure, cette mousse au chocolat a été préparée sans œuf. Comme Garuda a quelques connaissances sur les substituants d'œuf, il peut se faire une idée de la manière dont a été préparée cette mousse au chocolat. La mousse a pu être obtenue avec du soja soyeux battu qui a une texture proche des blancs montés en neige (il n'est pas très ferme, et on ne peut pas le cuire). Et à la place des jaunes, qui sont constitués principalement de lipides, Horus peut avoir mis du beurre fondu.

Le raisonnement de Garuda est une révision de sa connaissance de départ à propos de la réalisation de la mousse au chocolat par le fait que Horus n'a pas utilisé d'œuf. On s'attend à un résultat correspondant à celui de l'adaptation, c'est-à-dire au raisonnement effectué par Horus. Cela nous incite à considérer l'adaptation comme une forme de révision appliquée à des connaissances particulières : l'expérience de résolution de problèmes. L'intérêt de ce rapprochement est de fournir un cadre formel à l'adaptation en proposant une approche de l'adaptation définie à partir de la révision. En effet le RÀPC est à l'origine une théorie en sciences cognitives [Riesbeck et Schank, 1989] pour laquelle il n'y a pas de théorie formelle logique qui fasse consensus, en particulier pour l'adaptation. Diverses approches de l'adaptation ont été proposées, elles correspondent à différents types de connaissances d'adaptation. Mais il n'existe aucun cadre unifié permettant d'exploiter ces différentes connaissances d'adaptation ensemble. Nous proposons ici une formalisation de l'adaptation, l'adaptation par révision, dont nous voulons déterminer la portée. C'est-à-dire que nous voulons déterminer quelles approches de l'adaptation se ramènent à des problèmes de révision, et quelles sont les propriétés qui les caractérisent. En particulier, l'aspect minimal du changement, qui est caractérisé par plusieurs propriétés souvent associées à la révision, est-il pertinent pour l'adaptation ? Nous défendons l'idée que la plupart des approches de l'adaptation minimisent un « effort d'adaptation » qui leur est associé, notamment lorsque le RÀPC est le seul recours pour trouver une solution et n'est donc pas guidé par un autre système de résolution. Cette idée suit le principe que l'on retrouve souvent en RÀPC, par exemple dans [Kolodner, 1993] et qui dit qu'une solution a d'autant plus de chances d'être satisfaisante qu'elle a été obtenue par une adaptation nécessitant peu de modifications.

Plan du mémoire

Le chapitre 2 sera consacré en premier lieu à un état de l'art de la théorie du changement minimal des connaissances et en particulier de la révision des connaissances. En vue de l'application de la révision à l'adaptation en RÀPC nous nous intéressons aux opérateurs sur des formalismes

souvent utilisés par les applications du RÀPC. Nous proposons pour cela un opérateur de révision sur un formalisme attributs-valeurs simples (numériques, booléennes, etc.) très utilisé en RÀPC, et des travaux préliminaires à la définition d'un opérateur de révision sur une logique de descriptions dont l'intérêt est l'expressivité et le lien avec des standards de représentation des connaissances du Web sémantique.

Les fondements de cette thèse sont présentés au chapitre 3, à savoir la définition d'une approche de l'adaptation à partir de la révision et l'étude de ses propriétés. Nous établirons aussi des correspondances avec d'autres approches de l'adaptation. Cela nous amènera à proposer un cadre formel couvrant ces approches et permettant de les intégrer en un seul mécanisme d'adaptation. Une extension de l'adaptation à la combinaison de cas, exploitant plusieurs cas sources au lieu d'un seul, est définie ensuite à partir d'une généralisation de la révision, la fusion contrainte.

Le chapitre 4 traite de l'application des travaux de cette thèse au système TAAABLE. Il s'agit d'un système développé dans le cadre du *Computer Cooking Contest* dont le but est de répondre à des requêtes portant sur des recettes de cuisine à partir d'une base de recettes fournie par les organisateurs du concours. TAAABLE exploite différentes technologies, dont un moteur de raisonnement à partir de cas. La contribution principale de cette thèse à TAAABLE consiste en la mise en œuvre de l'adaptation par la révision dans un formalisme attributs-valeurs numériques pour l'adaptation des quantités d'ingrédients. Il est envisagé dans une version future de représenter les connaissances exploitées par TAAABLE dans une logique de descriptions. Cela permettra d'appliquer l'adaptation par révision dans ce formalisme. Plusieurs travaux en cours portent sur l'adaptation dans TAAABLE. L'adaptation par révision devrait permettre d'intégrer les approches concurrentes en un seul mécanisme d'adaptation. Pour cela ces différentes approches de l'adaptation devront être formalisées sous forme d'adaptation par révision.

Chapitre 1

Préliminaires formels

1.1 Représentation des connaissances

Nous utilisons le terme « connaissances » comme traduction du terme « belief » en anglais, car le mot « croyance » nous semble trop connoté religieusement. Nous ne faisons pas la distinction entre « connaissances » et « croyances », cette distinction est censée représenter deux niveaux de fiabilité, les connaissances étant irréfutables contrairement aux croyances. Nous considérons plutôt qu'il s'agit dans tous les cas de connaissances avec différents degrés de fiabilité.

Dans la suite, nous considérerons des connaissances représentées dans une logique formée d'un *langage* $\mathcal{L}$ qui donne la *syntaxe* de la représentation des connaissances (section 1.1.1), et d'une *sémantique* (section 1.1.2).

1.1.1 Syntaxe

Habituellement, $\mathcal{L}$ est structuré suivant une grammaire qui lui donne une définition récursive à partir d'éléments atomiques et de connecteurs logiques. Les élément de $\mathcal{L}$ sont appelés *formules*. Les connaissances sont représentées par des ensembles finis de formules, les *bases de connaissances* (BC).

Logique propositionnelle - syntaxe

Le langage d'une logique propositionnelle finie est définie à partir d'un ensemble fini de variables propositionnelles $\{p_1, \ldots, p_n\}$ et des connecteurs logiques $\wedge$ (et), $\neg$ (non) :

$$\begin{aligned} \psi \in \mathcal{L} \quad & \text{si} \quad \psi \in \{p_1, \ldots, p_n\} \\ & \text{ou si} \quad \psi = \neg\varphi \text{ avec } \varphi \in \mathcal{L} \\ & \text{ou si} \quad \psi = \varphi \wedge \mu \text{ avec } \varphi, \mu \in \mathcal{L} \end{aligned}$$

Pour simplifier les formules, on utilisera aussi les connecteurs $\vee$ (ou) et $\Rightarrow$ (implique) définis à partir de $\wedge$ et $\neg$, pour deux formules $\varphi, \mu \in \mathcal{L}$: $\varphi \vee \mu = \neg(\neg\varphi \wedge \neg\mu)$ et $\varphi \Rightarrow \mu = \neg(\varphi \wedge \neg\mu)$.

Exemple 1.1.1. *L'exemple des oiseaux (introduction, page 2) peut être exprimé de façon simplifiée en logique propositionnelle avec les variables* $\{\texttt{vole}, \texttt{oiseau}, \texttt{manchot}\}$ *servant à décrire Max, et les formules* ψ_{ex} *et* μ_{ex} *représentant respectivement les connaissances initiales et les nouvelles connaissances :*

$$\psi_{ex} = (\texttt{oiseau} \Rightarrow \texttt{vole}) \wedge (\texttt{manchot} \Rightarrow \texttt{oiseau})$$
$$\mu_{ex} = \texttt{manchot} \wedge \neg\texttt{vole}$$

La formule ψ_{ex} *signifie que si Max est un oiseau, alors il vole, et s'il est un manchot, alors il est un oiseau. La formule* μ_{ex} *signifie que Max est un manchot qui ne vole pas.*

1.1.2 Sémantique

Les différents formalismes logiques utilisés dans cette thèse ont des sémantiques définies à partir de la notion d'*interprétation* qui formalise ce qui est représenté par les connaissances. Les interprétations sont parfois appelées *mondes*, avec l'idée que le but des connaissances est de décrire un monde pour le distinguer des autres mondes possibles. On note Ω l'ensemble des interprétations possibles des connaissances exprimées dans $\mathcal{L}$.

Les connaissances peuvent être plus ou moins précises, et donc plusieurs interprétations peuvent en être faites, c'est-à-dire que plusieurs mondes peuvent correspondre à la description faite par ces connaissances. La relation $\models$ établit la correspondance entre interprétations et formules de $\mathcal{L}$: pour une interprétation $I \in \Omega$ et une formule $\varphi \in \mathcal{L}$, $I \models \varphi$ signifie que φ est *vérifiée* dans I, on dit aussi que I est un *modèle* de φ. Si I ne vérifie pas φ, on note $I \not\models \varphi$.

$\models$ est étendue à des ensembles de formules et à des ensembles d'interprétations :

$$\text{pour } I \in \Omega \text{ et } \Psi \text{ une BC}, \quad I \models \Psi \text{ si, pour tout } \varphi \in \Psi, \ I \models \varphi$$
$$\text{pour } A \in 2^{\Omega} \text{ et } \Psi \text{ une BC}, \quad A \models \Psi \text{ si, pour tout } I \in A, \ I \models \Psi$$

Où pour tout ensemble $\mathcal{U}$, on note $2^{\mathcal{U}}$ l'ensemble des parties de $\mathcal{U}$. On note Mod la fonction qui donne l'ensemble des modèles d'une BC :

$$\text{pour } \Psi \text{ une BC}, \quad \mathrm{Mod}\,(\Psi) = \{I \in \Omega \mid I \models \Psi\}$$

Pour simplifier, pour toute formule $\psi \in \mathcal{L}$, on notera $\mathrm{Mod}(\psi)$ au lieu de $\mathrm{Mod}(\{\psi\})$. On peut remarquer que pour Ψ et M deux BC :

$$\mathrm{Mod}(\Psi \cup \mathrm{M}) = \mathrm{Mod}(\Psi) \cap \mathrm{Mod}(\mathrm{M}) \tag{1.1}$$

c'est-à-dire que les modèles de $\Psi \cup \mathrm{M}$ sont les interprétations qui sont à la fois modèle de Ψ et de M.

Des connaissances peuvent être déduites d'autres connaissances. En reprenant l'exemple des

oiseaux, à partir de connaissances Ψ qui affirment que les manchots sont des oiseaux et que tous les oiseaux peuvent voler, on peut déduire les connaissances M qui disent que tous les manchots peuvent voler. M est vérifiée par tout modèle de Ψ. On dit alors que Ψ *entraîne* M, ce qui est noté $\Psi \models \mathrm{M}$:

$$\Psi \models \mathrm{M} \quad \text{si} \quad \mathrm{Mod}(\Psi) \subseteq \mathrm{Mod}(\mathrm{M})$$

C'est-à-dire, si pour tout $I \in \Omega$ tel que $I \models \Psi$, $I \models \mathrm{M}$. Si Ψ implique M et M implique Ψ, on dit que Ψ et M sont *équivalentes*, ce que l'on note $\Psi \equiv \mathrm{M}$, ce qui est équivalent à $\mathrm{Mod}(\Psi) = \mathrm{Mod}(\mathrm{M})$. On note Cn la fonction qui donne l'ensemble des formules impliquées par une BC :

$$\text{pour } \Psi \text{ une BC}, \quad \mathrm{Cn}(\Psi) = \{\varphi \in \mathcal{L} \mid \Psi \models \varphi\}$$

C'est-à-dire que les conséquences de Ψ sont les connaissances qui sont vérifiées dans les modèles de Ψ. Lorsque Ψ est vrai, ses conséquences le sont aussi.

Remarque 1.1.1. *On peut montrer que* Cn *est une opération de fermeture, c'est-à-dire que pour tous ensembles de formules Ψ et M :*

(idempotence) $\mathrm{Cn}(\mathrm{Cn}(\Psi)) = \mathrm{Cn}(\Psi)$.

(extensivité) $\Psi \subseteq \mathrm{Cn}(\Psi)$.

(croissance) *si* $\Psi \subseteq M$ *alors* $\mathrm{Cn}(\Psi) \subseteq \mathrm{Cn}(M)$.

Ces propriétés correspondent aux postulats de Tarsky pour caractériser une logique classique. En particulier, tel qu'elle est définie ici, Cn *est monotone (propriété de croissance), on ne considérera dans cette thèse que des logiques de ce type.*

Définition 1.1.1 (théorie)**.** *Une théorie est un ensemble (non nécessairement fini) Ψ de formules clos pour la déduction, c'est-à-dire tel que $\Psi = \mathrm{Cn}(\Psi)$.*

Il se peut qu'une BC n'ait pas de modèle, cela reflète une contradiction dans les connaissances exprimées. C'est le cas dans l'exemple entre les connaissances affirmant que tous les manchots peuvent voler et qu'il existe un manchot, Max, qui ne peut pas voler. Cette BC est alors dite incohérente :

Définition 1.1.2 (connaissances incohérentes)**.** *Une BC Ψ est dite* incohérente *si* $\mathrm{Mod}(\Psi) = \emptyset$*, c'est-à-dire qu'elle n'a pas de modèle.*

Une BC Ψ est dite incohérente avec une autre BC M lorsque $\Psi \cup M$ est incohérente. Une formule φ est dite incohérente si la BC $\{\varphi\}$ est incohérente.

De telles connaissances sont à éviter car inutiles, on peut montrer en effet que tout (et son contraire) est conséquence d'une BC incohérente.

Logique propositionnelle - sémantique

Une interprétation de la logique propositionnelle sur les variables $\{p_1, \ldots, p_n\}$ est une fonction I qui associe à chaque variables p_i une valeur ${p_i}^I \in \mathbb{B} = \{\texttt{Vrai}, \texttt{Faux}\}$, l'ensemble des booléens.

On assimile une interprétation I au n-uplet $({p_1}^I, \ldots, {p_n}^I) \in \mathbb{B}^n$. $\models$ est défini récursivement sur $\mathcal{L}$, pour $I \in \mathcal{L}$, $p \in \{p_1, \ldots, p_n\}$ et $\psi, \mu \in \mathcal{L}$:

$$
\begin{aligned}
&I \models p \quad \text{si } p^I = \texttt{Vrai} \\
&I \models \neg\psi \quad \text{si } I \not\models \psi \\
&I \models \psi \wedge \mu \quad \text{si } I \models \psi \text{ et } I \models \mu
\end{aligned}
$$

La définition de $\vee$ et $\Rightarrow$ entraîne que :

$$
\begin{aligned}
&I \models \psi \vee \mu \quad \text{ssi } I \models \psi \text{ ou } I \models \mu \\
&I \models \psi \Rightarrow \mu \quad \text{ssi } I \models \psi \text{ implique que } I \models \mu
\end{aligned}
$$

Mod vérifie les propriétés suivantes :

$$
\begin{aligned}
\mathrm{Mod}(\psi \wedge \mu) &= \mathrm{Mod}(\psi) \cap \mathrm{Mod}(\mu) \\
\mathrm{Mod}(\neg\psi) &= \Omega \setminus \mathrm{Mod}(\psi) \\
\mathrm{Mod}(\psi \vee \mu) &= \mathrm{Mod}(\psi) \cup \mathrm{Mod}(\mu) = \mathrm{Mod}(\neg(\neg\psi \wedge \neg\mu))
\end{aligned}
$$

Exemple 1.1.2 (Exemple des oiseaux - suite)**.** *Si on numérote les variables propositionnelles dans l'ordre suivant $p_1 =$ `vole`, $p_2 =$ `oiseau` et $p_3 =$ `manchot` :*

$$
\begin{aligned}
\mathrm{Mod}(\psi_{ex}) &= \{(\mathit{Faux}, \mathit{Faux}, \mathit{Faux}), (\mathit{Vrai}, \mathit{Faux}, \mathit{Faux}), (\mathit{Vrai}, \mathit{Vrai}, \mathit{Faux}), (\mathit{Vrai}, \mathit{Vrai}, \mathit{Vrai})\} \\
\mathrm{Mod}(\mu_{ex}) &= \{(\mathit{Faux}, \mathit{Faux}, \mathit{Vrai}), (\mathit{Faux}, \mathit{Vrai}, \mathit{Vrai})\} \\
\mathrm{Mod}(\psi_{ex} \wedge \mu_{ex}) &= \emptyset
\end{aligned}
$$

$\psi_{ex} \wedge \mu_{ex}$ est donc incohérente.

1.2 Pré-ordres et « distances »

1.2.1 Pré-ordres

Définition 1.2.1 (pré-ordre)**.** *Une relation binaire $\leqslant$ est un* pré-ordre *sur un espace $\mathcal{U}$ si elle vérifie les propriétés suivantes :*

(réflexivité) Pour tout $x \in \mathcal{U}$, $x \leqslant x$.

(transitivité) Pour tous $x, y, z \in \mathcal{U}$, si $x \leqslant y$ et $y \leqslant z$, alors $x \leqslant z$.

Un pré-ordre $\leqslant$ est dit total *si tous les éléments de $\mathcal{U}$ sont comparables par $\leqslant$, c'est-à-dire que pour tout $x, y \in \mathcal{U}$, soit $x \leqslant y$, soit $y \leqslant x$.*

Étant donné un pré-ordre $\leqslant$ sur un ensemble $\mathcal{U}$, pour $x, y \in \mathcal{U}$ on note $x < y$ lorsque $x \leqslant y$ est vrai et $y \leqslant x$ est faux. Si $\leqslant$ est total, alors $x < y$ ssi $y \leqslant x$ est faux.

Définition 1.2.2 (Minima). *Étant donné un pré-ordre $\leqslant$ sur un ensemble $\mathcal{U}$ et un sous-ensemble A de $\mathcal{U}$, on définit les minima de $\leqslant$ sur A par :*

$$\begin{aligned}\underset{\leqslant}{\mathrm{Min}}\, A &= \{x \in A \mid \textit{pour tout } y \in A,\ \textit{si } y \leqslant x,\ \textit{alors}\ \ x \leqslant y\}\\ &= \{x \in A \mid \textit{il n'existe pas de } y \in A,\ \textit{tel que } y < x\}\end{aligned}$$

Remarque 1.2.1. *Lorsque $\leqslant$ est un pré-ordre total sur $\mathcal{U}$, pour tout $x, y \in \mathcal{U}$, soit $x \leqslant y$, soit $y \leqslant x$, donc : $\underset{\leqslant}{\mathrm{Min}}\, A = \{x \in A \mid \textit{pour tout } y \in A,\ x \leqslant y\}$.*

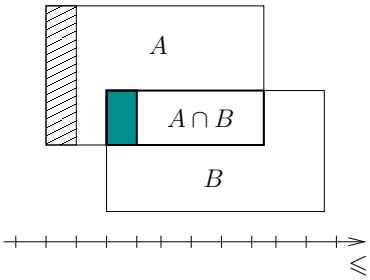
Les propositions 1.2.1 et 1.2.2 donnent une caractérisation de $\underset{\leqslant}{\mathrm{Min}}\,(\cdot)$ lorsque $\leqslant$ est un pré-ordre total. Elles sont inspirées par l'équivalence établie par [Katsuno et Mendelzon, 1991b] entre la caractérisation du changement minimal par les postulats postulats AGM (voir section 2.2) et la caractérisation par des pré-ordres sur l'ensemble des interprétations (voir section 2.2.2) :

Proposition 1.2.1. *Pour $A, B \subseteq \mathcal{U}$ et un pré-ordre total $\leqslant$ sur $\mathcal{U}$:*

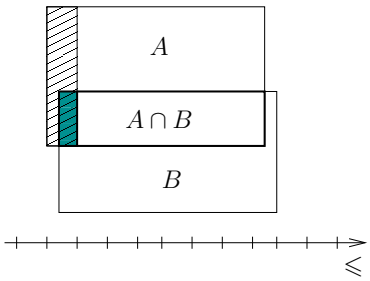
(1) $\left(\underset{\leqslant}{\mathrm{Min}}\, A\right) \cap B \subseteq \underset{\leqslant}{\mathrm{Min}}\,(A \cap B)$

(2) si $\left(\underset{\leqslant}{\mathrm{Min}}\, A\right) \cap B \neq \emptyset$, $\underset{\leqslant}{\mathrm{Min}}\,(A \cap B) \subseteq \left(\underset{\leqslant}{\mathrm{Min}}\, A\right) \cap B$

Donc soit $\left(\underset{\leqslant}{\mathrm{Min}}\, A\right) \cap B = \emptyset$,



soit $\underset{\leqslant}{\mathrm{Min}}\,(A \cap B) = \left(\underset{\leqslant}{\mathrm{Min}}\, A\right) \cap B$.



Preuve de la proposition 1.2.1. Pour le point (1), si $x \in \left(\underset{\leqslant}{\mathrm{Min}}\, A\right) \cap B$, alors $x \in A \cap B$ et $x \in \underset{\leqslant}{\mathrm{Min}}\, A$. Donc pour tout $y \in A \cap B \subseteq A$, si $y \leqslant x$ alors $x \leqslant y$. Donc $x \in \underset{\leqslant}{\mathrm{Min}}\,(A \cap B)$, et (1) est vérifié.

Pour le point (2), supposons que $\left(\underset{\leqslant}{\mathrm{Min}}\, A\right) \cap B \neq \emptyset$ et soit x_0 un élément de cet ensemble. Puisque $x_0 \in \underset{\leqslant}{\mathrm{Min}}\, A$ et que $\leqslant$ est total, pour tout $y \in A$, $x_0 \leqslant y$. Soit $x \in \underset{\leqslant}{\mathrm{Min}}\,(A \cap B)$, montrons que $x \in \left(\underset{\leqslant}{\mathrm{Min}}\, A\right) \cap B$. On sait déjà que $x \in B$, il reste à montrer que $x \in \underset{\leqslant}{\mathrm{Min}}\, A$. Puisque

$x_0 \in A \cap B$ et que $\leqslant$ est total, $x \leqslant x_0$, et donc, par transitivité de $\leqslant$, $x \leqslant y$ pour $y \in A$, ce qui implique que $x \in \underset{\leqslant}{\mathrm{Min}}\, A$. □

Proposition 1.2.2. *Soit $\sigma : 2^{\mathcal{U}} \to 2^{\mathcal{U}}$ vérifiant les propriétés suivantes :*

(i) $\sigma(A) \subseteq A$ *;*

(ii) si $A \neq \emptyset$*, alors* $\sigma(A) \neq \emptyset$ *;*

(iii) $\sigma(A) \cap B \subseteq \sigma(A \cap B)$ *;*

(iv) si $\sigma(A) \cap B \neq \emptyset$*, alors* $\sigma(A \cap B) \subseteq \sigma(A) \cap B$*.*

Dans ces conditions, la relation binaire $\leqslant$ sur $\mathcal{U}$ telle que pour tout $x, y \in \mathcal{U}$, $x \leqslant y$ si $x \in \sigma(\{x, y\})$ est un pré-ordre total sur $\mathcal{U}$ et pour tout $A \subseteq \mathcal{U}$, $\underset{\leqslant}{\mathrm{Min}}\, A = \sigma(A)$.

Preuve de la proposition 1.2.2. Démonstration adaptée de la preuve du théorème 3.3 de [Katsuno et Mendelzon, 1991b].

- **$\leqslant$ est réflexive :** $\{x\} \neq \emptyset$ donc d'après *(ii)*, $\sigma(\{x, x\}) \neq \emptyset$ et $x \in \sigma(\{x, x\})$, d'où $x \leqslant x$.
- **$\leqslant$ est transitive :** Soit $x, y, z \in \mathcal{U}$ distincts deux à deux tels que $x \leqslant y$ et $y \leqslant z$. $\{x, y, z\} \neq \emptyset$ donc d'après *(ii)*, $\sigma(\{x, y, z\}) \neq \emptyset$, or d'après *(i)*, $\sigma(\{x, y, z\}) \subseteq \{x, y, z\}$, donc $\sigma(\{x, y, z\}) \cap \{x, y, z\} \neq \emptyset$.
  - Supposons que $\sigma(\{x, y, z\}) \cap \{x, y\} = \emptyset$, on a alors $\sigma(\{x, y, z\}) = \{z\}$ et $\sigma(\{x, y, z\}) \cap \{y, z\} \neq \emptyset$, donc d'après *(iii)* et *(iv)*, $\sigma(\{x, y, z\} \cap \{y, z\}) = \sigma(\{x, y, z\}) \cap \{y, z\} = \{z\}$, $y \notin \sigma(\{y, z\}) = \sigma(\{x, y, z\} \cap \{y, z\})$, ce qui est contradictoire avec $y \leqslant z$.
  - Ainsi, $\sigma(\{x, y, z\}) \cap \{x, y\} \neq \emptyset$, et d'après *(iv)*, $\sigma(\{x, y\}) = \sigma(\{x, y, z\} \cap \{x, y\}) = \sigma(\{x, y, z\}) \cap \{x, y\}$, donc $\sigma(\{x, y\}) \subseteq \sigma(\{x, y, z\})$. Vu que $x \leqslant y$, $x \in \sigma(\{x, y\}) \subseteq \sigma(\{x, y, z\})$, ce qui implique encore que $\sigma(\{x, y, z\}) \cap \{x, z\} \neq \emptyset$, donc d'après *(iv)*, $x \in \sigma(\{x, y, z\}) \cap \{x, z\} = \sigma(\{x, y, z\} \cap \{x, z\}) = \sigma(\{x, z\})$ et $x \leqslant z$.
- **$\leqslant$ est donc un pré-ordre.**
- **$\leqslant$ est total :** Pour $x, y \in \mathcal{U}$, d'après *(i)* et *(ii)* $\sigma(\{x, y\}) \cap \{x, y\} \neq \emptyset$, donc soit $x \in \sigma(\{x, y\})$ et $x \leqslant y$, soit $y \in \sigma(\{x, y\})$ et $y \leqslant x$.
- **Pour $A \subseteq \mathcal{U}$, $\sigma(A) \subseteq \underset{\leqslant}{\mathrm{Min}}\, A$:** Si $\sigma(A) = \emptyset$, l'inclusion est vérifiée. Sinon, pour $x \in \sigma(A)$ et $y \in A$, $x \in \sigma(A) \cap \{x, y\}$ qui est donc non vide, donc d'après *(ii)* $x \in \sigma(A) \cap \{x, y\} = \sigma(A \cap \{x, y\}) = \sigma(\{x, y\})$ et $x \leqslant y$. Pour tout $y \in A$, $x \leqslant y$ et donc $x \in \underset{\leqslant}{\mathrm{Min}}\, A$.
- **Pour $A \subseteq \mathcal{U}$, $\underset{\leqslant}{\mathrm{Min}}\, A \subseteq \sigma(A)$:** Si $A = \emptyset$, par définition de $\underset{\leqslant}{\mathrm{Min}}$, $\underset{\leqslant}{\mathrm{Min}}\, A = \emptyset \subseteq \sigma(A)$. Sinon d'après *(ii)*, $\sigma(A) \neq \emptyset$. Soient $x \in \underset{\leqslant}{\mathrm{Min}}\, A$ et $y \in \sigma(A)$, puisque $\leqslant$ est total, soit $x \leqslant y$, soit $y \leqslant x$, ce qui entraîne aussi $x \leqslant y$ par définition de $\underset{\leqslant}{\mathrm{Min}}\, A$, donc dans tous les cas $x \leqslant y$. Comme $y \in \sigma(A)$, $\sigma(A) \cap \{x, y\} \neq \emptyset$, donc d'après *(iv)*, $\sigma(A) \cap \{x, y\} = \sigma(A \cap \{x, y\}) = \sigma(\{x, y\})$, or comme $x \leqslant y$, $x \in \sigma(\{x, y\})$ et $x \in \sigma(A)$.

□

Définition 1.2.3 (Minima d'une fonction réelle)**.** *Étant donné une fonction $f : \mathcal{U} \to \mathbb{R}$ à valeurs réelles, on définit le pré-ordre total $\leqslant_f$ sur $\mathcal{U}$ tel que pour $x, y \in \mathcal{U}$, $x \leqslant_f y$ ssi $f(x) \leqslant f(y)$.*

On note alors $\operatorname*{Min}_f A = \operatorname*{Min}_{\leqslant_f} A$ *pour tout* $A \subseteq \mathcal{U}$.

1.2.2 « Distance »

Par abus de langage, on utilise le terme « distance » sur un ensemble $\mathcal{U}$ pour désigner une fonction d de $\mathcal{U} \times \mathcal{U}$ dans $[0, +\infty]$ qui vérifie la séparation : pour tout $x, y \in \mathcal{U}$, $d(x, y) = 0$ ssi $x = y$.

On parlera aussi de « distance » entre un sous-ensemble A de $\mathcal{U}$ et un élément $x \in \mathcal{U}$, et de « distance » entre deux sous-ensembles A et B de $\mathcal{U}$ pour désigner les valeurs $d(A, x)$ et $d(A, B)$ telles que :

$$d(A, y) = \inf_{x \in A} d(x, y)$$
$$d(A, B) = \inf_{y \in B} d(A, y) = \inf_{x \in A,\, y \in B} d(x, y)$$

Par convention $\inf \emptyset = +\infty$, on a alors $d(\emptyset, x) = +\infty$ pour tout $x \in \mathcal{U}$ et $d(A, B) = +\infty$ lorsque A ou B sont vides.

Minimisation d'une « distance »

Proposition 1.2.3. *Étant donné une « distance » d sur un ensemble $\mathcal{U}$ et deux sous ensembles A et B de $\mathcal{U}$ avec $A = A_1 \cup \ldots \cup A_m$, et $B = B_1 \cup \ldots \cup B_n$:*

$$\operatorname*{Min}_{d(A,\cdot)} B = \bigcup_{\substack{1 \leqslant i \leqslant m,\, 1 \leqslant j \leqslant n \\ d(A_i, B_j) = d(A, B)}} \operatorname*{Min}_{d(A_i,\cdot)} B_j$$

Cette propriété est conséquence des propriétés 1.2.4 et 1.2.5.

Proposition 1.2.4. *Étant donné une « distance » d sur un ensemble $\mathcal{U}$ et deux sous ensembles A et B de $\mathcal{U}$ avec $A = A_1 \cup \ldots \cup A_m$ alors :*

$$\operatorname*{Min}_{d(A,\cdot)} B = \bigcup_{\substack{1 \leqslant i \leqslant m \\ d(A_i, B) = d(A, B)}} \operatorname*{Min}_{d(A_i,\cdot)} B$$

Preuve de la proposition 1.2.4. On prouve ici le cas $n = 2$, le cas général est obtenu par récursion.

$$\operatorname*{Min}_{d(A_1 \cup A_2,\cdot)} B = \{x \in B \mid d(A_1 \cup A_2, x) = d(A_1 \cup A_2, B)\}$$

or $d(A_1 \cup A_2, x) = \min(d(A_1, x), d(A_2, x))$ et $d(A_1 \cup A_2, B) = \min(d(A_1, B), d(A_2, B))$ donc :

Si $d(A_1, B) < d(A_2, B)$.

$$\begin{aligned}\operatorname*{Min}_{d(A_1\cup A_2,\cdot)} B &= \{x \in B \mid d(A_1 \cup A_2, x) = d(A_1, B)\} \\ &= \{x \in B \mid d(A_1, x) = d(A_1, B) \text{ ou } d(A_2, x) = d(A_1, B)\} \\ &= \{x \in B \mid d(A_1, x) = d(A_1, B)\} \cup \{x \in B \mid d(A_2, x) = d(A_1, B)\}\end{aligned}$$

or pour tout $x \in B$, $d(A_2, x) \geqslant d(A_2, B) > d(A_1, B)$, donc :

$$\operatorname*{Min}_{d(A_1\cup A_2,\cdot)} B = \operatorname*{Min}_{d(A_1,\cdot)} B$$

Si $d(A_2, B) < d(A_1, B)$. Par le même raisonnement que pour le cas précédant on obtient :

$$\operatorname*{Min}_{d(A_1\cup A_2,\cdot)} B = \operatorname*{Min}_{d(A_2,\cdot)} B$$

Si $d(A_1, B) = d(A_2, B)$.

$$\begin{aligned}\operatorname*{Min}_{d(A_1\cup A_2,\cdot)} B &= \{x \in B \mid d(A_1, x) = d(A_1, B) \text{ ou } d(A_2, x) = d(A_2, B)\} \\ &= \{x \in B \mid d(A_1, x) = d(A_1, B)\} \cup \{x \in B \mid d(A_2, x) = d(A_2, B)\} \\ &= \operatorname*{Min}_{d(A_1,\cdot)} B \cup \operatorname*{Min}_{d(A_2,\cdot)} B\end{aligned}$$

□

Proposition 1.2.5. *Étant donné une « distance » d sur un espace $\mathcal{U}$ et deux sous ensembles A et B de $\mathcal{U}$ avec $B = B_1 \cup \ldots \cup B_n$:*

$$\operatorname*{Min}_{d(A,\cdot)} B = \bigcup_{\substack{1\leqslant j\leqslant n \\ d(A,B_j)=d(A,B)}} \operatorname*{Min}_{d(A,\cdot)} B_j$$

Preuve de la proposition 1.2.5.

$$\operatorname*{Min}_{d(A,\cdot)} B = \{x \in B \mid d(A, x) = d(A, B)\} = \bigcup_{1\leqslant j\leqslant n} \{x \in B_j \mid d(A, x) = d(A, B)\}$$

Or pour tout j, si $d(A, B_j) \neq d(A, B)$, alors $d(A, B_j) > d(A, B)$ et pour tout $x \in B_j$, $d(A, x) \geqslant d(A, B) > d(A, B)$, donc $\{x \in B_j \mid d(A, x) = d(A, B)\} = \emptyset$. On peut donc ne conserver que les indices j tels que $d(A, B_j) = d(A, B)$:

$$\operatorname*{Min}_{d(A,\cdot)} B = \bigcup_{\substack{1\leqslant j\leqslant n \\ d(A,B_j)=d(A,B)}} \{x \in B_j \mid d(A, x) = d(A, B_j)\} = \bigcup_{\substack{1\leqslant j\leqslant n \\ d(A,B_j)=d(A,B)}} \operatorname*{Min}_{d(A,\cdot)} B_j$$

□

Adhérence

On considère un ensemble $\mathcal{U}$ muni d'une « distance » d.

Pour tout ensemble $A \subseteq \mathcal{U}$, on définit *l'adhérence* de A comme étant l'ensemble $\overline{A}$ tel que :

$$\overline{A} = \{u \in \mathcal{U} \mid d(A, u) = 0\}$$

Remarque 1.2.2. *Cette définition est justifiée par le fait que l'ensemble des boules ouvertes* $\mathcal{B}(x, \varepsilon) = \{y \in \mathcal{U} \mid d(x, y) < \varepsilon\}$, *où* $x \in \mathcal{U}$ *et* $\varepsilon > 0$, *engendre une topologie pour laquelle l'adhérence d'un ensemble* $A \subseteq \mathcal{U}$ *est le fermé contenant* A *minimal pour l'inclusion.*

Pour $x \in \mathcal{U}$ *et* $\varepsilon > 0$, $\mathcal{B}(x, \varepsilon)$ *est la boule ouverte de centre* x *et de rayon* ε.

Chapitre 2

Révision et changement minimal des connaissances

Les travaux de cette thèse s'appuient sur la théorie de la révision pour formaliser des raisonnements pour le RÀPC (chapitre 3). Dans ce chapitre nous nous intéressons à la révision des connaissances dans cette perspective. Les deux premières sections sont consacrées à un état de l'art de la révision, avec, en section 2.1, un tour d'horizon des problématiques de changement minimal de connaissances, dont fait partie la révision, puis, en section 2.2, une présentation du cadre AGM de la révision en logique propositionnelle, il s'agit d'une caractérisation largement admise des opérateurs de révision. Nous proposons une reformulation de la caractérisation de la révision dans un cadre plus large en section 2.3. Puis nous définissons des opérateurs de révision sur des formalismes utiles au RÀPC : une classe d'opérateurs de révision dans les formalismes attributs-valeurs simples en section 2.4 et une version préliminaire d'opérateur dans une logique de descriptions expressive en section 2.5.

2.1 La révision et autres problèmes de changement minimal de connaissances

La révision de connaissances et plusieurs autres problèmes rencontrés dans la littérature peuvent être regroupés sous le terme changement minimal de connaissances. Il s'agit d'apporter des modifications à des connaissances, suivant un critère de minimalité du changement, afin de satisfaire des contraintes.

2.1.1 Révision de connaissances

La révision de connaissances K par d'autres connaissances L est une opération qui vise à assimiler L à K en obtenant un résultat cohérent. Pour cela, il peut être nécessaire de remettre en cause en partie K, afin de le rendre cohérent avec L, ce qui explique le nom de « révision ».

Typiquement, cette opération a lieu lorsque les connaissances K représentent imparfaitement le domaine et que des connaissances L, considérées comme plus exactes, sont acquises.

> Dans l'exemple donné en introduction, les connaissances K « Tous les oiseaux peuvent voler, et les manchots, les rapaces et les autruches sont des oiseaux. » sont incohérentes avec les connaissances L « Max est un manchot qui ne peut pas voler. » La révision de K par L nécessite une remise en cause de K.

Abandonner K dans son ensemble pour ne conserver que L permet d'assurer la cohérence, si on suppose que L est elle-même cohérente. Mais ce n'est pas forcément le résultat le plus intéressant, K peut contenir des connaissances compatibles avec L et qu'il serait préférable de ne pas perdre. On ne veut pas remettre en cause plus de connaissances de K que nécessaire, ce qui justifie la notion de minimalité des changements apportés à K. Plusieurs postulats de la théorie AGM (voir section 2.2) caractérisent cette minimalité du changement.

Comme mentionné en introduction, il y a plusieurs révisions possibles de K par L, chacune correspondant à une modification différente de K. Suivant l'importance que l'on donne aux différentes connaissances, les unes par rapport aux autres, certains choix sont plus judicieux que d'autres.

> Par exemple, la connaissance zoologique qui classe les manchots dans la famille des oiseaux peut être jugée plus fiable que la connaissance empirique qui dit que tous les oiseaux peuvent voler. Il sera alors judicieux d'abandonner la seconde connaissance plutôt que la première car elle demande moins de remise en question.

La révision est donc guidée par la minimisation du changement qui prend en compte l'importance donnée aux connaissances manipulées. On peut être aussi plus ou moins fin pour les modifications apportées à K :

> On peut soit renoncer entièrement au fait que les oiseaux peuvent voler, soit ne renoncer qu'au fait que les manchots peuvent voler en conservant cette propriété pour les autres oiseaux, ou même être encore plus précis et considérer Max comme une exception. La réalité correspond au deuxième niveau, ce qui consiste à favoriser la modification des connaissances concernant les manchots seulement par rapport à la modification des connaissances sur l'ensemble des oiseaux, sans non plus distinguer Max des autres manchots et ne modifier que les connaissances le concernant.

La finesse de la révision est déterminée par la finesse du classement des modifications. Plusieurs représentions de classement de changements sont présentées en section 2.2 avec leur lien avec la théorie AGM.

2.1.2 Fusion des connaissances

La fusion de connaissances est aussi une opération d'agrégation de connaissances potentiellement incohérentes entre elles. Contrairement à la révision, il ne s'agit pas de corriger des connaissances, mais de faire la synthèse de connaissances provenant de plusieurs sources. Il n'y a

donc pas *a priori* de primauté donnée à certaines sources par rapport à d'autres, le but est plutôt d'obtenir un compromis nécessitant un minimum de concessions de l'ensemble des sources.

> Par exemple, trois amateurs de cyclisme discutent des résultats du contre la montre qui a eu lieu le matin. Malheureusement aucun n'a pu le voir en entier, mais le premier pense que le concurrent a a fait mieux que b ($a < b$), le deuxième pense que b a fait mieux que c ($b < c$), et le troisième pense que c a fait mieux que a ($c < a$). Mises ensemble, ces connaissances sont incohérentes et des modifications sont nécessaires pour obtenir une synthèse cohérente.

Comme pour la révision, le résultat dépend de l'importance accordée aux différentes connaissances.

> Dans cet exemple, on considérera deux niveaux de remise en cause : affaiblir l'inégalité $x < y$ en $x \leqslant y$ (x a fait au moins aussi bien que y) pour un coût de 1 ou oublier complètement la connaissance pour un coût de 2.

Il y a aussi différentes stratégies pour obtenir un compromis entre les sources.

> Par exemple, si on minimise le coût total du changement à apporter aux connaissances des différentes sources, on conclut que l'un des trois amis s'est complètement trompé et on a l'un des classements possibles $a < b < c$ (le troisième s'est trompé), $b < c < a$ (le premier s'est trompé) ou $c < a < b$ (le deuxième s'est trompé) pour un coût de changement de 2. Les autres solutions ont au moins un coût de 3. Cette solution fait supporter tous les changement par une seule source, on peut forcer à répartir les changement en minimisant le coût maximal de changement supporté par l'une des sources. Ce qui donne le résultat $a = b = c$ avec un coût de 1 par source.

La fusion contrainte de connaissances [Konieczny, 1999] et [Konieczny *et al.*, 2004] tient compte de connaissances supplémentaires, les contraintes avec lesquelles le résultat de la fusion doit être cohérent.

> Reprenons l'exemple précédant, alors que les trois amis discutent au PMU du classement, une interview du cycliste b passe à la télé, dans laquelle il se dit heureux d'avoir fini second sachant que le troisième le talonnait. Puisque c'est à la télé, c'est que c'est vrai, b est donc deuxième et il n'a pas d'*ex aequo*, les classements $a = b = c$ et $c < a < b$ ne sont donc pas possibles. *A priori*, le résultat de la fusion des connaissances apportées par les trois amis sous contraintes des connaissances apportées par la télé est soit $a < b < c$, soit $b < c < a$ (un autre concurrent non nommé est vainqueur).

La révision des connaissances correspond au cas particulier de la fusion de connaissances provenant d'une seule source, les connaissances initiales, contrainte par les nouvelles connaissances. La fusion contrainte de connaissances généralise donc la révision. [Konieczny, 1999] proposent des postulats caractérisant un opérateur de fusion contrainte. La fusion contrainte des connaissances est présentée plus en détails en section 2.7.

Cette thèse traite principalement d'un approche de l'adaptation en RÀPC définie à partir de la révision (voir chapitre 3). En nous appuyant sur la fusion contrainte de connaissances plutôt que la révision, nous proposons en section 3.4, une combinaison de cas, où ce n'est plus un seul mais plusieurs cas sources qui sont exploités pour résoudre un problème.

2.1.3 Contraction

La contraction est en quelque sorte l'opération inverse de la révision, la contraction de K par L consiste à modifier au minimum les connaissances K afin qu'elles n'entraînent plus L.

> Par exemple, les cochons de *La Ferme des Animaux* [Orwell, 1945] sont des experts en contraction : c'est un outil de propagande idéale pour empêcher les animaux de parvenir à des conclusions gênantes. Un des commandements de la ferme établit que « Nul animal ne dormira dans un lit », or un beau jour les cochons décident de s'installer dans la maison et de dormir dans les lits. À partir de ces connaissances K, les animaux pourraient déduire L : « Les cochons enfreignent les commandements de la ferme. » Il faut contracter K par L !

Comme pour la révision, différentes contractions sont possibles.

> Par exemple, oublier le fait que les cochons dorment dans des lits, ou effacer le commandement interdisant de dormir dans un lit, ou plus fin en l'affaiblissant en « Nul animal ne dormira dans un lit avec des draps. » Ce qui permet aux cochons d'assurer aux autres animaux qu'ils respectent les commandements puisqu'ils dorment sans draps.

La révision et la contraction soulèvent toutes deux le problème de la modification de connaissances afin qu'elles n'aient plus certaines conséquences. Dans le cas de la révision de K par L, il faut modifier K afin qu'elle n'ait pas de conséquences incohérentes avec L. Cela équivaut à contracter K par l'ensemble des connaissances incohérentes avec L, si celles-ci peuvent être exprimées dans le formalisme utilisé. Dans le cadre AGM de la révision, ces deux opérations sont liées par les identités de Levi et de Harper [Gärdenfors, 1992].

2.1.4 Mise à jour de connaissances

Comme la révision, la mise à jour est une opération qui consiste à modifier des connaissances pour les rendre cohérentes avec d'autres connaissances. Mais l'hypothèse faite sur la cause des incohérences potentielles est différente : dans le cas de la révision elles sont dues à l'imperfection des connaissances initiales, alors que dans le cas de la mise à jour, elles sont dues à une évolution dans le temps du domaine. L'hypothèse faite dans le cas de la mise à jour implique aussi une minimalité du changement : si rien n'indique que quelque chose a changé, il n'y a pas de raison de penser que cette chose ait changé. Cependant, [Katsuno et Mendelzon, 1991b] établissent une différence entre les opérations de révision et de mise à jour. Cette différence se traduit par des

critères de minimalité différents, liés à des significations différentes des changements à apporter à K.

2.1.5 Débogage d'ontologies

Le débogage de BC est un problème d'ingénierie des connaissances, il consiste à modifier les formules d'une BC pour qu'elle respecte des propriétés désirées. *A priori*, la cohérence de la BC fait partie de ces propriétés. En effet, nous avons vu en préliminaires qu'une BC incohérente n'a pas de sens d'un point de vue logique.

Plusieurs travaux récents[Kalyanpur, 2006], [Schlobach *et al.*, 2007] portent sur le débogage d'ontologies. Une *ontologie* est une conceptualisation d'un domaine, c'est-à-dire une représentation ensembliste où des concepts représentent des ensembles d'éléments et des relations sont décrites entre éléments de ces ensembles. Les ontologies sont largement utilisées pour représenter des connaissances, notamment au sein du web sémantique [Berners-Lee *et al.*, 2001]. Le formalisme standard de représentation des ontologies du web sémantique, OWL, repose sur des logiques de descriptions qui lui donnent une sémantique précise.

[Flouris *et al.*, 2008] proposent un état de l'art de différents problèmes de changements de connaissances dans les ontologies, ces problèmes sont liés notamment aux aspects collaboratifs et distribués des connaissances sur le web sémantique. Plusieurs approches de révision s'appuient sur des outils de débogage [Halaschek-Wiener *et al.*, 2006], [Haase *et al.*, 2005], [Qi *et al.*, 2008]. En Section 2.5, nous proposons aussi une version préliminaire d'opérateur de révision dans une logique de descriptions s'appuyant sur des outils de débogage d'ontologies.

2.2 Théorie AGM de la révision des connaissances

La révision est une opération, notée $\dot{+}$, entre des connaissances représentées par des bases de connaissances Ψ et M. Cette opération consiste à modifier Ψ afin de lui intégrer M tout en restant cohérent. Le résultat de la révision de Ψ par M est noté $\Psi \dot{+} \mathrm{M}$. Nous avons vu avec l'exemple des oiseaux (Section 2.1.1) qu'il peut y avoir plusieurs manières de réviser Ψ par M, il n'y a donc pas un unique opérateur de révision $\dot{+}$. Plutôt que de définir un opérateur de révision particulier, [Alchourrón *et al.*, 1985] établissent une liste de propriétés attendues d'un opérateur de révision, appelés postulats AGM (initiales des auteurs). Beaucoup de travaux sur la révision sont faits en logique propositionnelle et s'appuient sur une reformulation des postulats AGM proposée dans [Katsuno et Mendelzon, 1991b] que nous présenterons en section 2.2.1. [Katsuno et Mendelzon, 1991b] proposent aussi une représentation d'opérateur de révision sous forme de relations d'ordre sur les interprétations, une assignation fidèle, présentée en section 2.2.2. Nous présentons en section 2.2.3 l'opérateur de révision de Dalal, défini à partir d'une « distance » entre interprétations. En RÀPC les connaissances sont souvent exprimées dans des formalismes plus expressifs.

2.2.1 Postulats de la révision en logique propositionnelle

En logique propositionnelle toute BC est équivalente à une formule, on considère alors pour simplifier $\dot{+}$ comme une opération entre deux formules et dont le résultat est aussi une formule. Les postulats de [Katsuno et Mendelzon, 1991b] sont les suivants :

(R1) $\psi \dot{+} \mu \models \mu$.

(R2) Si $\psi \wedge \mu$ est satisfiable alors $\psi \dot{+} \mu \equiv \psi \wedge \mu$.

(R3) Si μ est satisfiable alors $\psi \dot{+} \mu$ est également satisfiable.

(R4) Si $\psi_1 \equiv \psi_2$ et $\mu_1 \equiv \mu_2$ alors $\psi_1 \dot{+} \mu_1 \equiv \psi_2 \dot{+} \mu_2$.

(R5) $(\psi \dot{+} \mu) \wedge \varphi \models \psi \dot{+} (\mu \wedge \varphi)$.

(R6) Si $(\psi \dot{+} \mu) \wedge \varphi$ est satisfiable alors $\psi \dot{+} (\mu \wedge \varphi) \models (\psi \dot{+} \mu) \wedge \varphi$.

Où ψ, ψ_1, ψ_2, μ, μ_1, μ_2 et φ sont des formules.

(R1) établit que le résultat de $\psi \dot{+} \mu$ intègre μ, c'est-à-dire que μ est une conséquence de $\psi \dot{+} \mu$. (R2) établit que si ψ et μ ne sont pas contradictoires, alors il n'est pas nécessaire de modifier ψ, il suffit d'y ajouter μ. (R3) établit le fait que le résultat de la révision doit, si possible, être cohérent. Si μ est incohérente, la révision par μ ne peut pas donner un résultat cohérent, c'est une conséquence de (R1). Mais si μ est cohérent, on doit obtenir un résultat cohérent (quitte à abandonner ψ en entier). (R4) établit l'indépendance à la syntaxe de la révision. Ce postulat, qui établit que seul le sens importe et non l'expression, n'est pas toujours souhaité, pour des raisons de temps de calcul [Fuhrmann, 1991], [Nebel, 1992], [Dixon et Wobcke, 1993] ou parce que dans certaines applications l'expression est importante [Hansson, 1991] Les postulats (R5) et (R6) sont des critères de minimalité du changement. Intuitivement, ils correspondent respectivement aux propriétés (1) et (2) de la proposition 1.2.1 (page 9) où A représente les connaissances représentées par μ, B les connaissances représentées par φ, et $\leqslant$ compare les modifications par rapport aux connaissances représentée par ψ. Cette correspondance est établie dans la section suivante grâce à la notion d'assignation fidèle.

2.2.2 Assignation Fidèle

[Katsuno et Mendelzon, 1991b] donnent une représentation de la notion de changement minimal propre à chaque opérateur de révision sous la forme d'un pré-ordre, pour chaque formule, de l'ensemble des interprétations. Ce système de pré-ordres est appelé une assignation fidèle [1].

On reprend les notations des préliminaires où Ω est l'ensemble des interprétations, et pour une formule $\psi \in \mathcal{L}$, $\mathrm{Mod}(\psi)$ est l'ensemble des modèles de ψ .

Définition 2.2.1. *[Assignation fidèle [Katsuno et Mendelzon, 1991b]] Une* assignation fidèle *est une application $\leqslant_{.}$ qui à une formule ψ sur L associe un pré-ordre $\leqslant_{\psi}$ sur Ω tel que :*

1. Dans [Konieczny, 1999] le mot « assignement » est utilisé à la place de assignation, mais il n'est pas dans le dictionnaire [Robert, 2007].

$(A1)$ *si* $I, I' \in \mathrm{Mod}(\psi)$ *alors non*$(I <_\psi I')$
$(A2)$ *si* $I \in \mathrm{Mod}(\psi)$ *et* $I' \notin \mathrm{Mod}(\psi)$ *alors* $I <_\psi I'$
$(A3)$ *si* $\psi \equiv \varphi$ *alors* $\leqslant_\psi = \leqslant_\varphi$

Intuitivement, pour une formule ψ, $\leqslant_\psi$ représente l'éloignement par rapport aux modèles de ψ : une interprétation classée avant une autre par $\leqslant_\psi$ est considérée comme plus proche des modèles de ψ. $(A1)$ et $(A2)$ établissent le fait que les minima de $\leqslant_\psi$ sont les modèles de ψ. $(A3)$ établit l'indépendance à la syntaxe. Le résultat de la révision de ψ par μ est déterminé par les modèles de μ les plus proches de ψ :

Théorème 2.2.1 ([Katsuno et Mendelzon, 1991b]). *Un opérateur* $\dot{+}$ *satisfait les postulats* (R*1*) *à* (R*6*) *ssi il existe une assignation fidèle* $\leqslant_\cdot$ *telle que pour toute formule* ψ, $\leqslant_\psi$ *est un pré-ordre total et pour toute formule* μ *:*

$$\mathrm{Mod}(\psi \dot{+} \mu) = \underset{\leqslant_\psi}{\mathrm{Min}}\,(\mathrm{Mod}(\mu)) \tag{2.1}$$

Remarque 2.2.1. *La preuve de ce théorème donné par [Katsuno et Mendelzon, 1991b] s'appuie sur plusieurs propriétés de la logique propositionnelle :*

- $\mathcal{L}$ *est stable par* $\wedge$ *(utilisé dans les postulats* (R*2*), (R*5*) *et* (R*6*)*).*
- *Pour tout pré-ordre* $\leqslant$ *sur* Ω, $\leqslant$ *admet des minima sur tout sous-ensemble non vide* A *de* Ω, *c'est-à-dire que* $\underset{\leqslant}{\mathrm{Min}}\, A \neq \emptyset$ *(nécessaire à la satisfaction de* (R*4*)*). Ceci est une conséquence du fait que* Ω *est fini, et donc que tout* $A \subseteq \Omega$ *l'est aussi.*
- *Pour tout ensemble* $A \subseteq \Omega$, *il existe une formule* $\psi \in \mathcal{L}$ *telle que* $\mathrm{Mod}(\psi) = A$. *Cette propriété permet d'assurer l'existence d'une opération* $\dot{+}$ *satisfaisant* (2.1). *Étant donné que nous n'accordons pas d'importance à la syntaxe (dans l'esprit du postulat* (R*4*)*), le résultat de la révision peut être donné sous la forme d'un ensemble d'interprétations* A *en considérant par exemple que l'on lui associe la formule canonique* $\mathrm{form}(A)$ *:*

$$\mathrm{form}(A) = \bigvee_{I \in A} \mathrm{form}(I)$$

$$\textit{où} \qquad \mathrm{form}(I) = \ell_1 \wedge \ldots \wedge \ell_n \quad \textit{avec} \quad \ell_i = \begin{cases} p_i & \textit{si } {p_i}^I = \texttt{Vrai} \\ \neg p_i & \textit{si } {p_i}^I = \texttt{Faux} \end{cases}$$

Ainsi, (2.1) *permet de construire, modulo la syntaxe, un opérateur de révision* $\dot{+}$ *à partir d'une assignation fidèle* $\leqslant_\cdot$. *Réciproquement, pour tout opérateur de révision* $\dot{+}$, *[Katsuno et Mendelzon, 1991b] définissent une assignation fidèle* $\leqslant_\cdot$ *satisfaisant* (2.1) *en posant pour tout* $\psi \in \mathcal{L}$ *et* I, $J \in \Omega$, $I \leqslant_\psi J$ *ssi* $I \in \mathrm{Mod}(\psi \dot{+} \mathrm{form}(\{I, J\}))$.

En section 2.4 nous proposerons une généralisation de la définition d'assignation fidèle à des formalismes plus généraux.

2.2.3 Opérateur de révision de Dalal et une généralisation

Les assignations fidèles caractérisent l'importance des changements de connaissances en comparant la proximité des interprétations entre elles. Une assignation fidèle permet de définir un opérateur de révision. [Dalal, 1988] propose aussi un opérateur de révision défini à partir du changement mesuré au niveau des interprétations, il mesure le changement entre deux interprétations en fonction du nombre de variables propositionnelles dont la valeur change. [Katsuno et Mendelzon, 1991b] reformulent la définition de cet opérateur de révision en s'appuyant sur une assignation fidèle définie à partir de la distance de Hamming :

Définition 2.2.2 (distance de Hamming)**.** *On assimile les interprétations de la logique propositionnelle sur les variables $\mathcal{V} = \{p_1, \ldots, p_n\}$ à $\Omega = \mathbb{B}^n$ ($\mathbb{B} = \{\texttt{Vrai}, \texttt{Faux}\}$, voir les préliminaires). La distance de Hamming entre deux interprétation I et J est définie comme étant le nombre de variables qui prennent une valeur différente avec I et J :*

$$d_H(I, J) = card\{p \in \mathcal{V} \mid p^I \neq p^J\}$$

En reprenant les notations vues en préliminaires (section 1.2.2), étant donné un ensemble d'interprétations $A \subseteq \Omega$ et une interprétation $I \in \Omega$, $d_{\mathrm{H}}(A, I)$ représente la « distance » de A à I.

- Si $I \in A$, $d_{\mathrm{H}}(A, I) = 0$, car $0 \leqslant d_{\mathrm{H}}(A, I) \leqslant d_{\mathrm{H}}(I, I) \leqslant 0$.
- Si $I \notin A$, $d_{\mathrm{H}}(A, I) \geqslant 1$, car pour tout $J \in A$, $d_{\mathrm{H}}(J, I) \geqslant 1$.

Cette « distance » permet de comparer le changement entre des interprétations et les modèles d'une formule, c'est-à-dire de définir une assignation fidèle :

Proposition 2.2.1 (inspirée de [Katsuno et Mendelzon, 1991b])**.** *La fonction $\leqslant_{\cdot}^{d_H}$ qui à une formule $\psi \in \mathcal{L}$ associe la relation $\leqslant_{\psi}^{d_H}$ telle que pour $I, J \in \Omega$:*

$$I \leqslant_{\psi}^{d_H} J \quad ssi \quad d_H(\mathrm{Mod}(\psi), I) \leqslant d_H(\mathrm{Mod}(\psi), J)$$

est une assignation fidèle.

Démonstration. Pour $\psi \in \mathcal{L}$:

$\leqslant_{\psi}^{d_{\mathrm{H}}}$ **est réflexive :** pour $I \in \Omega$, $d_{\mathrm{H}}(\mathrm{Mod}(\psi), I) \leqslant d_{\mathrm{H}}(\mathrm{Mod}(\psi), I)$, donc $I \leqslant_{\psi}^{d_{\mathrm{H}}} I$.

$\leqslant_{\psi}^{d_{\mathrm{H}}}$ **est transitive :** pour $I_1, I_2, I_3 \in \Omega$ tels que $I_1 \leqslant_{\psi}^{d_{\mathrm{H}}} I_2$ et $I_2 \leqslant_{\psi}^{d_{\mathrm{H}}} I_3$, on a, par définition de $\leqslant_{\psi}^{d_{\mathrm{H}}}$, $d_{\mathrm{H}}(\mathrm{Mod}(\psi), I_1) \leqslant d_{\mathrm{H}}(\mathrm{Mod}(\psi), I_2)$ et $d_{\mathrm{H}}(\mathrm{Mod}(\psi), I_2) \leqslant d_{\mathrm{H}}(\mathrm{Mod}(\psi), I_3)$, donc $d_{\mathrm{H}}(\mathrm{Mod}(\psi), I_1) \leqslant d_{\mathrm{H}}(\mathrm{Mod}(\psi), I_3)$ et $I_1 \leqslant_{\psi}^{d_{\mathrm{H}}} I_3$.

$\leqslant_{\psi}^{d_{\mathrm{H}}}$ est donc un pré-ordre sur Ω.

$\leqslant_{\psi}^{d_{\mathrm{H}}}$ **est total :** pour I, $J \in \Omega$, $d_{\mathrm{H}}(\mathrm{Mod}(\psi), I)$ et $d_{\mathrm{H}}(\mathrm{Mod}(\psi), J)$ sont comparables dans $\mathbb{R}$, donc soit $I \leqslant_{\psi}^{d_{\mathrm{H}}} J$, soit$J \leqslant_{\psi}^{d_{\mathrm{H}}} I$.

$\leqslant_{\bullet}^{d_{\mathrm{H}}}$ **vérifie bien la condition (A3) de la définition 2.2.1 :** elle est définie à partir des $\mathrm{Mod}(\psi)$.

$\leqslant_{\bullet}^{d_{\mathrm{H}}}$ **vérifie bien la condition (A1) de la définition 2.2.1 :** pour $I \in \mathrm{Mod}(\psi)$, $d_{\mathrm{H}}(\mathrm{Mod}(\psi), I) = 0$. Donc pour tout $I, J \in \mathrm{Mod}(\psi)$, $I \leqslant_{\psi}^{d_{\mathrm{H}}} J$ et $J \leqslant_{\psi}^{d_{\mathrm{H}}} I$.

$\leqslant_{\bullet}^{d_{\mathrm{H}}}$ **vérifie bien la condition (A2) de la définition 2.2.1 :** pour $I \in \mathrm{Mod}(\psi)$ et $J \notin \mathrm{Mod}(\psi)$, $d_{\mathrm{H}}(\mathrm{Mod}(\psi), I) = 0$ et $d_{\mathrm{H}}(I', J) \geqslant 1$. Donc $d_{\mathrm{H}}(\mathrm{Mod}(\psi), I) < d_{\mathrm{H}}(I', J)$, ce qui entraîne que $I \leqslant_{\psi}^{d_{\mathrm{H}}} J$ et non $J \leqslant_{\psi}^{d_{\mathrm{H}}} I$, soit $I <_{\psi}^{d_{\mathrm{H}}} J$.

□

L'opérateur de révision de Dalal est l'opérateur de révision obtenu à partir de l'assignation fidèle $\leqslant_{\bullet}^{d_{\mathrm{H}}}$ par la formule 2.1 :

Définition 2.2.3 (Opérateur de révision de Dalal)**.** *Pour $\psi, \mu \in \mathcal{L}$, la révision $\psi \dot{+}_D \mu$ est définie par :*

$$\begin{aligned}\mathrm{Mod}(\psi \dot{+}_D \mu) &= \operatorname*{Min}_{\leqslant_{\psi}^{d_H}} \mathrm{Mod}(\mu) \\ &= \{I \in \mathrm{Mod}(\mu) \mid d_H(\mathrm{Mod}(\psi), I) = d_H(\mathrm{Mod}(\psi), \mathrm{Mod}(\mu))\}\end{aligned}$$

Les modèles de $\psi \dot{+}_{\mathrm{D}} \mu$ sont dons les modèles de μ les plus proches de ceux de ψ pour d_{H} . Comme noté dans la remarque 2.2.1, cette définition d'un opérateur de révision par ses modèles est justifiée par le fait que l'on n'accorde pas d'importance à la syntaxe et que tout ensemble d'interprétations correspond aux modèles d'une formule. Cette définition ne dépend elle-même pas directement de ψ et de μ, mais de leurs modèles $\mathrm{Mod}(\psi)$ et $\mathrm{Mod}(\mu)$, elle est donc indépendante de la syntaxe de ces formules.

Exemple 2.2.1 (Exemple des oiseaux)**.** *En reprenant la formulation simplifiée de l'exemple en logique propositionnelle des préliminaires :*

$$\begin{aligned}\psi_{ex} &= (\neg \mathtt{oiseau} \vee \mathtt{vole}) \wedge (\neg \mathtt{manchot} \vee \mathtt{oiseau}) \\ \mu_{ex} &= \mathtt{manchot} \wedge \neg \mathtt{vole}\end{aligned}$$

Les deux modèles de μ sont à « distance » 1 de ceux de ψ_{ex}, soit en partant du modèle de ψ_{ex} où Max est un manchot, un oiseau et qu'il vole puis en remettant en cause le fait qu'il vole, soit en partant du modèle de ψ_{ex} dans lequel Max n'est ni un manchot, ni un oiseau mais il vole et on remet en cause le fait que ce n'est pas un manchot. Ainsi $\mathrm{Mod}(\psi_{ex} \dot{+}_D \mu_{ex})) = \mathrm{Mod}(\mu_{ex})$.

Généralisation avec une « distance » quelconque

Les seules propriétés de d_{H} utilisées dans la preuve de la proposition 2.2.1 pour montrer que $\leqslant_{\bullet}^{d_{\mathrm{H}}}$ est une assignation fidèle sont :

- $d_{\mathrm{H}}(I, I) = 0$ pour tout $I \in \Omega$.

- $d_{\mathrm{H}}(A, I) > 0$ pour tout $A \subseteq \Omega$ et $I \in \Omega$. Étant donné que Ω est fini, ceci est équivalent au fait que pour tous $I, J \in \Omega$, si $I \neq J$ alors $d_{\mathrm{H}}(I, J) > 0$.

Ainsi, une assignation fidèle peut être définie à partir d'une « distance » quelconque :

Proposition 2.2.2. *Étant donné une « distance » sur Ω, la fonction $\leqslant^d_{\cdot}$ définie pour toute formule ψ et $I, J \in \Omega$ par :*

$$I \leqslant^d_{\psi} J \quad ssi \quad d(\mathrm{Mod}(\psi), I) \leqslant d(\mathrm{Mod}(\psi), J) \tag{2.2}$$

est une assignation fidèle satisfaisant (A1), (A2) et (A3).

Dans la suite, on notera $\dot{+}^d$ l'opérateur de révision défini par l'équation (2.1) avec l'assignation fidèle définie par l'équation (2.2) avec la « distance » d. On a alors pour toute formules ψ et μ :

$$\mathrm{Mod}(\psi \dot{+}^d \mu) = \underset{d(\mathrm{Mod}(\psi),\cdot)}{\mathrm{Min}} \mathrm{Mod}(\mu) \tag{2.3}$$

Le choix de « distances » différentes de d_{H} permet d'exprimer des préférences différentes pour les connaissances à préserver lors de la révision. Par exemple, la « distance »

$$d(I, J) = \sum_{p \in \mathcal{V},\, p^I \neq p^J} w_p \tag{2.4}$$

qui généralise la distance de Hamming (d_{H} correspond aux cas particulier où $w_p = 1$ pour tout $p \in \mathcal{V}$), permet de donner des « poids » plus ou moins importants aux variables. C'est-à-dire que lors de la révision de ψ par μ, si $w_{p_i} < w_{p_j}$, un modèle I de μ obtenu à partir d'un modèle I' de ψ en modifiant la valeur de la variable p_i sera préféré à un modèle J de μ obtenu à partir d'un modèle J' de ψ en modifiant la valeur de la variable p_j. Les connaissances de ψ dépendant de p_j ont donc la priorité sur celles dépendant de p_i, dans le sens où une dépendance établie par ψ entre la valeur de p_i et celles des autres variables peut ne plus être respectée dans I, contrairement à une dépendance entre la valeur de p_j et celles des autres variables (à l'exclusion de p_i).

Exemple 2.2.2 (Exemple des oiseaux suite)**.** *On reprend l'exemple des oiseaux, avec la « distance » d de la forme* (2.4) *avec les coefficient suivants :*

- $w_1 = 1$ *pour la variable* $p_1 =$ `vole`,
- $w_2 = 1$ *pour la variable* $p_2 =$ `oiseau`,
- $w_3 = 2$ *pour la variable* $p_3 =$ `manchot`.

Dans ψ, `oiseau` *est une conséquence de* `manchot` *et* $\neg$`oiseau` *est une conséquence de* $\neg$`vole`. $w_3 < w_1$ *signifie que l'on préfère remettre en cause les conséquences de* $\neg$`vole` *plutôt que les conséquences de* `manchot`. *Ainsi* $\psi_{ex} \dot{+} \mu$ *est équivalent à* `manchot` $\wedge$ `oiseau` $\wedge \neg$`vole`.

Remarque 2.2.2. *La caractérisation de la révision par une assignation fidèle réduit le calcul de la révision d'une BC Ψ par une BC M à celle d'une minimisation sous contrainte :*

$$\textit{Trouver les } I \textit{ minimales pour } \leqslant_{\Psi} \textit{ sous la contrainte } I \in \mathrm{Mod}(M)$$

sous contrainte que ces interprétations soient des modèles de M. Dans le cas d'une assignation fidèle $\leqslant^d_{\cdot}$ *définie à partir d'une distance, le calcul de* $\Psi \dot{+}^d M$ *équivaut à :*

$$\textit{Trouver les } x \textit{ minimisant } d(\mathrm{Mod}(\Psi), x) \textit{ sous la contrainte } x \in \mathrm{Mod}(M)$$

Il existe des méthodes de résolution efficaces pour de plusieurs familles de problèmes d'optimisation, en particulier pour des problèmes numériques. En section 2.4.2, nous proposons une réduction de la révision de connaissances numériques sous forme de problèmes d'optimisation linéaire.

2.3 Révision dans d'autres formalismes

Bien que les postulats AGM originaux de la révision se placent dans un cadre plus général que la logique propositionnelle, plusieurs hypothèses sont faites sur la logique utilisée [G. Flouris, 2005], [Konieczny, 1999], dont l'existence de la négation des formules dans $\mathcal{L}$. Or nous nous intéressons à des formalismes où ces hypothèses ne sont pas vérifiées, par exemple pour les logiques de descriptions (voir section 2.5). [Qi *et al.*, 2006] proposent une reformulation des postulats de Katsuno et Mendelzon (voir section 2.2.1) dans le cadre de la révision de connaissances exprimées dans une logique de descriptions. Cependant ces postulats ne s'appuient pas sur des caractéristiques propres aux logiques de descriptions et on les considérera pour les révisions dans toute logique satisfaisant les hypothèses faites dans les préliminaires (section 1.1). Les postulats de [Qi *et al.*, 2006] s'expriment sous forme de contraintes sur les modèles de la révision, faisant ainsi abstraction de la syntaxe :

(G1) $\mathrm{Mod}(\Psi \dot{+} \mathrm{M}) \subseteq \mathrm{Mod}(\mathrm{M})$.

(G2) Si $\mathrm{Mod}(\Psi) \cap \mathrm{Mod}(\mathrm{M}) \neq \emptyset$ alors $\mathrm{Mod}(\Psi \dot{+} \mathrm{M}) = \mathrm{Mod}(\Psi) \cap \mathrm{Mod}(\mathrm{M})$.

(G3) Si $\mathrm{Mod}(\mathrm{M}) \neq \emptyset$ alors $\mathrm{Mod}(\Psi \dot{+} \mathrm{M}) \neq \emptyset$.

(G4) Si $\mathrm{Mod}(\Psi_1) = \mathrm{Mod}(\Psi_2)$ et $\mathrm{Mod}(\mathrm{M}_1) = \mathrm{Mod}(\mathrm{M}_2)$
alors $\mathrm{Mod}(\Psi_1 \dot{+} \mathrm{M}_1) = \mathrm{Mod}(\Psi_2 \dot{+} \mathrm{M}_2)$.

(G5) $\mathrm{Mod}(\Psi \dot{+} \mathrm{M}_1) \cap \mathrm{Mod}(\mathrm{M}_2) \subseteq \mathrm{Mod}(\Psi \dot{+} (\mathrm{M}_1 \cup \mathrm{M}_2))$.

(G6) Si $\mathrm{Mod}(\Psi \dot{+} \mathrm{M}_1) \cap \mathrm{Mod}(\mathrm{M}_2) \neq \emptyset$
alors $\mathrm{Mod}(\Psi \dot{+} (\mathrm{M}_1 \cup \mathrm{M}_2)) \subseteq \mathrm{Mod}(\Psi \dot{+} \mathrm{M}_1) \cap \mathrm{Mod}(\mathrm{M}_2)$.

Où Ψ, Ψ_1, Ψ_2, M, M_1 et M_2 sont des BC.

Ces postulats étendent bien ceux de Katsuno et Mendelzon ((Ri) est généralisé en (Gi) pour tout i), on peut le montrer en s'appuyant sur le fait que pour des formules ψ et μ, $\psi \models \mu$ ssi $\mathrm{Mod}(\psi) \subseteq \mathrm{Mod}(\mu)$, et $\mathrm{Mod}(\{\psi\} \cup \{\mu\}) = \mathrm{Mod}(\psi \wedge \mu) = \mathrm{Mod}(\psi) \cap \mathrm{Mod}(\mu)$.

2.3.1 Généralisation des assignations fidèles

En logique propositionnelle, [Katsuno et Mendelzon, 1991b] donnent une représentation d'opérateur de révision par une assignation fidèle qui associe à toute formule un pré-ordre sur Ω vérifiant les propriétés (A1), (A2) et (A3) (définition 2.2.1). Cette définition est généralisée en appelant assignation fidèle une fonction qui associe à toute base de connaissance un pré-ordre sur Ω vérifiant les propriétés équivalentes. Cependant, contrairement à ce qui se passe en logique propositionnelle, étant donné un ensemble d'interprétations $A \subseteq \Omega$, il n'existe pas toujours dans le cadre général de base de connaissances Ψ telle que $\mathrm{Mod}(\Psi) = A$. Pour définir la révision à partir de l'ensemble de ses modèles, il faut donc s'assurer que la logique permette d'exprimer cet ensemble d'interprétations. On note $\mathcal{P}$ l'ensemble des parties de Ω exprimables par des bases de connaissances :

$$\mathcal{P} = \{\mathrm{Mod}(\Psi) \mid \Psi \text{ est une BC sur } \mathcal{L}\}$$

Vu que pour deux BC Ψ et M, $\mathrm{Mod}(\Psi \cup \mathrm{M}) = \mathrm{Mod}(\Psi) \cap \mathrm{Mod}(\mathrm{M})$, $\mathcal{P}$ est stable par $\cap$.

Définition 2.3.1 (Généralisation des assignations fidèles)**.** *Une assignation fidèle est une fonction $\leqslant_{\cdot}$ qui associe à une BC Ψ un pré-ordre $\leqslant_\Psi$ sur Ω tel que :*

$$\begin{array}{ll}
(\mathrm{A1}') & \textit{Si } I, I' \in \mathrm{Mod}(\Psi) \textit{ alors non}(I <_\Psi I') \\
(\mathrm{A2}') & \textit{Si } I \in \mathrm{Mod}(\Psi) \textit{ et } I' \notin \mathrm{Mod}(\Psi) \textit{ alors } I <_\Psi I' \\
(\mathrm{A3}') & \textit{Si } \Psi \equiv M \textit{ alors } \leqslant_\Psi = \leqslant_M
\end{array}$$

Les postulats (A1′), (A2′) et (A3′) généralisent respectivement les postulats $(A1)$, $(A2)$ et $(A3)$. Le théorème 2.2.1 (page 21) qui définit un opérateur de révision $\dot{+}$ à l'aide d'une assignation fidèle $\leqslant_{\cdot}$ fait l'hypothèse implicite que le postulat suivant est vérifié :

$$(\mathrm{A4}') \quad \text{Pour } A \in \mathcal{P}, \ \operatorname*{Min}_{\leqslant_\Psi} A \in \mathcal{P}$$

Ce postulat est toujours vérifié en logique propositionnelle, en effet, pour tout ensemble d'interprétations $A \subseteq \mathcal{U}$, il existe une formule φ telle que $\mathrm{Mod}(\varphi) = A$. Ce n'est pas vrai dans le cadre général, on sera notamment confronté à ce problème avec les logiques de descriptions en section 2.6.2. L'équation (2.1), page 21, peut alors être généralisée dans le cadre plus général considéré ici :

$$\mathrm{Mod}(\Psi \dot{+} \mathrm{M}) = \operatorname*{Min}_{\leqslant_\Psi} (\mathrm{Mod}(\mathrm{M})) \tag{2.5}$$

Une autre hypothèse est faite implicitement dans le théorème 2.2.1 :

$$(\mathrm{A5}') \quad \text{Pour } A \in \mathcal{P}, \ \text{si } A \neq \emptyset \text{ alors } \operatorname*{Min}_{\leqslant_\Psi} A \neq \emptyset$$

Comme pour (A4′), ce postulat est toujours vérifié en logique propositionnelle puisque $\mathcal{U}$ est fini et donc tout ensemble $A \in \mathcal{P}$ l'est aussi, mais il est faux en général. Par exemple, avec $\mathcal{U} = \mathbb{R}$,

$\mathcal{P} = 2^{\mathcal{U}}$, $\leqslant$ l'ordre usuel sur $\mathbb{R}$ et $A =]1, 2]$, $\underset{\leqslant}{\mathrm{Min}}\, A = \emptyset$. Ce postulat est nécessaire pour que l'opérateur de révision défini selon l'équation (2.5) vérifie le postulat (G3) :

Proposition 2.3.1 ([Cojan et Lieber, 2009b]).

1. *Si $\leqslant_{\cdot}$ est une assignation fidèle qui associe à toute BC Ψ un pré-ordre total $\leqslant_{\Psi}$ sur Ω et qui satisfait les postulats* (A1′) *à* (A4′)*, alors l'opérateur de révision $\dot{+}$ obtenu par l'équation* (2.5) *satisfait les postulats* (G1)*,* (G2)*,* (G4)*,* (G5) *et* (G6)*.*
2. *Si $\leqslant_{\cdot}$ satisfait en plus la propriété* (A5′)*, alors l'opérateur de révision $\dot{+}$ satisfait aussi le postulat* (G*3*)*.*

Démonstration. La preuve du « si » (1.) de [Katsuno et Mendelzon, 1991b] reste valide ici. Pour le postulat (G3) la propriété (A5′) est cependant implicite car toujours satisfaite en logique propositionnelle finie. □

Pour la réciproque du théorème 2.2.1, [Katsuno et Mendelzon, 1991b] définissent pour tout opérateur de révision $\dot{+}$ une assignation fidèle qui satisfait l'équation (2.1). Pour toute formule ψ un pré-ordre $\leqslant_{\psi}$ est défini sur Ω en fonction du résultat de la révision de ψ par les formules μ telles que $\mathrm{Mod}(\mu) = \{I_1, I_2\}$ pour $I_1, I_2 \in \Omega$. L'existence de telles formules est vérifiée en logique propositionnelle mais est fausse en général, comme c'est le cas dans les formalismes des sections 2.4 et 2.5. L'intérêt de la révision par ces formules viens du fait que $\{I_1, I_2\} \subseteq \mathrm{Mod}(\mu)$ et $\mathrm{Mod}(\psi \dot{+} \mu) \cap \{I_1, I_2\} \neq \emptyset$, ce qui permet de savoir si l'une des ces deux interprétations est sélectionnée plutôt que l'autre lors de la révision de ψ, ce qui signifie qu'elle est plus proche des modèles de ψ suivant la notion de minimalité du changement associée à $\dot{+}$. L'existence de formules vérifiant ces deux dernière propriétés est suffisant pour définir une relation $\leqslant_{\psi}$ totale sur Ω. La démonstration de la transitivité de cette relation s'appuie sur des formules satisfaisant les relations identiques avec les triplets d'interprétations.

Proposition 2.3.2 ([Cojan et Lieber, 2009b]). *Si $\dot{+}$ est un opérateur de révision satisfaisant les postulats* (G*1*) *à* (G*6*) *et tel que pour toute BC Ψ et $I_1, I_2, I_3 \in \Omega$, il existe une BC M telle que :*

$$\begin{cases} \{I_1, I_2, I_3\} \subseteq \mathrm{Mod}(M) \\ \mathrm{Mod}(\Psi \dot{+} M) \cap \{I_1, I_2, I_3\} \neq \emptyset \end{cases}$$

Alors il existe une assignation fidèle $\leqslant_{\cdot}$ qui vérifie l'équation (2.5)*.*

Démonstration. On peut déjà remarquer que la propriété spécifiée dans la proposition 2.3.2 implique que pour toute BC Ψ et $I_1, I_2 \in \Omega$, il existe une BC M telle que $\{I_1, I_2\} \subseteq \mathrm{Mod(M)}$ et $\mathrm{Mod}(\Psi \dot{+} \mathrm{M}) \cap \{I_1, I_2\} \neq \emptyset$ (cela correspond au cas particulier $I_3 = I_2$).

On établit alors que $I_1 \leqslant_{\Psi} I_2$ si $I_1 \in \mathrm{Mod}(\Psi \dot{+} \mathrm{M})$. Cela ne dépend pas de la formule M choisie, en effet si M_1 et M_2 vérifient toutes deux les propriétés ci-dessus. On a $\mathrm{Mod}(\Psi \dot{+} \mathrm{M}_1) \cap \mathrm{Mod}(\mathrm{M}_2) \neq \emptyset$ car $\{I_1, I_2\} \subseteq \mathrm{Mod}(\mathrm{M}_2)$ et $\mathrm{Mod}(\Psi \dot{+} \mathrm{M}_1) \cap \{I_1, I_2\} \neq \emptyset$. Donc d'après (G5) et (G6), $\mathrm{Mod}(\Psi \dot{+} \mathrm{M}_1) \cap \mathrm{Mod}(\mathrm{M}_2) = \mathrm{Mod}(\Psi \dot{+} (\mathrm{M}_1 \cup \mathrm{M}_2))$, on a de même $\mathrm{Mod}(\Psi \dot{+} \mathrm{M}_2) \cap \mathrm{Mod}(\mathrm{M}_1) =$

$\mathrm{Mod}(\Psi \dot{+} (\mathrm{M}_1 \cup \mathrm{M}_2))$. Ainsi $\mathrm{Mod}(\Psi \dot{+} \mathrm{M}_1) \cap \mathrm{Mod}(\mathrm{M}_2) = \mathrm{Mod}(\Psi \dot{+} \mathrm{M}_2) \cap \mathrm{Mod}(\mathrm{M}_1)$, et puisque $\{I_1, I_2\} \subseteq \mathrm{Mod}(\mathrm{M}_1)$ et $\{I_1, I_2\} \subseteq \mathrm{Mod}(\mathrm{M}_2)$, $\mathrm{Mod}(\Psi \dot{+} \mathrm{M}_1) \cap \{I_1, I_2\} = \mathrm{Mod}(\Psi \dot{+} \mathrm{M}_2) \cap \{I_1, I_2\}$. On a donc $I_1 \in \mathrm{Mod}(\Psi \dot{+} \mathrm{M}_1)$ ssi $I_1 \in \mathrm{Mod}(\Psi \dot{+} \mathrm{M}_2)$.

La démonstration du fait que la fonction $(\Psi \mapsto \leqslant_\Psi)$ satisfait l'équation (2.5) est obtenue en remplaçant dans la démonstration de [Katsuno et Mendelzon, 1991b] les formules $\mathrm{form}(\{I_1, I_2, I_3\})$ par les bases de connaissances M mentionnées dans l'énoncé de la proposition 2.3.2.

Le postulat (A4′) est impliqué par le fait que pour deux bases de connaissances Ψ et M, $\underset{\leqslant_\Psi}{\mathrm{Min}}\,(\mathrm{Mod}(\mathrm{M})) = \mathrm{Mod}(\Psi \dot{+} \mathrm{M})$ et $\Psi \dot{+} \mathrm{M} \in \mathcal{L}$.

Le postulat (G3) garantit que $\leqslant_{\cdot}$ satisfait bien la propriété (A5′). □

2.3.2 Opérateurs de révision définis à l'aide d'une « distance »

Des opérateurs de révision peuvent être définis à l'aide de « distances » sur Ω comme en logique propositionnelle (voir section 2.2.3). L'équation (2.2) (page 24) permettant de définir une assignation en logique propositionnelle peut être reprise dans ce cadre plus général en posant pour une BC Ψ et une « distance » d sur Ω :

$$I \leqslant^d_\Psi J \quad \text{si} \quad d(\mathrm{Mod}(\Psi), I) \leqslant d(\mathrm{Mod}(\Psi), J) \tag{2.6}$$

La fonction $\leqslant^d_{\cdot}$: $\Psi \mapsto \leqslant^d_\Psi$ vérifie les postulats (A1′) et (A3′), mais des conditions sur d sont cependant nécessaires pour que $\leqslant^d_{\cdot}$ vérifie les postulats (A2′), (A4′) et (A5′) :

(A2′) : Si $I \in \mathrm{Mod}(\Psi)$, alors $d(\mathrm{Mod}(\Psi), I) = 0$. Ainsi, $\leqslant^d_{\cdot}$ vérifie (A2′) ssi pour $I' \notin \mathrm{Mod}(\Psi)$, $d(\mathrm{Mod}(\Psi), I') > 0$, ou en prenant la contraposée $d(\mathrm{Mod}(\Psi), I') = 0$ ssi $I' \in \mathrm{Mod}(\Psi)$. C'est-à-dire que les ensembles dans $\mathcal{P}$ sont fermés pour d (pour tout $A \in \mathcal{P}$, $\{x \in \mathcal{U} \mid d(A, x) = 0\} \subseteq A$).

(A4′) : Ce postulat se réécrit en prenant les notations de la section 1.2.2 : pour $\psi \in \mathcal{L}$ et $B \in \mathcal{P}$, $\underset{d(\mathrm{Mod}(\psi),\cdot)}{\mathrm{Min}}\, B \in \mathcal{P}$. Ainsi $\leqslant^d_{\cdot}$ satisfait (A4′) ssi pour $A, B \in \mathcal{P}$, $\underset{d(A,\cdot)}{\mathrm{Min}}\, B \in \mathcal{P}$.

(A5′) : De même $\leqslant^d_{\cdot}$ satisfait (A5′) ssi pour $A, B \in \mathcal{P}$, $B \neq \emptyset$ implique que $\underset{d(A,\cdot)}{\mathrm{Min}}\, B \neq \emptyset$. Par exemple, si d prend des valeurs discrètes, ou si d vérifie l'inégalité triangulaire et les ensembles de $\mathcal{P}$ sont des compacts de (Ω, d). (suivant la topologie engendrée par les boules ouvertes $B(x, \varepsilon) = \{y \in \mathcal{U} \mid d(y, x) < \varepsilon\}$).

2.3.3 Révision sur l'adhérence

Pour s'affranchir en partie de ces problèmes, [Schwind, 2010] effectue la révision sur l'adhérence des ensembles de modèles, étant donné des BC Ψ et M, l'équation (2.3) (page 24) qui définit $\Psi \dot{+}^d \mathrm{M}$, devient :

$$\mathrm{Mod}(\Psi \dot{+}^d \mathrm{M}) = \underset{d(\overline{\mathrm{Mod}(\Psi)},\cdot)}{\mathrm{Min}}\, \overline{Mod(\mathrm{M})} \tag{2.7}$$

où, étant donné un ensemble A, $\overline{A} = \{u \in \mathcal{U} \mid d(A,u) = 0\}$ est *l'adhérence* de A (voir les préliminaires). On peut montrer qu'un tel opérateur satisfait le postulat :

$$\text{(Ferm)} \quad \mathrm{Mod}(\Psi \dot{+} \mathrm{M}) = \overline{\mathrm{Mod}(\Psi \dot{+} \mathrm{M})}$$

ainsi que les postulats $(\overline{\mathrm{G1}})$, $(\overline{\mathrm{G2}})$, $(\overline{\mathrm{G4}})$, $(\overline{\mathrm{G5}})$ et $(\overline{\mathrm{G6}})$:

$(\overline{\mathrm{G1}})$ $\mathrm{Mod}(\Psi \dot{+} \mathrm{M}) \subseteq \overline{\mathrm{Mod}(\mathrm{M})}$.

$(\overline{\mathrm{G2}})$ Si $\overline{\mathrm{Mod}(\Psi)} \cap \overline{\mathrm{Mod}(\mathrm{M})} \neq \emptyset$ alors $\mathrm{Mod}(\Psi \dot{+} \mathrm{M}) = \overline{\mathrm{Mod}(\Psi)} \cap \overline{\mathrm{Mod}(\mathrm{M})}$.

$(\overline{\mathrm{G3}})$ Si $\mathrm{Mod}(\mathrm{M}) \neq \emptyset$ alors $\mathrm{Mod}(\Psi \dot{+} \mathrm{M}) \neq \emptyset$.

$(\overline{\mathrm{G4}})$ Si $\overline{\mathrm{Mod}(\Psi_1)} = \overline{\mathrm{Mod}(\Psi_2)}$ et $\overline{\mathrm{Mod}(\mathrm{M}_1)} = \overline{\mathrm{Mod}(\mathrm{M}_2)}$
alors $\mathrm{Mod}(\Psi_1 \dot{+} \mathrm{M}_1) = \mathrm{Mod}(\Psi_2 \dot{+} \mathrm{M}_2)$.

$(\overline{\mathrm{G5}})$ $\mathrm{Mod}(\Psi \dot{+} \mathrm{M}_1) \cap \overline{\mathrm{Mod}(\mathrm{M}_2)} \subseteq \mathrm{Mod}(\Psi \dot{+} ((\mathrm{M}_1 \dot{+} \mathrm{M}_1) \cup (\mathrm{M}_2 \dot{+} \mathrm{M}_2)))$.

$(\overline{\mathrm{G6}})$ Si $\mathrm{Mod}(\Psi \dot{+} \mathrm{M}_1) \cap \overline{\mathrm{Mod}(\mathrm{M}_2)} \neq \emptyset$
alors $\mathrm{Mod}(\Psi \dot{+} ((\mathrm{M}_1 \dot{+} \mathrm{M}_1) \cup (\mathrm{M}_2 \dot{+} \mathrm{M}_2))) \subseteq \mathrm{Mod}(\Psi \dot{+} \mathrm{M}_1) \cap \overline{\mathrm{Mod}(\mathrm{M}_2)}$.

Remarque 2.3.1. *L'utilisation des termes* $(M_1 \dot{+} M_1)$ *et* $(M_2 \dot{+} M_2)$ *dans les postulats* $(\overline{\mathrm{G5}})$ *et* $(\overline{\mathrm{G6}})$ *peut sembler étrange, il s'agit en fait d'une astuce pour obtenir la fermeture des modèles de* M_1 *(et de* M_2 *respectivement) :*

$$\mathrm{Mod}(M_1 \dot{+} M_1) = \overline{\mathrm{Mod}(M_1)}$$

La reformulation du postulat (G5) *en* $(\overline{\mathrm{G5}})$ *est nécessaire pour être compatible avec* $(\overline{\mathrm{G1}})$ *et* $(\overline{\mathrm{G2}})$. *En effet, si on considère un opérateur de révision* $\dot{+}$ *sur* $\mathcal{U} = \mathbb{R}$ *qui satisfait* $(\overline{\mathrm{G1}})$ *et* $(\overline{\mathrm{G2}})$ *et des formules* Ψ, M_1 *et* M_2 *telles que :*

$$\mathrm{Mod}(\Psi) = \mathcal{U} = \mathbb{R} \qquad \mathrm{Mod}(M_1) = \mathbb{Q} \qquad \mathrm{Mod}(M_2) = \mathbb{R} \setminus \mathbb{Q}$$

$\mathrm{Mod}(\Psi \dot{+} M_1) = \mathbb{R}$ *d'après le postulat* $(\overline{\mathrm{G2}})$ *donc* $\mathrm{Mod}(\Psi \dot{+} M_1) \cap \overline{\mathrm{Mod}(M_2)} = \mathbb{R}$. *Or* $\mathrm{Mod}(M_1 \cup M_2) = \mathrm{Mod}(M_1) \cap \mathrm{Mod}(M_2) = \emptyset$, *donc d'après* $(\overline{\mathrm{G1}})$ $\mathrm{Mod}(\Psi \dot{+} (M_1 \cup M_2)) = \emptyset$, *et*

$$\mathbb{R} = \mathrm{Mod}(\Psi \dot{+} M_1) \cap \overline{\mathrm{Mod}(M_2)} \not\subseteq \mathrm{Mod}(\Psi \dot{+} (M_1 \cup M_2)) = \emptyset$$

En revanche $\mathrm{Mod}(M_1 \dot{+} M_1) = \overline{\mathrm{Mod}(M_1)} = \mathbb{R}$ *et* $\mathrm{Mod}(M_2 \dot{+} M_2) = \overline{\mathrm{Mod}(M_2)} = \mathbb{R}$, *donc*

$$\mathbb{R} = \mathrm{Mod}((M_1 \dot{+} M_1) \cup (M_2 \dot{+} M_2)) \subseteq \mathrm{Mod}(\Psi \dot{+} ((M_1 \dot{+} M_1) \cup (M_2 \dot{+} M_2))) = \mathbb{R}$$

$(\overline{\mathrm{G5}})$ *est donc bien vérifié.*

L'utilisation des termes $(M_1 \dot{+} M_1)$ *et* $(M_2 \dot{+} M_2)$ *dans* $(\overline{\mathrm{G6}})$ *n'est pas nécessaire pour sa*

satisfaction, mais cela permet de conserver la symétrie avec le postulat $\left(\overline{\text{G5}}\right)$.

Le postulat $\left(\overline{\text{G3}}\right)$ n'est pas toujours vérifié par $\dot{+}^d$. Par exemple, si $\mathcal{U} = \mathbb{R}^2$,

$$\begin{aligned}\text{Mod}(\Psi) &= \{(x, y) \in \mathbb{R}^2 \mid x > 0, \text{ et } y = 1/x\}\\ \text{Mod}(\text{M}) &= \{(x, y) \in \mathbb{R}^2 \mid y = 0\}\end{aligned}$$

Pour la « distance » d définie pour $u_1 = (x_1, y_1), u_2 = (x_2, y_2) \in \mathbb{R}$, par $d(u_1, u_2) = |x_2 - x_1| + |y_2 - y_1|$, $d(\text{Mod}(\Psi), \text{Mod}(\text{M})) = 0$, mais elle n'est jamais atteinte donc $\text{Mod}(\Psi \dot{+}^d \text{M}) = \emptyset$.

Pour assurer la satisfaction du postulat $\left(\overline{\text{G3}}\right)$, [Schwind, 2010] fait l'hypothèse supplémentaire que les ensembles représentés par les formules sont des compacts, ce qui garantit la satisfaction de ce postulat pour les « distances » vérifiant l'inégalité triangulaire : pour $u, v, w \in \mathcal{U}$, $d(u, w) \leqslant d(u, v) + d(v, w)$ (c'est le cas de la distance d définie ci-dessus).

2.3.4 Révision floue

Comme nous l'avons vu en section 2.3.2, l'opérateur de révision $\dot{+}^d$ peut ne pas vérifier le postulat (G3) parce que la distance entre ensembles de modèles peut ne pas être atteinte. La révision sur l'adhérence de ces ensembles, présentée dans la section précédente, ne résout pas ce problème. Pour contourner ce problème nous proposons ici de considérer un seuil flou sur la distance dans l'espace des interprétations. Cette partie reprend des travaux publiés dans [Cojan et Lieber, 2008a].

α-révision

Afin de contourner les problèmes de satisfaction du postulat (G3), nous allons assouplir la condition sur la distance de notre opérateur de révision. Étant donnée une distance d, si $\Delta = d(\text{Mod}(\Psi), \text{Mod}(\text{M}))$:

$$u \in \text{Mod}(\Psi \dot{+}^d \text{M}) \quad \text{ssi} \quad u \in \text{Mod}(\text{M}) \text{ et } d(\text{Mod}(\Psi), u) = \Delta$$

Au lieu de ne prendre que les u pour laquelle la distance est atteinte, nous allons laisser une marge $\varepsilon \geqslant 0$, c'est-à-dire que

$$u \in \text{Mod}(\Psi \dot{+}^d_\varepsilon \text{M}) \quad \text{ssi} \quad u \in \text{Mod}(\text{M}) \text{ et } d(\text{Mod}(\Psi), u) \leqslant \Delta + \varepsilon$$

Pour nous placer dans le formalisme habituel du flou on considérera une mesure de similarité $\mathcal{S}$ plutôt qu'une « distance ».

Exemple 2.3.1. *On peut, par exemple, définir une mesure de similarité* $\mathcal{S}$ *grâce à une « distance »* d *en posant pour* $u, v \in \mathcal{U}$ *:*

$$\mathcal{S}(u, v) = e^{-d(u,v)}$$

La similarité entre deux éléments u, v, est maximale ($\mathcal{S}(u,v) = 1$) lorsque $u = v$, et elle décroît vers 0 à mesure que les éléments u, v s'éloignent. Inversement, une « distance » d peut être définie à partir d'une mesure de similarité $\mathcal{S}$ en posant $d(u,v) = \ln(\mathcal{S}(u,v))$ pour $u, v \in \mathcal{U}$ (si $\mathcal{S}$ peut prendre la valeur 0, alors d peut prendre la valeur $+\infty$).

On remplacera le paramètre $\varepsilon \in [0, +\infty]$ par $\alpha \in [0,1]$ en réécrivant la relation précédente :

$$u \in \mathrm{Mod}(\Psi \dot{+}^d_\alpha \mathrm{M}) \quad \text{ssi} \quad u \in \mathrm{Mod}(\mathrm{M}) \text{ et } \mathcal{S}(\mathrm{Mod}(\Psi), u) \geqslant \Sigma \times \alpha$$

où $\Sigma = \mathcal{S}(\mathrm{Mod}(\Psi), \mathrm{Mod}(\mathrm{M})) = e^{-\Delta}$ et $\alpha = e^{-\varepsilon} \in [0,1]$.

Définition 2.3.2. *On définit donc $\dot{+}^d_\alpha$ tel que :*

$$\mathrm{Mod}(\Psi \dot{+}^d_\alpha M) = \{x \in \mathrm{Mod}(M) \mid \mathcal{S}(\mathrm{Mod}(\Psi), x) \geqslant \Sigma \times \alpha\}$$

Proposition 2.3.3. *Comme attendu $\dot{+}^d_1$ et $\dot{+}^d$ donnent des résultats logiquement équivalents, et pour tous $1 \geqslant \alpha \geqslant \beta \geqslant 0$:*

$$\begin{aligned} \mathrm{Mod}(\Psi \dot{+}^d_1 M) \subseteq \mathrm{Mod}(\Psi \dot{+}^d_\alpha M) \subseteq \mathrm{Mod}(\Psi \dot{+}^d_\beta M) \subseteq \mathrm{Mod}(\Psi \dot{+}^d_0 M) \\ \textit{où :} \quad \mathrm{Mod}(\Psi \dot{+}^d_1 M) = \mathrm{Mod}(\Psi \dot{+}^d M) \\ \mathrm{Mod}(\Psi \dot{+}^d_0 M) = \mathrm{Mod}(M) \end{aligned}$$

Proposition 2.3.4. *Pour $\alpha \in [0,1]$, $\dot{+}^d_\alpha$ vérifie les postulats* (G1), (G4) *et* (G5).

Proposition 2.3.5. *Pour $\alpha < 1$, $\dot{+}^d_\alpha$ vérifie le postulat* (G3).

En revanche, dans le cadre général pour $\alpha < 1$, $\dot{+}^d_\alpha$ ne vérifie pas (G2) et (G6). Pour le postulat (G2), l'ensemble d'interprétations retenu peut être trop grand, il l'est même d'autant plus que α est petit.

Exemple 2.3.2. *Avec $\mathcal{U} = \mathbb{Z}$ et $d(x,y) = |y-x|$, soit Ψ, M tels que $\mathrm{Mod}(\Psi) = \{0\}$ et $\mathrm{Mod}(M) = \{0,1\}$, $\Psi \wedge M$ est satisfiable. Alors $d(\Psi, M) = 0$ donc $\mathcal{S}(\Psi, M) = 1$ et*

$$\mathrm{Mod}\left(\Psi \dot{+}^d_{e^{-1}} M\right) = \left\{x \in \mathrm{Mod}(M) \mid \mathcal{S}(\Psi, x) \geqslant e^{-1}\right\} = \{0,1\} \neq \mathrm{Mod}(\Psi \wedge M)$$

Dans cet exemple $\dot{+}^d_{e^{-1}}$ ne vérifie pas (G2) *alors que $\dot{+}^d_1$ si.*

Pour le postulat (G6), l'ensemble d'interprétations retenu peut aussi être trop grand. Cela n'est pas surprenant vu que ce postulat exprime d'après [Katsuno et Mendelzon, 1991b] le changement minimal, or, le principe de l'α-révision est justement de ne pas se restreindre au minimum.

Exemple 2.3.3. *Toujours avec $\mathcal{U} = \mathbb{Z}$ et $d(x,y) = |y-x|$, soient Ψ, M, Φ des BC telles que $\mathrm{Mod}(\Psi) = \{0\}$ et $\mathrm{Mod}(M) = \{1,2,3\}$ et $\mathrm{Mod}(\Phi) = \{2,3\}$. $\mathcal{S}(\Psi, M) = e^{-1}$ et*

$\mathrm{Mod}\left(\Psi \dot{+}^d_{e^{-1}} M\right) = \left\{x \in \mathrm{Mod}(M) \mid \mathcal{S}(\Psi, x) \geqslant e^{-2}\right\} = \{1, 2\}$. $\left(\Psi \dot{+}^d_{e^{-1}} M\right) \wedge \Phi$ *est donc satisfiable. Mais* $\mathcal{S}(\Psi, M \wedge \Phi) = e^{-2}$ *donc*

$$\begin{aligned}\mathrm{Mod}\left(\Psi \dot{+}^d_{e^{-1}} (M \wedge \Phi)\right) &= \left\{x \in \mathrm{Mod}(M \wedge \Phi) \mid \mathcal{S}(\Psi, x) \geqslant e^{-3}\right\} = \{2, 3\} \\ &\not\subseteq \mathrm{Mod}\left(\left(\Psi \dot{+}^d_{e^{-1}} M\right) \wedge \Phi\right) = \{2\}\end{aligned}$$

Révision floue

Rappels sur les sous-ensembles flous. Dans [Zadeh, 1965], Lotfi Zadeh définit un *sous-ensemble flou* F d'un univers $\mathcal{U}$ comme étant une application F de $\mathcal{U}$ dans le segment $[0, 1]$ des réels. Pour $x \in \mathcal{U}$, $F(x)$ représente son degré d'appartenance à F. En particulier un sous-ensemble classique A de $\mathcal{U}$ est un sous-ensemble flou de $\mathcal{U}$:

$$x \in \mathcal{U} \mapsto A(x) = \begin{cases} 1 \text{ si } x \in A \\ 0 \text{ sinon} \end{cases}$$

On note $\mathcal{F}(\mathcal{U})$ l'ensemble des sous-ensembles flous de $\mathcal{U}$. L. Zadeh généralise l'intersection, le complémentaire et l'union aux sous-ensembles flous de $\mathcal{U}$ par les égalités suivantes où $x \in \mathcal{U}$:

$$\begin{aligned}&(F \cap G)(x) = \min(F(x), G(x)) && (\mathcal{U} \setminus F)(x) = 1 - F(x) \\ &(F \cup G)(x) = \max(F(x), G(x)) = (\mathcal{U} \setminus ((\mathcal{U} \setminus F) \cap (\mathcal{U} \setminus G)))(x)\end{aligned}$$

Pour $\alpha \in]0, 1]$, une α-coupe de F est un sous-ensemble classique de $\mathcal{U}$ défini par :

$$F_\alpha = \{x \in \mathcal{U} \mid F(x) \geqslant \alpha\}$$

Un sous-ensemble flou F définit ainsi une famille de sous-ensembles emboîtés $(F_\alpha)_{\alpha \in]0,1]}$ (si $\beta \geqslant \alpha$, alors $F_\beta \subseteq F_\alpha$). Réciproquement, une famille de sous-ensembles classiques emboîtés $(F_\alpha)_{\alpha \in]0,1]}$ définit un unique sous-ensemble flou F de $\mathcal{U}$ par :

$$F(x) = \sup\{\alpha \in]0, 1] \mid x \in F_\alpha\} \tag{2.8}$$

avec la convention $\sup \emptyset = 0$ dans $]0, 1]$ (d'où $F(x) = 0$ si x n'appartient à aucun F_α).

Révision floue avec des BC classiques. L'α-révision est définie par des ensembles d'interprétations emboîtés paramétrés par $\alpha \in]0, 1]$, ce qui motive la définition de révision floue $\Box^d$ telle que la révision floue d'une BC classique Ψ par une autre BC classique M donne une BC floue $\Psi \Box^d \mathrm{M}$ caractérisée par une extension floue $\mathrm{Mod}\left(\Psi \Box^d \mathrm{M}\right)$ telle que pour $\alpha \in]0, 1]$, l'α-coupe $\left(\mathrm{Mod}\left(\Psi \Box^d \mathrm{M}\right)\right)_\alpha = \mathrm{Mod}\left(\Psi \dot{+}^d_\alpha \mathrm{M}\right)$, soit, en reprenant la formule (2.8) pour tout $u \in \mathcal{U}$:

$$\mathrm{Mod}\left(\Psi \Box^d \mathrm{M}\right)(u) = \sup\left\{\alpha \in]0, 1] \mid u \in \mathrm{Mod}\left(\Psi \dot{+}^d_\alpha \mathrm{M}\right)\right\}$$

Par homogénéité, il faudrait étendre la révision floue à des BC floues. Cette question est laissée en perspectives.

2.3.5 Autres travaux concernant la révision dans d'autres formalismes

Plusieurs travaux portent sur l'extension des travaux sur la révision des connaissances à d'autres formalismes logiques.

Généralisation des postulats

Les postulats AGM [Alchourrón *et al.*, 1985] sont formulés dans un cadre plus large que la logique propositionnelle mais qui ne couvre pas le formalisme logique présenté en préliminaires. Ils font notamment l'hypothèse que le langage contient les connecteurs $\wedge$, $\vee$ et $\neg$ et que $\models$ contient les déductions logiques classiques. Certains formalismes qui nous intéressent ne vérifient pas ces hypothèses, c'est le cas du formalisme attributs-valeurs simples présenté en section 2.4.2, et de la LD $\mathcal{ALC}$. Or, [Flouris *et al.*, 2006] montre que l'existence d'un opérateur de contraction vérifiant les postulats AGM de la contraction n'est pas assurée si les hypothèses faites par [Alchourrón *et al.*, 1985] ne sont pas vérifiées, Cette remarque n'implique cependant pas qu'il n'existe pas d'opérateur de révision satisfaisant les postulats AGM dans ces formalismes. En effet l'identité de Harper, qui définit un opérateur de contraction à l'aide d'un opérateur de révision, contient le connecteur $\neg$, le théorème de [Alchourrón *et al.*, 1985] qui établit que l'on peut obtenir un opérateur de contraction satisfaisant les postulats AGM de la contraction à l'aide de l'identité de Harper et un opérateur de révision satisfaisant les postulats AGM de la révision n'est donc pas applicable.

Nous avons présenté au début de la section 2.3, les postulats de [Qi *et al.*, 2006] généralisant les postulats de [Katsuno et Mendelzon, 1991b]. Ces postulats sont des relations entre les modèles des BC fournies à un opérateur de révision et les modèles attendus de la BC obtenue. Cependant plusieurs auteurs [Fuhrmann, 1991], [Nebel, 1992], [Dixon et Wobcke, 1993] considèrent que les opérateurs satisfaisant ces postulats sont trop complexes à définir et à calculer en pratique et considèrent plutôt la révision de BC, où le résultat de la révision de Ψ par M est une BC constituée des formules de M et d'une partie des formules de Ψ, c'est-à-dire que $\mathrm{M} \subseteq (\Psi \dot{+} \mathrm{M}) \subseteq \Psi \cup \mathrm{M}$. [Fuhrmann, 1991] et [Flouris *et al.*, 2006] proposent des postulats pour caractériser ce type d'opérateur de révision.

Opérateurs de révision syntaxiques

Plusieurs algorithmes de calcul de révision de BC ont été décrits dans la littérature, par exemple [Haase *et al.*, 2005], [Halaschek-Wiener *et al.*, 2006], [Qi *et al.*, 2008] s'appuient sur des outils de débogage d'ontologies pour effectuer la révision de BC exprimées dans des logiques de descriptions. Nous donnons plus de détails sur ces travaux en section 2.5.5.

[Dixon et Wobcke, 1993, Williams, 1997] proposent un algorithme de révision de BC exprimées en logique du premier ordre (L1O). Puisque le problème de l'évaluation de la cohérence d'une BC de L1O n'est pas décidable, les auteurs proposent des méthodes approchées pour le calcul de la révision de Ψ par M qui garantissent que le résultat soit cohérent, sous condition que M est cohérente, mais qui ne garantissent pas que les modification apportées à Ψ (abandon de formule) soient minimales. Ces opérateurs de révision satisfont les postulats (G1), (G2), (G3), (G5) et (G6) mais seulement une version affaiblie du postulat (G4) [Qi *et al.*, 2006] :

$$(\mathrm{G4}') \qquad \text{Si} \quad \begin{cases} \mathrm{Mod}(\psi_i^1) = \mathrm{Mod}(\psi_i^2) & \text{pour tout } 1 \leqslant i \leqslant n \\ & \text{où} \quad \Psi_1 = \{\psi_1^1, \ldots, \psi_n^1\} \quad \text{et} \quad \Psi_2 = \{\psi_1^2, \ldots, \psi_n^2\}, \\ \mathrm{Mod}(\mathrm{M}_1) = \mathrm{Mod}(\mathrm{M}_2) & \end{cases}$$
$$\text{alors} \quad \mathrm{Mod}(\Psi_1 \dot{+} \mathrm{M}_1) = \mathrm{Mod}(\Psi_2 \dot{+} \mathrm{M}_2)$$

[Meyer *et al.*, 2005], [Qi *et al.*, 2006] proposent des méthodes de révision syntaxiques plus fines où les formules supprimées sont remplacées par des formules plus générales. Les opérateurs de révision présentés dans ces deux articles sont définis dans des logiques de descriptions particulières et exploitent la structure de ces logiques pour exprimer le résultat de la révision. L'opérateur de révision défini par [Qi *et al.*, 2006] satisfait les postulats (G1), (G2), (G3), (G4′), (G5) et (G6). Ce résultat n'est pas donné pour l'opérateur défini dans [Meyer *et al.*, 2005], mais il doit être identique.

Opérateurs de révision sémantiques

Les travaux portant sur des opérateurs de révision définis sémantiquement sont moins nombreux.

[Chou et Winslett, 1994] définit un algorithme de révision à l'aide d'une distance entre interprétations dans un fragment de L1O avec prédicat d'égalité. Cependant, des hypothèses fortes sont faites sur les interprétations considérées, en particulier, le domaine d'interprétation des instances est fixé et fini. Donc seul nombre fini d'interprétations est donc considéré, et ces interprétations sont finies.

[Qi et Du, 2009] propose un opérateur de révision de BC exprimées dans une logique de descriptions défini grâce à d'une « distance » entre interprétations partageant le même domaine. Cette « distance » est définie à l'aide du nombre de concepts nommés dont l'image change d'une interprétation à l'autre. Le calcul de cet opérateur de révision est réduit à un problème d'oubli de concepts [Wang *et al.*, 2008]. Cependant, d'après [Wang *et al.*, 2009], le résultat de cette opération ne peut pas toujours être exprimé dans certaines LD dont $\mathcal{ALC}$. C'est à dire que le postulat (A4′) n'est pas vérifié pour l'assignation fidèle engendré par la « distance » utilisée ici.

[Schwind, 2010] définit plusieurs opérateurs de fusion contrainte sur des réseaux de contraintes qualitatives. Certains de ces opérateurs sont définis sémantiquement, d'autres syntaxiquement. Comme nous le verrons en section 2.7, la fusion contrainte généralise la révision.

2.4 Un opérateur de révision dans un formalisme attributs-valeurs simples

Cette section reprend les travaux présentés dans [Cojan et Lieber, 2008b], [Cojan et Lieber, 2009b] et [Cojan et Lieber, 2008a].

2.4.1 Formalisme attributs-valeurs simples

Un formalisme attributs-valeurs simples permet de représenter un système caractérisé par les valeurs que prennent certains paramètres. On se donne alors un ensemble d'*attributs* $\mathtt{a}_1, \ldots, \mathtt{a}_n$ associés respectivement à des *domaines* $\mathcal{U}_1, \ldots, \mathcal{U}_n$. Les attributs correspondent aux différents paramètres et les domaines associés aux valeurs possibles de ces paramètres. Typiquement, les domaines $\mathcal{U}_k$ sont des ensembles de valeurs numériques ou des ensembles finis (d'où le terme valeurs simples).

Un formalisme attributs-valeurs simples sur les attributs $\mathtt{a}_1, \ldots, \mathtt{a}_n$ correspond à une logique dont le langage $\mathcal{L}$ est constitué de formules du type $\mathrm{P}(\mathtt{a}_1, \ldots, \mathtt{a}_n)$ où P est un prédicat représentant un sous-ensemble A_{P} de $\mathcal{U}$ déterminé par des contraintes sur les valeurs représentant les attributs. Une interprétation est une fonction I qui associe à chaque attribut $\mathtt{a}_k$ une valeur ${\mathtt{a}_k}^I \in \mathcal{U}_k$. On assimile une interprétation I aux n-uplets $({\mathtt{a}_1}^I, \ldots, {\mathtt{a}_n}^I) \in \Omega = \mathcal{U}_1 \times \ldots \times \mathcal{U}_n$. Pour une formule $\mathrm{P}(\mathtt{a}_1, \ldots, \mathtt{a}_n) \in \mathcal{L}$ et $I \in \Omega$:

$$I \models \mathrm{P}(\mathtt{a}_1, \ldots, \mathtt{a}_n) \quad \text{si} \quad I = ({\mathtt{a}_1}^I, \ldots, {\mathtt{a}_n}^I) \in A_{\mathrm{P}}$$

Ce formalisme étend la logique propositionnelle finie. Les variables propositionnelles $p_1, \ldots, p_n$ sont des attributs de domaines $\mathbb{B} = \{\mathtt{Vrai}, \mathtt{Faux}\}$. Les prédicats sont les combinaisons de connecteurs logiques appliquées aux variables propositionnelles.

Dans la remarque 2.2.2 (page 24), nous avons noté que la révision pouvait être mise sous la forme de problèmes d'optimisation sous contrainte. Les formalismes attributs-valeurs numériques (tous les $\mathcal{U}_k$ sont des ensembles de valeurs numériques) se prêtent bien à la résolution de problèmes d'optimisation. Nous nous intéressons dans la suite à un formalisme attributs-valeurs numériques pour lequel ces problèmes d'optimisation sont réductibles à des problèmes de programmation linéaire.

2.4.2 Réduction sous hypothèses à un problème d'optimisation linéaire

Rappel sur la programmation linéaire

Un problème de programmation linéaire consiste à déterminer les minima d'une forme linéaire sur un ensemble déterminé par une conjonction finie de contraintes linéaires. Plus précisément, soient $x_1, \ldots, x_n$, n variables, chaque variable x_k ayant pour domaine de définition un intervalle de $\mathbb{R}$ ou de $\mathbb{Z}$. Les contraintes sont du type $\alpha_1 x_1 + \ldots + \alpha_n x_n \leqslant \beta$, elles peuvent être représentées

sous forme matricielle :

$$A \times X \leqslant B, \quad \text{c'est-à-dire} \quad \begin{cases} \alpha_{11}x_1 & +\ldots+ & \alpha_{1n}x_n & \leqslant \beta_1 \\ \vdots & \vdots & \vdots & \vdots \\ \alpha_{p1}x_1 & +\ldots+ & \alpha_{pn}x_n & \leqslant \beta_p \end{cases}$$

avec $X = (x_1, \ldots, x_n)$, et la fonction objectif à minimiser est du type

$$C \times X = \gamma_1 x_1 + \ldots + \gamma_n x_n$$

Dans la littérature de la recherche opérationnelle (voir, p.ex. [Dantzig, 1963]), le terme programmation linéaire concerne les problèmes pour lesquels toutes les variables prennent des valeurs réelles, c'est-à-dire que leur domaine de définition est un intervalle de $\mathbb{R}$. Dans ce cadre il existe des algorithmes de résolution polynômiaux [Karmarkar, 1984] par rapport au nombre de variables et de contraintes linéaires. Dans le cas où certaines variables prennent des valeurs entières, on parle de programmation linéaire mixte et la résolution devient NP-difficile.

Représentation à l'aide de simplexes

Pour se placer dans le cadre de la programmation linéaire (mixte), $\mathcal{U}$ est supposé être de la forme : $\mathcal{U} = I_1 \times \ldots \times I_n$ où chaque I_k est un intervalle de $\mathbb{R}$ ou de $\mathbb{Z}$. On prend les notations de la programmation linéaire, les attributs sont appelés variables et sont notés x_i.

$\mathcal{L}$ est formé de l'ensemble des contraintes linéaires sur $\mathcal{U}$, une formule $(\sum_k \alpha_k x_k \leqslant \beta) \in \mathcal{L}$ détermine un sous-ensemble $\mathrm{Mod}\,(\sum_k \alpha_k x_k \leqslant \beta) \in 2^{\mathcal{U}}$ de $\mathcal{U}$. $\mathcal{P}$ est l'ensemble des parties de $\mathcal{U}$ déterminées par des conjonctions finies de contraintes linéaires. Un élément M de $\mathcal{P}$ est donc un *simplexe* et il est égal à l'intersection des sous-ensembles de $\mathcal{U}$ engendrés par ses contraintes linéaires.

On cherche maintenant à définir une assignation fidèle du type $\leqslant^d_\cdot$ telle que la minimisation d'un $\leqslant^d_\Psi$ puisse être réduite à un problème de minimisation linéaire. Si $\mathcal{U}$ contient au moins deux éléments, une « distance » sur $\mathcal{U}$ ne peut pas être linéaire, mais le problème de la minimisation de certaines « distances » se réduit à celui de la minimisation d'une fonction linéaire.

Exemple 2.4.1. *Soient $w_1, \ldots, w_n > 0$ des réels et d la « distance » définie sur $x, y \in \mathcal{U}$ par : $d(x,y) = \sum_k w_k |y_k - x_k|$. Pour $M, N \in \mathcal{P}$, la minimisation de d entre M et N est équivalente à celle de la forme linéaire sur $\mathcal{U}^3$: $(x,y,z) \mapsto \sum_k w_k z_k$ sous les contraintes : $x \in A$, $y \in B$ et pour tout i, $z_k \geqslant y_k - x_k$ et $z_k \geqslant x_k - y_k$.*

Exemple 2.4.2. *La technique utilisée dans l'exemple précédent pour réduire la minimisation de la « distance » d à celle d'une fonction linéaire s'applique plus généralement aux fonctions convexes définies à partir d'un nombre fini de formes linéaires. C'est le cas, par exemple pour $d : (x,y) \mapsto \max_k (w_k |y_k - x_k|)$.*

Pour les « distances » présentées ci-dessus, les éléments de $\mathcal{P}$ sont bien des fermés, ce qui garantit la satisfaction de (A2′) par $\leqslant^d_\cdot$. Les minima d'une fonction linéaire sur un simplexe forment encore un simplexe, ainsi les minima de $\leqslant^\Psi_d$ sur un élément de $\mathcal{P}$ forment un élément de $\mathcal{P}$ et $\leqslant^d_\cdot$ vérifie (A4′). $\leqslant^d_\cdot$ est bien une assignation fidèle au sens de la définition 2.3.1. De plus, comme un problème de programmation linéaire dont les contraintes sont satisfaisables et dont la fonction objectif est minorée sous ces contraintes admet une solution, pour tout $A \in \mathcal{P}$ non vide $\underset{\leqslant_\Psi}{\mathrm{Min}}\, A \neq \emptyset$, et donc $\leqslant^d_\cdot$ satisfait (A5′). Ainsi, d'après la proposition 2.3.1, avec les « distances » considérées plus haut, l'opérateur $\dot{+}$ engendré par $\leqslant^d_\cdot$ satisfait les postulats (G1) à (G6).

Exemple

Lors d'une épreuve de relais 2×100 m nage libre (épreuve non reconnue aux Jeux Olympiques mais pratique pour l'exemple), un spectateur essaie de relever le temps des nageurs des deux meilleures équipes. L'équipe 1 est constituée des nageurs a et b, l'équipe 2 est constituée des nageurs c et d, on note respectivement a, b, c et d les temps en secondes des nageurs. L'équipe 2 l'emporte. Avec ses moyens de mesure imprécis et non infaillibles, le spectateur relève que : $52 \leqslant a \leqslant 53$, $50 \leqslant b \leqslant 52$, $49 \leqslant c \leqslant 50$, $54 \leqslant d \leqslant 55$. Ainsi :

$$\psi = \left\{ \begin{array}{ccc} & a+b \geqslant c+d, & \\ 52 \leqslant a \leqslant 53, & & 49 \leqslant c \leqslant 50, \\ 50 \leqslant b \leqslant 52, & & 54 \leqslant d \leqslant 55 \end{array} \right\}$$

À la remise des médailles, l'équipe 2 reçoit bien l'or et l'équipe 1 l'argent. Le prix du nageur le plus rapide est remis au nageur b, pour un temps de 49 secondes. Ainsi, en relâchant les inégalités strictes en inégalités larges :

$$\mu = \left\{ \begin{array}{ccc} a+b \geqslant c+d, & & b = 49, \\ b \leqslant a, & b \leqslant c, & b \leqslant d \end{array} \right\}$$

ψ est incohérent avec μ : l'inégalité $50 \leqslant b$ ne peut pas être vérifiée, et les inégalités $49 \leqslant c$, $54 \leqslant d$ et $a \leqslant 53$ ne peuvent pas être toutes vérifiées à la fois, car on aurait alors $a + b \leqslant 102 < 103 \leqslant c + d$. Il faut donc revoir a à la hausse ($a = 53 + \delta_a$) et c et d à la baisse ($c = 49 - \delta_c$, $d = 54 - \delta_d$). En utilisant une « distance » du type de celle de l'exemple 2.4.1 avec les coefficients $w_k = 1$, le résultat de la révision correspond à l'ensemble des $(a, b, c, d) \in \mathbb{R}^4$ tels que :

$$\begin{cases} a = 53 + \delta_a, & \delta_a \geqslant 0, \\ b = 50 - \delta_b, & \delta_b \geqslant 0, \\ c = 49 - \delta_c, & \delta_c \geqslant 0, \\ d = 54 - \delta_d, & \delta_d \geqslant 0 \end{cases} \quad \text{avec} \quad \begin{cases} (a, b, c, d) \in \mathrm{Mod}(\mu) \\ \delta_a + \delta_b + \delta_c + \delta_d \ \text{ minimal} \end{cases}$$

$(a, b, c, d) \in \mathrm{Mod}(\mu)$ se traduit par :

$$\begin{cases} b = 49 & \text{soit } \delta_b = 1, \\ b \leqslant a & \text{soit } \delta_a \geqslant -4 \text{ (déjà impliqué par } \delta_a \geqslant 0), \\ b \leqslant c & \text{soit } \delta_c \leqslant 0 \text{ qui implique } \delta_c = 0 \text{ puisque } \delta_c \geqslant 0, \\ b \leqslant d & \text{soit } \delta_d \leqslant 5, \\ a + b \geqslant c + d & \text{soit } \delta_a + \delta_d \geqslant 1 \end{cases} \tag{2.9}$$

Puisque $\delta_b = 1$ et $\delta_c = 0$, minimiser $\delta_a + \delta_b + \delta_c + \delta_d$ est équivalent à minimiser $\delta_a + \delta_d$. Or $\delta_a + \delta_d \geqslant 1$, on ne peut donc pas faire mieux que $\delta_a + \delta_d = 1$. Réciproquement, tous les δ_a, δ_b, δ_c, δ_d tels que $\delta_a \geqslant 0$, $\delta_b = 1$, $\delta_c = 0$, $\delta_d \geqslant 0$ et $\delta_a + \delta_d = 1$ satisfont les contraintes (2.9). Après avoir appliqué ce résultat à a, b, c et d, en notant que $\delta_d = 1 - \delta_a$ équivaut à $a = d$, on obtient :

$$\psi \dot{+} \mu \equiv \left\{ \begin{array}{ll} 53 \leqslant a \leqslant 54, & b = 49, \\ c = 49, & d = a \end{array} \right\}$$

avec $d(\mathrm{Mod}(\psi), \mathrm{Mod}(\psi \dot{+} \mu)) = \delta_a + \delta_b + \delta_c + \delta_d = 2$.

Si le spectateur a plus confiance envers le score noté pour le nageur c qu'envers le score noté pour le nageur a (il supporte l'équipe 2), alors une « distance » avec des coefficients $w_.$ tels que $w_c > w_a$ est plus appropriée, et le résultat de la révision est :

$$\psi \dot{+} \mu \equiv \left\{ \begin{array}{ll} a = 54, & c = 49, \\ b = 49, & d = 54 \end{array} \right\}$$

Ce qui correspond à $\delta_a = 1$ au dessus, et à un coût de $1 + w_a$ (inférieur au coût $1 + \delta_a.w_a + (1 - \delta_a).w_c$ pour $\delta_a < 1$).

Remarque 2.4.1. *Le passage au problème dual est souvent utilisé en optimisation pour simplifier les calculs. Il serait intéressant d'étudier la signification du problème dual pour un problème de révision. Cette question est laissée en perspective.*

Travaux de [Schwind, 2010] sur la fusion dans les espaces affines

[Schwind, 2010] définit un opérateur de fusion contrainte, qui généralise la révision (voir section 2.7), de connaissances numériques représentées par des contraintes linéaires sur des espaces $\mathbb{R}^n$. Ces travaux ont aussi été publiés dans [Condotta *et al.*, 2010]. Le formalisme de représentation utilisé dans [Schwind, 2010] est plus général que celui que nous avons présenté dans cette section bien que quelques hypothèses supplémentaires soient faites pour assurer la satisfaction de certains postulats.

Alors que nous considérons des BC composées de contraintes linéaire, donc équivalentes à leur conjonction, [Schwind, 2010] considère le langage obtenu par clôture des contraintes linéaires par conjonction, négation et disjonction. Ce formalisme étend la logique propositionnelle, ce qui n'est pas le cas avec notre formalisme. Les ensembles représentés dans [Schwind, 2010] sont donc

des disjonctions finies de simplexes et leurs compléments. Or ces ensembles ne sont pas fermés, on retrouve donc le problème de la satisfaction du postulat (G3), en effet, la « distance » entre les modèles de BC n'est pas forcément atteinte. Pour palier à ce problème, les ensembles de modèles sont clos (pour la topologie) avant d'être fusionnés, comme proposé en section 2.3.3, et Mod(IC) est supposé être borné (Mod(M) pour la révision). La révision obtenue satisfait les postulats $(\overline{\text{G1}})$ à $(\overline{\text{G6}})$, le résultat équivalent doit être vrai pour la fusion contrainte.

[Schwind, 2010] ne propose pas de méthode de calcul exact de la fusion dans le formalisme proposé. Il propose une méthode de calcul approchée pour un opérateur de fusion défini grâce à une « distance » d définie pour $x, y \in \mathbb{R}^n$ par $d(x, y) = \sum_i w_i|y_i - x_i|$. Cette approximation s'appuie sur le fait que les simplexes peuvent être approchés par des unions finies de pavés qui sont des simplexes délimités par des contraintes linéaires du type $x_i \leqslant \beta_i$ ou $x_i \geqslant \beta_i$. Les minima de d entre des pavés sont calculable simplement en projetant sur les axes.

En perspective, nous pouvons considérer étendre notre réduction de la révision en un problème de programmation linéaire à la révision dans le formalisme plus général de [Schwind, 2010]. En effet, l'adhérence du complémentaire d'un simplexe est une union finie de simplexes : c'est le cas pour un simplexe délimité par une seule contrainte linéaire, le résultat pour un simplexe quelconque est obtenu par distributivité entre complémentaire et conjonction. Le calcul de la révision de connaissances représentées dans le formalisme de [Schwind, 2010] consiste à minimiser une « distance » entre disjonctions finies de simplexes. La propriété 1.2.3 (page 1.2.3) permet de réduire ce calcul à celui de la minimisation d'une « distance » entre simplexes, ce que l'on sait réduire à un programme de programmation linéaire pour certaines « distances » (par exemple les « distances » vues dans les exemples 2.4.1 et 2.4.2). L'extension au calcul de la fusion est directe pour un opérateur de fusion défini à l'aide de la fonction d'agrégation $\sum$, nous détaillons cette extension en section 2.7.3 (page 75). Il serait intéressant d'étudier cette réduction pour d'autres fonctions d'agrégation.

Outre l'intérêt pour le calcul de la fusion (et de la révision) de cette réduction, elle permettrait aussi de montrer que le résultat est lui-même exprimable dans ce formalisme. Comme nous l'avons vu plus haut (à propos de la satisfaction de (A4′)) les minima d'une « distance linéarisable » entre simplexes forment eux-mêmes un simplexe, la propriété 1.2.3 (page 1.2.3) montre alors que les minima d'une telle « distance » entre unions finies de simplexes forment aussi une union finie de simplexes. Elle permet aussi de s'affranchir de l'hypothèse que Mod(IC) est borné (resp. Mod(M) pour la révision).

2.5 Un opérateur de révision dans la logique de descriptions $\mathcal{ALC}$

Cette section reprend les travaux présentés dans [Cojan et Lieber, 2010c], [Cojan et Lieber, 2010b] et [Cojan et Lieber, 2011b].

L'étude d'opérateurs de révisions sur les logiques de descriptions est motivé par l'intérêt de celles-ci pour la représentation des connaissances. Elles offrent différents compromis entre expres-

sivité et complexité de raisonnement. Les logiques de descriptions forment une famille de logiques classiques équivalentes à des fragments décidables de la logique du premier ordre avec quelques extensions (prédicat d'égalité, domaines concrets) [Baader *et al.*, 2003] et pour lesquelles il existe des outils de raisonnement efficaces. Bien que la complexité théorique des inférences soit très élevée théoriquement (ExpTime-complet), les temps de calcul sont raisonnables en pratique. Les logiques de descriptions sont de plus en plus utilisées pour la représentation des connaissances, notamment en tant que base logique du standard OWL du web sémantique pour la représentation des ontologies. Elles étendent aussi les formalismes attributs-valeurs, fréquemment utilisés en raisonnement à partir de cas.

Nous nous intéressons ici plus particulièrement à la logique de descriptions $\mathcal{ALC}$ qui est la plus simple des logiques de descriptions expressives, c'est-à-dire des logiques de descriptions qui étendent la logique propositionnelle. Les autres logiques de descriptions expressives sont présentées dans [Baader *et al.*, 2003] comme des extensions de $\mathcal{ALC}$.

$\mathcal{ALC}$ est décrite en section 2.5.1. Pour notre algorithme de révision sur $\mathcal{ALC}$, nous nous appuyons sur un algorithme d'inférence sur les logiques de descriptions (et sur bien d'autres formalismes) : la méthode des tableaux sémantiques, présentée en section 2.5.2. Dans les logiques de descriptions, les BC sont décomposées en deux parties : la TBox qui décrit des connaissances générales et la ABox qui décrit des connaissances portant sur des individus particuliers. Nous présentons notre algorithme pour la révision d'ABox modulo une TBox en section 2.5.3, dont nous étudions les propriétés en section 2.5.4. Cet algorithme est dans un état préliminaire, en particulier, il ne satisfait pas tous les postulats de la révision. Plusieurs perspectives sont présentées en section 2.6 pour corriger cet algorithme mais aussi pour l'étendre à la révision de bases de connaissances complètes (ABox et TBox).

2.5.1 Préliminaires concernant la logique de descriptions $\mathcal{ALC}$

Syntaxe

Les élément du langage de représentation de $\mathcal{ALC}$ sont les concepts, les rôles, les instances et les formules.

Intuitivement, un *concept* représente un sous-ensemble du domaine d'interprétation. Un concept est soit un *concept atomique* (c.-à-d. un nom de concept), ou une expression conceptuelle de l'une des forme suivantes : $\top$, $\bot$, $\neg\mathtt{C}$, $\mathtt{C} \sqcap \mathtt{D}$, $\mathtt{C} \sqcup \mathtt{D}$, $\exists\mathtt{r}.\mathtt{C}$, et $\forall\mathtt{r}.\mathtt{C}$, où $\mathtt{C}$ et $\mathtt{D}$ sont des concepts (atomiques ou non) et $\mathtt{r}$ est un rôle. À un concept peut être associée une formule du premier ordre avec une variable libre x.

Par exemple, au concept

$$\mathtt{Champ} \sqcap \exists\mathtt{produit}.\mathtt{Mirabelle} \tag{2.10}$$

qui représente l'ensemble des champs qui produisent des mirabelles, peut être associée la formule

du premier ordre

$$\texttt{Champ}(\texttt{x}) \wedge (\exists \texttt{y}, \texttt{produit}(\texttt{x}, \texttt{y}) \wedge \texttt{Mirabelle}(\texttt{y}))$$

Intuitivement, un *rôle* représente une relation binaire. Dans $\mathcal{ALC}$ les rôles sont atomiques, c.-à-d. des noms de rôles. Leur pendant en logique du premier ordre sont les prédicats binaires. Le rôle apparaissant dans (2.10) est `produit`.

Intuitivement, une *instance* représente un élément du domaine d'interprétation. Dans $\mathcal{ALC}$ les instance sont atomiques, c.-à-d. des noms d'instances. Leur pendant en logique du premier ordre sont les constantes.

Il y a quatre types de formules dans $\mathcal{ALC}$ (suivies par leurs significations) :

(1) $\texttt{C} \sqsubseteq \texttt{D}$ (`C` est plus spécifique que `D`)

(2) $\texttt{C} \equiv \texttt{D}$ (`C` et `D` sont des concepts équivalents),

(3) $\texttt{C}(\texttt{a})$ (`a` est une instance de `C`), et

(4) $\texttt{r}(\texttt{a}, \texttt{b})$ (`r` relie `a` à `b`),

où `C` et `D` sont des concepts, `a` et `b` sont des instances, et `r` est un rôle.

Les formules de types (1) et (2) sont appelées des *formules terminologiques*. Les formules de types (3) et (4) sont appelées des *formules assertionnelles*, ou simplement des *assertions*.

Une BC Ψ en $\mathcal{ALC}$ est un ensemble fini de formules $\mathcal{ALC}$. La partie terminologique (ou *TBox* pour *terminological box*) de Ψ est l'ensemble de ses formules terminologiques. La partie assertionnelle (ou *ABox* pour *assertional box*) de Ψ est l'ensemble de ses formules assertionnelles.

On développera l'exemple suivant dans la suite de cette section.

Exemple 2.5.1. *Lors d'un voyage, je fais une pause près d'un champs d'arbres fruitiers que j'identifie comme étant des mirabelliers. Mais de retour à la voiture, je me rends compte que je ne suis plus en Lorraine. Or mon séjour à Nancy m'a appris non seulement que les mirabelles sont des prunes, mais qu'elle ne sont produites qu'en Lorraine*[2]*. Ces connaissances sont respectivement représentées par les ABox $\mathcal{A}_\psi$ et $\mathcal{A}_\mu$ et la TBox $\mathcal{T}$:*

$$\mathcal{A}_\psi = \{\exists \mathtt{produit}.\mathtt{Mirabelle}(\mathtt{champ})\}$$
$$\mathcal{A}_\mu = \{(\neg \mathtt{ChpLorrain})(\mathtt{champ})\}$$
$$\mathcal{T} = \left\{ \begin{array}{l} \mathtt{Mirabelle} \sqsubseteq \mathtt{Prune}, \\ \exists \mathtt{produit}.\mathtt{Mirabelle} \sqsubseteq \mathtt{ChpLorrain} \end{array} \right\}$$

Mes connaissances, avant d'apprendre que le champ n'est pas en Lorraine, sont représentées par la BC $\mathcal{T} \cup \mathcal{A}_\psi$. Ensuite, mes connaissances devraient être le résultat de la révision de $\mathcal{T} \cup \mathcal{A}_\psi$ par $\mathcal{T} \cup \mathcal{A}_\mu$.

2. Appellation non contrôlée mais défendue malgré tout.

Sémantique

Une interprétation est un couple $I = (\Delta_I, \cdot^I)$ où Δ_I est un ensemble non vide (le *domaine d'interprétation*) et où $\cdot^I$ associe à un concept $\mathtt{C}$ un sous-ensemble $\mathtt{C}^I$ de Δ_I, à un rôle $\mathtt{r}$ une relation binaire $\mathtt{r}^I$ sur Δ_I (pour $x, y \in \Delta_I$, x est lié à y par $\mathtt{r}^I$ est noté $(x, y) \in \mathtt{r}^I$) et, à une instance $\mathtt{a}$ un élément $\mathtt{a}^I$ de Δ_I.

Étant donné une interprétation I, les différents types d'expressions conceptuelles sont interprétées par :

$$\begin{array}{lll} \top^I = \Delta_I & (\mathtt{C} \sqcap \mathtt{D})^I = \mathtt{C}^I \cap \mathtt{D}^I & (\neg\mathtt{C})^I = \Delta_I \setminus \mathtt{C}^I \\ \bot^I = \emptyset & (\mathtt{C} \sqcup \mathtt{D})^I = \mathtt{C}^I \cup \mathtt{D}^I & \\ \multicolumn{3}{l}{(\exists \mathtt{r}.\mathtt{C})^I = \{x \in \Delta_I \mid \text{il existe } \ y \ \text{ tel que } \ (x,y) \in \mathtt{r}^I \text{ et } y \in \mathtt{C}^I\}} \\ \multicolumn{3}{l}{(\forall \mathtt{r}.\mathtt{C})^I = \{x \in \Delta_I \mid \text{pour tout } \ y, \quad \text{si} \quad (x,y) \in \mathtt{r}^I \text{ alors } y \in \mathtt{C}^I\}} \end{array}$$

Par exemple, si $\mathtt{Champ}^I$ est l'ensemble des champs et $\mathtt{Mirabelle}^I$ l'ensemble des mirabelles, et si $\mathtt{produit}^I$ est la relation « produit », alors le concept de l'équation (2.10) représente l'ensemble des champs où sont produites des mirabelles.

Étant donné une interprétation I :

$$\begin{array}{llll} I \models \mathtt{C} \sqsubseteq \mathtt{D} & \text{si } \mathtt{C}^I \subseteq \mathtt{D}^I & I \models \mathtt{C}(\mathtt{a}) & \text{si } \mathtt{a}^I \in \mathtt{C}^I \\ I \models \mathtt{C} \equiv \mathtt{D} & \text{si } \mathtt{C}^I = \mathtt{D}^I & I \models \mathtt{r}(\mathtt{a},\mathtt{b}) & \text{si } (\mathtt{a}^I, \mathtt{b}^I) \in \mathtt{r}^I \end{array}$$

Inférences

Soit Ψ une BC. Parmi les inférences classiques associées à $\mathcal{ALC}$ il y a les tests de la forme $\Psi \models f$, où f est une formule. Par exemple, tester si $\Psi \models \mathtt{C} \sqsubseteq \mathtt{D}$ est appelé le *test de subsomption* : il teste si, étant donné Ψ, le concept $\mathtt{C}$ est plus spécifique que le concept $\mathtt{D}$, ce qui est utile pour organiser les concepts en hiérarchies.

La *classification de concept* consiste, étant donné un concept $\mathtt{C}$, à trouver les concepts atomiques $\mathtt{A}$ apparaissant dans Ψ tels que $\Psi \models \mathtt{C} \sqsubseteq \mathtt{A}$ (les subsumants de $\mathtt{C}$) et les concepts atomiques $\mathtt{B}$ apparaissant dans Ψ tels que $\Psi \models \mathtt{B} \sqsubseteq \mathtt{C}$ (les subsumés de $\mathtt{C}$). La *classification d'instance* consiste, étant donné une instance $\mathtt{a}$, à trouver les concepts atomiques $\mathtt{A}$ apparaissant dans Ψ tels que $\Psi \models \mathtt{A}(\mathtt{a})$. Ces deux inférences peuvent être utilisées pour l'étape de remémoration d'un système de RÀPC.

La *satisfiabilité d'une ABox* consiste à tester si, étant donné une TBox $\mathcal{T}$, une ABox $\mathcal{A}$ a un modèle. Des inférences importantes peuvent se ramener à cette inférence, p. ex. $\mathcal{T} \models \mathtt{C} \sqsubseteq \mathtt{D}$ ssi $\mathcal{A} = \{(\mathtt{C} \sqcap \neg\mathtt{D})(\mathtt{a})\}$ est non satisfiable modulo $\mathcal{T}$, où $\mathtt{a}$ est une nouvelle instance (n'apparaissant ni dans $(\mathtt{C} \sqcap \neg\mathtt{D})$, ni dans $\mathcal{T}$).

D'autres inférences sont définies dans la littérature, mais ne sont pas utiles pour nos travaux.

Dans le cadre de l'exemple 2.5.1, nous devons déterminer si $\mathcal{T} \cup \mathcal{A}_\psi \cup \mathcal{A}_\mu$ est satisfaisable. Si oui, la révision de $\mathcal{T} \cup \mathcal{A}_\psi$ par $\mathcal{T} \cup \mathcal{A}_\mu$ ne demande pas de modifications, sinon il faudra effectuer

des corrections.

Remarque 2.5.1. *Pour exprimer le résultat de la révision de BC, nous aurons besoin de représenter des alternatives, par exemple, soit* $\mathtt{Mirabelle}(i)$*, soit* $\mathtt{produit}(\mathtt{champ}, i)$*, ce qui ne peut pas être exprimé dans une BC. Pour exprimer ces connaissances, nous nous servirons de disjonction de base de connaissances (*DBC*) aussi utilisées par [Meyer* et al.*, 2005] et [Qi* et al.*, 2006].*

Une DBC est un ensemble de BC $\mathcal{D} = \{\Psi_1, \ldots, \Psi_m\}$ interprétées disjonctivement :

$$\mathrm{Mod}(\mathcal{D}) = \bigcup_{i=1}^{m} \Psi_i$$

L'algorithme des tableaux présenté dans la section suivante manipule des disjonction d'ABox qui sont des DBC dont toutes les BC sont des ABox.

2.5.2 Une procédure de déduction classique pour $\mathcal{ALC}$: la méthode des tableaux

Soit Ψ une BC, $\mathcal{T}_0$, sa TBox, et $\mathcal{A}_0$, sa ABox. La procédure permet de tester si $\mathcal{A}_0$ est satisfiable, étant donnée $\mathcal{T}_0$.

Prétraitement. La première étape du prétraitement consiste à substituer $\mathcal{T}_0$ par une TBox équivalente $\mathcal{T}_0'$ de la forme $\{\top \sqsubseteq \mathtt{K}\}$, pour un certain concept $\mathtt{K}$. Cela peut être fait tout d'abord en substituant chaque formule $\mathtt{C} \equiv \mathtt{D}$ par deux formules $\mathtt{C} \sqsubseteq \mathtt{D}$ et $\mathtt{D} \sqsubseteq \mathtt{C}$. La TBox résultante est de la forme $\{\mathtt{C}_i \sqsubseteq \mathtt{D}_i\}_{1 \leqslant i \leqslant n}$ et on peut montrer qu'elle équivaut à $\{\top \sqsubseteq \mathtt{K}\}$, avec $\mathtt{K} = (\neg\mathtt{C}_1 \sqcup \mathtt{D}_1) \sqcap \ldots \sqcap (\neg\mathtt{C}_n \sqcup \mathtt{D}_n)$.

Exemple 2.5.2. *La TBox $\mathcal{T}$ de l'exemple 2.5.1 est remplacée par $\mathcal{T}' = \{\top \sqsubseteq K\}$ qui lui est équivalente où :*

$$K = (\neg\mathtt{Mirabelle} \sqcup \mathtt{Prune}) \sqcap ((\neg\exists\mathtt{produit.Mirabelle}) \sqcup \mathtt{ChpLorrain})$$

La deuxième étape du prétraitement consiste à mettre $\mathcal{T}_0$ et $\mathcal{A}_0$ sous forme normale négative (FNN), ce qui signifie que dans chaque concept apparaissant dans ces bases de connaissances, le signe de négation $\neg$ apparaît uniquement devant des concepts atomiques. Il est toujours possible d'effectuer une telle normalisation en appliquant, tant que c'est possible, les équivalences suivantes (de gauche à droite) :

$\neg\top \equiv \bot$	$\neg(\mathtt{C} \sqcap \mathtt{D}) \equiv \neg\mathtt{C} \sqcup \neg\mathtt{D}$	$\neg\exists\mathtt{r}.\mathtt{C} \equiv \forall\mathtt{r}.\neg\mathtt{C}$	$\neg\neg\mathtt{C} \equiv \mathtt{C}$
$\neg\bot \equiv \top$	$\neg(\mathtt{C} \sqcup \mathtt{D}) \equiv \neg\mathtt{C} \sqcap \neg\mathtt{D}$	$\neg\forall\mathtt{r}.\mathtt{C} \equiv \exists\mathtt{r}.\neg\mathtt{C}$	$\mathtt{C} \sqcup \bot \equiv \mathtt{C}$

Par exemple, le concept $\neg(\forall\mathtt{r}.(\neg\mathtt{A} \sqcup \exists\mathtt{s}.\mathtt{B}))$ est équivalent au concept $\exists\mathtt{r}.(\mathtt{A} \sqcap \forall\mathtt{s}.\neg\mathtt{B})$, qui est sous FNN.

Exemple 2.5.3. *Le concept K de l'exemple 2.5.2 est remplacé par le concept K′ sous FNN, qui lui est équivalent :*

$$K' = (\neg \mathit{Mirabelle} \sqcup \mathit{Prune}) \sqcap ((\forall \mathit{produit}.\neg \mathit{Mirabelle}) \sqcup \mathit{ChpLorrain})$$

Processus principal. Étant donné une TBox $\mathcal{T}_0 = \{\top \sqsubseteq \mathtt{K}\}$ et une ABox $\mathcal{A}_0$, toutes deux sous FNN, la méthode des tableaux manipule des disjonction d'ABox (introduites dans la remarque 2.5.1), en partant du singleton $\mathcal{D}_0 = \{\mathcal{A}_0^{\mathtt{K}}\}$, avec

$$\mathcal{A}_0^{\mathtt{K}} = \mathcal{A}_0 \cup \{\mathtt{K}(\mathtt{a}) \mid \mathtt{a} \text{ est une instance apparaissant dans } \mathcal{A}_0\} \tag{2.11}$$

Chacune des étapes suivantes de l'algorithme consiste à transformer la disjonction d'ABox courante $\mathcal{D}$ en une autre disjonction d'ABox $\mathcal{D}'$, en appliquant des règles de transformation sur les ABox : quand une règle de transformation ϱ, applicable sur une ABox $\mathcal{A} \in \mathcal{D}$, est sélectionnée par le processus, alors $\mathcal{D}' = (\mathcal{D} \setminus \{\mathcal{A}\}) \cup \{\mathcal{A}^1, \ldots, \mathcal{A}^p\}$ où les $\mathcal{A}^i$ sont obtenues par application de ϱ sur $\mathcal{A}$ (voir plus loin une description de ces règles).

Le processus termine quand aucune règle de transformation n'est applicable.

Une ABox est *fermée* quand elle contient un *conflit* (*clash* en anglais), c.-à-d., une contradiction « évidente ». Pour $\mathcal{ALC}$, les conflits sont les paires d'assertions de la forme $\{\mathtt{A}(\mathtt{a}), (\neg\mathtt{A})(\mathtt{a})\}$.

Par conséquent, une ABox fermée est insatisfiable. Une ABox *ouverte* est une ABox non fermée.

Une ABox est *complète* si aucune règle de transformation ne peut être appliquée sur elle.

Soit $\mathcal{D}_{\text{final}}$, la disjonction d'ABox à la fin du processus, c.-à-d., quand chacune des $\mathcal{A} \in \mathcal{D}$ est complète. Il a été montré (voir, p. ex., [Baader *et al.*, 2003]) que, avec les règles de transformation présentées ci-dessous, le processus termine toujours, et $\mathcal{A}_0$ est satisfiable étant donné $\mathcal{T}_0$ ssi $\mathcal{D}_{\text{final}}$ contient au moins une ABox ouverte.

Les règles de transformation. Il y a quatre règles de transformation pour la méthode des tableaux appliquée à $\mathcal{ALC}$: $\longrightarrow_{\sqcap}$, $\longrightarrow_{\sqcup}$, $\longrightarrow_{\forall}$ et $\longrightarrow_{\exists}^{\mathtt{K}}$. Aucune de ces règles n'est applicable sur une ABox fermée. L'ordre de ces règles affecte seulement la performance du système, à l'exception de la règle $\longrightarrow_{\exists}^{\mathtt{K}}$ qui doit être appliquée seulement quand aucune autre règle n'est applicable sur la disjonction d'ABox actuelle (ceci afin d'assurer la terminaison). Ces règles correspondent à des étapes de déduction au sens où elles ajoutent des assertions déduites d'assertions déjà existantes [3].

La règle $\longrightarrow_{\sqcap}$ est applicable sur une ABox $\mathcal{A}$ si cette dernière contient une assertion de la forme $(\mathtt{C}_1 \sqcap \ldots \sqcap \mathtt{C}_p)(\mathtt{a})$, et est telle qu'au moins une assertion $\mathtt{C}_k(\mathtt{a})$ $(1 \leqslant k \leqslant p)$ n'appartient

3. Plus précisément, chacune de ces règles transforme une disjonction d'ABox $\mathcal{D}$ en une autre disjonction d'ABox $\mathcal{D}'$, telle que, étant donné $\mathcal{T}_0$, $\mathcal{D}'$ est satisfiable ssi $\mathcal{D}$ l'est. Cette forme d'équivalence (l'équivalence des satisfiabilités) est celle que l'on retrouve dans la skolémisation des formules de logique du premier ordre (qui remplace les quantificateurs existentiels par des symboles de fonction).

pas à $\mathcal{A}$. Cette règle retourne l'ABox $\mathcal{A}'$ définie par

$$\mathcal{A}' = \mathcal{A} \cup \{\mathtt{C}_k(\mathtt{a}) \mid 1 \leqslant k \leqslant p\}$$

La règle $\longrightarrow_{\sqcup}$ est applicable sur une ABox $\mathcal{A}$ si cette dernière contient une assertion de la forme $(\mathtt{C}_1 \sqcup \ldots \sqcup \mathtt{C}_p)(\mathtt{a})$ mais aucune assertion de la forme $\mathtt{C}_k(\mathtt{a})$ $(1 \leqslant k \leqslant p)$. Cette règle retourne les ABox $\mathcal{A}^1, \ldots, \mathcal{A}^p$ définies, pour $1 \leqslant k \leqslant p$, par

$$\mathcal{A}^k = \mathcal{A} \cup \{\mathtt{C}_k(\mathtt{a})\}$$

La règle $\longrightarrow_{\forall}$ est applicable sur l'ABox $\mathcal{A}$ si cette dernière contient deux assertions de formes respectives $(\forall \mathtt{r}.\mathtt{C})(\mathtt{a})$ et $\mathtt{r}(\mathtt{a}, \mathtt{b})$ (avec le même $\mathtt{r}$ et le même $\mathtt{a}$) et si $\mathcal{A}$ ne contient pas l'assertion $\mathtt{C}(\mathtt{b})$. Cette règle retourne l'ABox $\mathcal{A}'$ définie par

$$\mathcal{A}' = \mathcal{A} \cup \{\mathtt{C}(\mathtt{b})\}$$

La règle $\longrightarrow_{\exists}^{\mathtt{K}}$ est applicable sur une ABox $\mathcal{A}$ si

(i) $\mathcal{A}$ contient une assertion de la forme $(\exists \mathtt{r}.\mathtt{C})(\mathtt{a})$;

(ii) $\mathcal{A}$ ne contient pas à la fois une assertion de la forme $\mathtt{r}(\mathtt{a}, \mathtt{b})$ et une assertion de la forme $\mathtt{C}(\mathtt{b})$ (avec le même $\mathtt{b}$, et avec les mêmes $\mathtt{C}$ et $\mathtt{a}$ que dans la condition précédente) ;

(iii) Il n'y a aucune instance $\mathtt{c}$ telle que $\{\mathtt{C} \mid \mathtt{C}(\mathtt{a}) \in \mathcal{A}\} \subseteq \{\mathtt{C} \mid \mathtt{C}(\mathtt{c}) \in \mathcal{A}\}$ (cette troisième condition est introduite pour assurer la terminaison de l'algorithme).

Si ces conditions sont remplies, soit $\mathtt{b}$ une nouvelle instance, cette règle retourne

$$\mathcal{A}' = \mathcal{A} \cup \{\mathtt{r}(\mathtt{a}, \mathtt{b}), \mathtt{C}(b)\} \cup \{\mathtt{K}(\mathtt{b})\}$$

On peut noter que la TBox $\mathcal{T}_0 = \{\top \sqsubseteq \mathtt{K}\}$ est utilisée ici : étant donné qu'une nouvelle instance $\mathtt{b}$ est introduite, cette instance doit satisfaire la TBox, ce qui correspond à l'assertion $\mathtt{K}(\mathtt{b})$.

Remarque 2.5.2. *Après l'application d'une de ces règles sur une ABox de $\mathcal{D}$, la disjonction d'ABox résultante est équivalente à $\mathcal{D}$ modulo $\mathcal{T}_0$ (voir note 3, page 44).*

Exemple 2.5.4. *La figure 2.1 représente le déroulement de l'algorithme de tableaux sur l'ABox $\mathcal{A}_0 = \mathcal{A}_\psi$ de l'exemple 2.5.1, modulo la TBox $\mathcal{T}_0 = \mathcal{T} = \{\top \sqsubseteq K\}$, c'est-à-dire sur la BC $\mathcal{T}_0 \cup \mathcal{A}_\psi$. L'arbre représente la disjonction d'ABox $\mathcal{D}$ générée par l'algorithme, chaque branche correspond à une ABox de $\mathcal{D}$, les formules précédant un embranchement étant communes aux ABox des différentes branches. Au départ, les seuls nœuds de cet arbre sont les formules de $\mathcal{A}_\psi^K$ (équation (2.11)). Puis, les règles de transformation sont appliquées. On peut noter que seule la règle $\longrightarrow_{\sqcup}$ conduit à la création de plusieurs nœuds-fils d'un nœud. Quand un conflit a été détecté sur une branche (p. ex. $\{\mathtt{Mirabelle}(i), (\neg\mathtt{Mirabelle})(i)\}$) la branche représente une ABox fermée (le conflit est symbolisé par $\square$). Pour des raisons de lisibilité, le développement de*

l'algorithme n'est pas fini. Le développement des formules issues de $K(\mathtt{champ})$ *et* $K(i)$ *génère de nouveaux embranchements dans la branche ouverte (branche de droite). Ces formules ne génèrent aucun conflit et on ne les développe pas pour réduire la taille de l'exemple. Une branche reste ouverte après application des règles,* $\mathcal{A}_\psi$ *est donc satisfiable modulo* $\mathcal{T}_0$*, c'est-à-dire que* $\mathcal{T}_0 \cup \mathcal{A}_\psi$ *est satisfiable, et qu'elle est équivalente, au renommage près de l'instance introduite* i*, à la ABox correspondant à la branche ouverte de la figure 2.1 :*

$$\mathcal{A} = \mathcal{A}_\psi^K \cup \left\{ \begin{array}{ll} \mathit{Prune}(i), & (\neg \mathit{Mirabelle} \sqcup \mathit{Prune})(i) \\ \mathit{ChpLorrain}(\mathit{champ}), & (\forall \mathit{produit}.\neg \mathit{Mirabelle} \sqcup \mathit{ChpLorrain})(\mathit{champ}) \end{array} \right\}$$

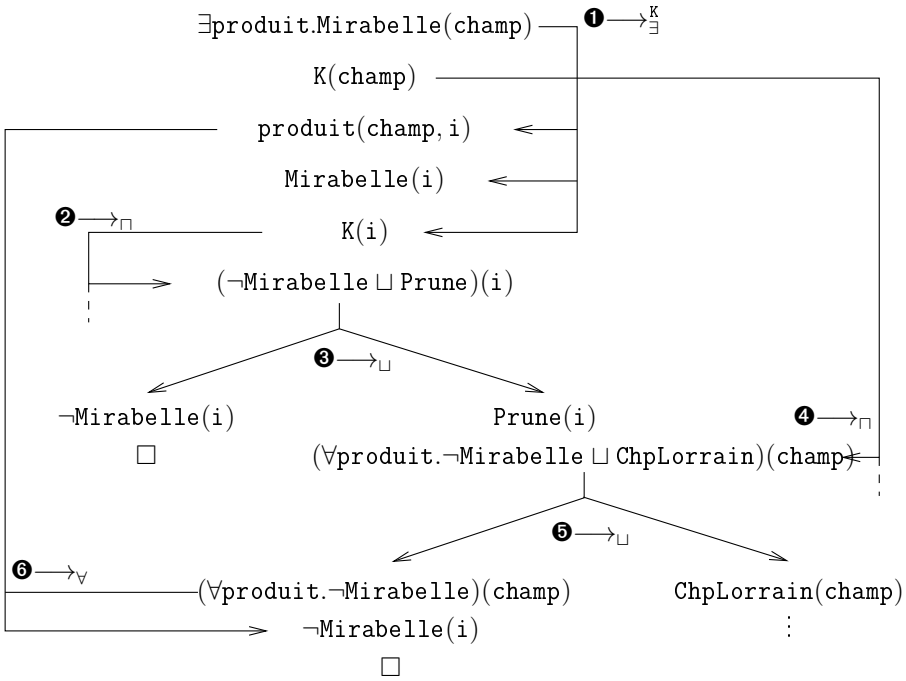


FIGURE 2.1 – Application de la méthode des tableaux montrant que l'ABox $\mathcal{A}_\psi$ est satisfiable modulo la TBox $\{\top \sqsubseteq \mathtt{K}\}$ (exemple 2.5.1). L'ordre d'application des règles (indiqué par les nombres sur fond noir) a été choisi de manière à rendre le résultat plus lisible, il ne respecte pas la contrainte « $\longrightarrow_\exists^K$ doit être appliquée en dernier ». Afin de garder une taille raisonnable, certaines formules ont été omises et la branche de droite n'a pas été totalement développée.

La figure 2.2 représente le déroulement de l'algorithme sur la ABox $\mathcal{A}_0 = \mathcal{A}_\psi \cup \mathcal{A}_\mu$*. Une grande partie des formules générées sont identiques à celles de la figure 2.1, cependant la formule de* $\mathcal{A}_\mu$ *ajoutée ici provoque un conflit dans la branche de droite. Toutes les branches sont donc fermées, ce qui permet de déduire que* $\mathcal{A}_\psi \cup \mathcal{A}_\mu$ *n'est pas satisfiable modulo* $\mathcal{T}_0$*.* $\mathcal{A}_\psi$ *devra donc être modifiée lors de sa révision par* $\mathcal{A}_\mu$*.*

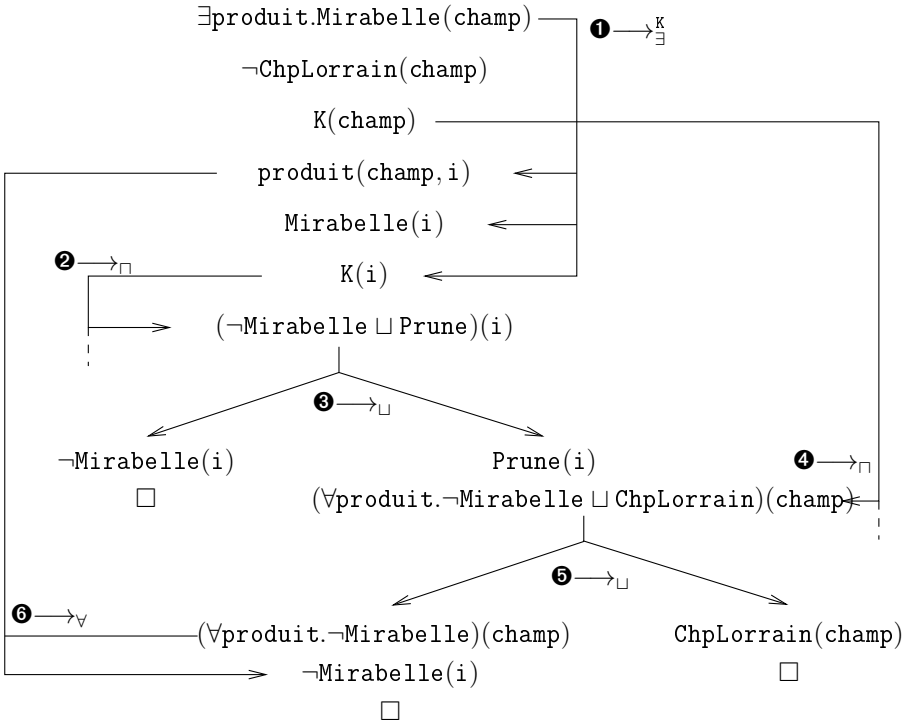


FIGURE 2.2 – Application de la méthode des tableaux montrant que l'ABox $\mathcal{A}_\psi \cup \mathcal{A}_\mu$ n'est pas satisfiable modulo la TBox $\{\top \sqsubseteq \mathtt{K}\}$ (exemple 2.5.1). Comme pour la figure 2.1, dans le but de rendre le résultat lisible, l'ordre d'application des règles choisi ne respecte pas toutes la contrainte « $\longrightarrow^{\mathtt{K}}_{\exists}$ doit être appliquée en dernier ».

2.5.3 Algorithme de révision sur des ABox (modulo une TBox)

Cette section présente un algorithme de révision entre deux bases de connaissances en $\mathcal{ALC}$ dont les TBox sont identiques, $\Psi = \mathcal{T} \cup \mathcal{A}_\psi$ et $\mathrm{M} = \mathcal{T} \cup \mathcal{A}_\mu$ où $\mathcal{A}_\psi$ et $\mathcal{A}_\mu$ sont des ABox et $\mathcal{T}$ est une TBox. Les assertions étant les mêmes dans les deux BC, elles ne sont pas révisées.

Paramètres et résultat de l'algorithme

L'algorithme prend les deux ABox $\mathcal{A}_\psi$ et $\mathcal{A}_\mu$ et la TBox $\mathcal{T} = \{\top \sqsubseteq \mathtt{K}\}$ en entrée. Une fonction $\mathtt{coût}$ associe à un littéral ℓ une valeur numérique $\mathtt{coût}(\ell) > 0$, où un littéral est soit un concept atomique (littéral positif) soit un concept de la forme $\neg\mathtt{A}$ où $\mathtt{A}$ est atomique (littéral négatif). Intuitivement, plus $\mathtt{coût}(\ell)$ est grand, plus il est difficile de renoncer à la vérité d'une assertion $\ell(\mathtt{a})$. Le résultat de l'algorithme est une disjonction d'ABox $\mathcal{D}$, représentant le résultat de la révision de $\mathcal{A}_\psi$ par $\mathcal{A}_\mu$ modulo $\mathcal{T}$, c'est-à-dire la révision de Ψ par M. En effet, lorsque deux alternatives pour la révision de $\mathcal{A}_\psi$ par $\mathcal{A}_\mu$ ne peuvent pas être départagées par $\mathtt{coût}$, le résultat ne peut pas toujours être exprimé sous forme d'une seule ABox.

Grandes lignes de l'algorithme de révision de $\mathcal{A}_\psi$ par $\mathcal{A}_\mu$ modulo $\mathcal{T}$

L'algorithme des tableaux est appliqué à $\mathcal{A}_\psi \cup \mathcal{A}_\mu$, cela permet de tester sa satisfiabilité modulo $\mathcal{T}$. Si $\mathcal{A}_\psi \cup \mathcal{A}_\mu$ est satisfiable, c'est-à-dire cohérent, alors d'après (R3), $(\mathcal{T} \cup \mathcal{A}_\psi) \dot{+} (\mathcal{T} \cup \mathcal{A}_\mu)$ équivaut à $\mathcal{T} \cup (\mathcal{A}_\psi \cup \mathcal{A}_\mu)$, il n'y a pas besoin de modifier $\mathcal{A}_\psi$. En revanche, si $\mathcal{T} \cup \mathcal{A}_\psi \cup \mathcal{A}_\mu$ n'est pas satisfiable, il faut modifier $\mathcal{A}_\psi$ afin de le rendre cohérent avec $\mathcal{A}_\mu$ modulo $\mathcal{T}$. Pour cela, on va « réparer » les conflits générés par l'application des règles à partir de $\mathcal{A}_\psi \cup \mathcal{A}_\mu$, sans modifier $\mathcal{A}_\mu$. Pour cette « réparation », il n'est pas suffisant de supprimer les conflits, les formules dont ces conflits dérivent doivent aussi être supprimées. Cela motive l'introduction dans la section 2.5.3 des AGraph qui étendent les ABox en conservant la trace de l'application des règles. De plus, pour obtenir une révision plus fine, $\mathcal{A}_\psi$ et $\mathcal{A}_\mu$ sont complétées par la méthode des tableaux avant d'être combinées.

Étapes de l'algorithme

$\mathcal{T}$, $\mathcal{A}_\psi$ et $\mathcal{A}_\mu$ sont au préalable mises sous FNN.

Appliquer la méthode des tableaux sur $\mathcal{A}_\psi$ et sur $\mathcal{A}_\mu$, en mémorisant l'application des règles de transformation. L'implantation de cette étape et des suivantes fait intervenir la notion de *graphe assertionnel* (ou *AGraph*) introduite ici. Un AGraph $\mathcal{G}$ est un graphe simple dont l'ensemble des nœuds, $\mathit{Nœuds}(\mathcal{G})$, est une ABox, et dont les arcs sont étiquetés par des règles de transformation : si $(\alpha, \beta) \in \mathit{Arcs}(\mathcal{G})$, l'ensemble des arcs orientés, alors $\lambda_{\mathcal{G}}(\alpha, \beta) = \varrho$ indique que β a été obtenu en appliquant ϱ sur α et, peut-être, sur d'autres assertions ($\lambda_{\mathcal{G}}$ est la fonction d'étiquetage du graphe $\mathcal{G}$).

La méthode des tableaux sur les AGraph repose sur les règles de transformation $\Longrightarrow_\sqcap$, $\Longrightarrow_\sqcup$, $\Longrightarrow_\forall$ et $\Longrightarrow_\exists^{\mathtt{K}}$. Elles sont similaires aux règles de transformation sur les ABox $\mathcal{ALC}$, à quelques différences près.

La règle $\Longrightarrow_\sqcap$ est applicable sur un AGraph $\mathcal{G}$ si

(i) $\mathcal{G}$ contient un nœud α de la forme $(\mathtt{C}_1 \sqcap \ldots \sqcap \mathtt{C}_p)(\mathtt{a})$;

(ii) $\mathcal{G} \neq \mathcal{G}'$ (c.-à-d., $\mathit{Nœuds}(\mathcal{G}) \neq \mathit{Nœuds}(\mathcal{G}')$ ou $\mathit{Arcs}(\mathcal{G}) \neq \mathit{Arcs}(\mathcal{G}')$) avec $\mathcal{G}'$ défini par

$$\begin{aligned}
\mathit{Nœuds}(\mathcal{G}') &= \mathit{Nœuds}(\mathcal{G}) \cup \{\mathtt{C}_k(\mathtt{a}) \mid 1 \leqslant k \leqslant p\} \\
\mathit{Arcs}(\mathcal{G}') &= \mathit{Arcs}(\mathcal{G}) \cup \{(\alpha, \mathtt{C}_k(\mathtt{a})) \mid 1 \leqslant k \leqslant p\} \\
\lambda_{\mathcal{G}'}(\alpha, \mathtt{C}_k(\mathtt{a})) &= \Longrightarrow_\sqcap \text{ pour } 1 \leqslant k \leqslant p \\
\lambda_{\mathcal{G}'}(e) &= \lambda_{\mathcal{G}}(e) \text{ pour } e \in \mathit{Arcs}(\mathcal{G})
\end{aligned}$$

Sous ces conditions, l'application de cette règle renvoie $\mathcal{G}'$.

La principale différence entre la règle $\longrightarrow_\sqcap$ sur une ABox et la règle $\Longrightarrow_\sqcap$ sur un AGraph est que cette dernière peut être appliquée à $\alpha = (\mathtt{C}_1 \sqcap \ldots \sqcap \mathtt{C}_p)(\mathtt{a})$ même quand $\mathtt{C}_k(\mathtt{a}) \in \mathit{Nœuds}(\mathcal{G})$ pour tout k, $1 \leqslant k \leqslant p$. Dans ce cas de figure, $\mathit{Nœuds}(\mathcal{G}') = \mathit{Nœuds}(\mathcal{G})$ mais $\mathit{Arcs}(\mathcal{G}') \neq \mathit{Arcs}(\mathcal{G})$:

un nouvel arc $(\alpha, \mathtt{C}_k)$ indique ici que $\alpha \models \mathtt{C}_k(\mathtt{a})$ et donc, que si $\mathtt{C}_k(\mathtt{a})$ doit être supprimé, alors α doit aussi être supprimé (voir plus loin, l'étape de réparation de l'algorithme).

De même, les règles $\longrightarrow_{\sqcup}$, $\longrightarrow_{\forall}$, et $\longrightarrow_{\exists}^{\mathtt{K}}$ sont modifiées en $\Longrightarrow_{\sqcup}$, $\Longrightarrow_{\forall}$, et $\Longrightarrow_{\exists}^{\mathtt{K}}$, détaillées dans la figure 2.3.

On peut appliquer la méthode des tableaux présentée en section 2.5.2 sur une ABox $\mathcal{A}_0$, avec comme seule différence que ce sont des AGraph qui sont manipulés à la place d'ABox, ce qui entraîne que :

(1) un AGraph initial $\mathcal{G}_0$ doit être construit à partir de $\mathcal{A}_0$ (il est défini par $\mathit{Nœuds}(\mathcal{G}_0) = \mathcal{A}_0$ et $\mathit{Arcs}(\mathcal{G}_0) = \emptyset$) ;
(2) les règles $\Longrightarrow.$ sont utilisées à la place des règles $\longrightarrow.$;
(3) le résultat est un ensemble d'AGraph complets et ouverts (qui est vide ssi $\mathcal{G}_0$ n'est pas satisfiable).

Remarque 2.5.3. *Les règles $\Longrightarrow.$ ont été définies afin de généraliser l'algorithme des tableaux sur ABox (avec les règles $\longrightarrow.$) par un algorithme des tableaux sur AGraph. En effet, les règles $\Longrightarrow.$ génèrent les mêmes nœuds que les règles $\longrightarrow.$, et établissent en plus les arêtes donnant l'origine de ces nœuds. Plus précisément, étant donné une ABox $\mathcal{A}_0$ et l'AGraph $\mathcal{G}_0$ tel que $\mathtt{Nœuds}(\mathcal{G}_0) = \mathcal{A}_0$ et $\mathtt{Arcs}(\mathcal{G}_0) = \emptyset$, soient $\{\mathcal{A}_k\}_k$ la disjonction d'ABox obtenue par l'application des règles $\longrightarrow.$ sur $\mathcal{A}_0$ et $\{\mathcal{G}_\ell\}_\ell$ la disjonction d'AGraph obtenue par l'application des règles $\Longrightarrow.$ dans le même ordre sur $\mathcal{G}_0$, on a alors $\{\mathcal{A}_k\}_k = \{\mathcal{B}_\ell\}_\ell$ au renommage près des instances introduites, où $\mathcal{B}_\ell$ est l'ensemble des nœuds de $\mathcal{G}_\ell$.*

Soient $\{\mathcal{G}_i\}_{1 \leqslant i \leqslant m}$ et $\{\mathcal{H}_j\}_{1 \leqslant j \leqslant n}$ les ensembles d'AGraph ouverts et complets obtenus par l'application de la méthode des tableaux respectivement sur $\mathcal{A}_0 = \mathcal{A}_\psi$ et $\mathcal{A}_0 = \mathcal{A}_\mu$. Si, $\mathcal{A}_\psi$ et $\mathcal{A}_\mu$ sont satisfiables, alors $m \neq 0$ et $n \neq 0$. Si $m = 0$ ou $n = 0$, l'algorithme s'arrête avec la valeur $\mathcal{D} = \{\mathcal{A}_\mu\}$.

Exemple 2.5.5. *Dans l'exemple 2.5.1 (page 41) , l'algorithme des tableaux génère plusieurs $\mathcal{G}_i$, qui contiennent le AGraph correspondant à la branche ouverte de la figure 2.1 et les nœuds supplémentaires n'interviennent pas dans les conflits. Pour simplifier, on considérera qu'il n'y a qu'un seul $\mathcal{G}_i$ (figure 2.4). Les seules règles applicables sur l'ABox $\mathcal{A}_\mu$ développent la formule $\mathtt{K}(\mathtt{léo})$, elles n'ont pas d'intérêt dans l'exemple, on simplifie en disant qu'il n'y a qu'un seul AGraph $\mathcal{H}_j$: $\mathcal{A}_\mu^{\mathtt{K}}$.*

Générer des conflits explicites à partir de $\mathcal{G}_i$ et $\mathcal{H}_j$. On considère un nouveau type d'assertion, réifiant la notion de conflit : l'*assertion de conflit* $\Box\pm\mathtt{A}(\mathtt{a})$ réifie le conflit $\{\mathtt{A}(\mathtt{a}), (\neg\mathtt{A})(\mathtt{a})\}$. Elles sont générées par la règle $\Longrightarrow_{\Box}$. Cette règle est applicable sur l'AGraph $\mathcal{G}$ si

(i) $\mathcal{G}$ contient deux nœuds $\mathtt{A}(\mathtt{a})$ et $(\neg\mathtt{A})(\mathtt{a})$ (avec le même $\mathtt{A}$ et le même $\mathtt{a}$) ;

Une condition nécessaire pour appliquer $\Longrightarrow_{\sqcup}$ sur un AGraph $\mathcal{G}$ est que $\mathcal{G}$ contienne un nœud α de la forme $(\mathtt{C}_1 \sqcup \ldots \sqcup \mathtt{C}_p)(\mathtt{a})$. Si c'est le cas, alors deux conditions peuvent être considérées :

(a) $\mathcal{G}$ ne contient aucune assertion $\mathtt{C}_k(\mathtt{a})$ $(1 \leqslant k \leqslant p)$. Sous cette condition, la règle renvoie les AGraph $\mathcal{G}^1, \ldots, \mathcal{G}^p$ définis, pour $1 \leqslant k \leqslant p$, par

$$\begin{aligned}
\mathit{Nœuds}(\mathcal{G}^k) &= \mathit{Nœuds}(\mathcal{G}) \cup \{\mathtt{C}_k(\mathtt{a})\} \\
\mathit{Arcs}(\mathcal{G}^k) &= \mathit{Arcs}(\mathcal{G}) \cup \{(\alpha, \mathtt{C}_k(\mathtt{a}))\} \\
\lambda_{\mathcal{G}^k}(\alpha, \mathtt{C}_k(\mathtt{a})) &= \Longrightarrow_{\sqcup} \\
\lambda_{\mathcal{G}^k}(e) &= \lambda_{\mathcal{G}}(e) \text{ for } e \in \mathit{Arcs}(\mathcal{G})
\end{aligned}$$

(b) $\mathcal{G}$ contient une ou plusieurs assertion $\beta_k = \mathtt{C}_k(\mathtt{a})$ telle que $(\alpha, \beta_k) \notin \mathit{Arcs}(\mathcal{G})$. Sous cette condition, $\Longrightarrow_{\sqcup}$ renvoie l'AGraph $\mathcal{G}'$ obtenu en ajoutant cet arc (α, β_k) à $\mathcal{G}$, avec $\lambda_{\mathcal{G}'}(\alpha, \beta_k) = \Longrightarrow_{\sqcup}$.

La règle $\Longrightarrow_{\forall}$ est applicable sur un AGraph $\mathcal{G}$ si

(i) $\mathcal{G}$ contient un nœud α_1 de la forme $(\forall \mathtt{r}.\mathtt{C})(\mathtt{a})$ et un nœud α_2 de la forme $\mathtt{r}(\mathtt{a}, \mathtt{b})$;

(ii) $\mathcal{G} \neq \mathcal{G}'$ avec $\mathcal{G}'$ défini par

$$\begin{aligned}
\mathit{Nœuds}(\mathcal{G}') &= \mathit{Nœuds}(\mathcal{G}) \cup \{\mathtt{C}(\mathtt{b})\} \\
\mathit{Arcs}(\mathcal{G}') &= \mathit{Arcs}(\mathcal{G}) \cup \{(\alpha_1, \mathtt{C}(\mathtt{b})), (\alpha_2, \mathtt{C}(\mathtt{b}))\} \\
\lambda_{\mathcal{G}'}(\alpha_1, \mathtt{C}(\mathtt{b})) &= \lambda_{\mathcal{G}'}(\alpha_2, \mathtt{C}(\mathtt{b})) = \Longrightarrow_{\forall} \\
\lambda_{\mathcal{G}'}(e) &= \lambda_{\mathcal{G}}(e) \text{ pour } e \in \mathit{Arcs}(\mathcal{G})
\end{aligned}$$

Sous ces conditions, la règle renvoie $\mathcal{G}'$.

La règle $\Longrightarrow_{\exists}^{\mathtt{K}}$ est applicable sur un AGraph $\mathcal{G}$ si

(i) $\mathcal{G}$ contient un nœud α de la forme $(\exists \mathtt{r}.\mathtt{C})(\mathtt{a})$;

(ii) (a) Soit $\mathcal{G}$ ne contient pas à la fois $\mathtt{r}(\mathtt{a}, \mathtt{b})$ et $\mathtt{C}(\mathtt{b})$ pour aucune instance $\mathtt{b}$;

(b) Soit $\mathcal{G}$ contient deux assertions $\beta_1 = \mathtt{r}(\mathtt{a}, \mathtt{b})$ et $\beta_2 = \mathtt{C}(\mathtt{b})$, telles que $(\alpha, \beta_1) \notin \mathit{Arcs}(\mathcal{G})$ ou $(\alpha, \beta_2) \notin \mathit{Arcs}(\mathcal{G})$;

(iii) Il n'y a pas d'instance $\mathtt{c}$ telle que $\{\mathtt{C} \mid \mathtt{C}(\mathtt{a}) \in \mathit{Nœuds}(\mathcal{G})\} \subseteq \{\mathtt{C} \mid \mathtt{C}(\mathtt{c}) \in \mathit{Nœuds}(\mathcal{G})\}$ (condition du *set-blocking*, servant à assurer la terminaison de l'algorithme).

Si la condition (ii-a) est satisfaite, soit $\mathtt{b}$ une nouvelle instance. La règle renvoie $\mathcal{G}'$ défini par

$$\begin{aligned}
\mathit{Nœuds}(\mathcal{G}') &= \mathit{Nœuds}(\mathcal{G}) \cup \{\mathtt{r}(\mathtt{a}, \mathtt{b}), \mathtt{C}(\mathtt{b}), \mathtt{K}(\mathtt{b})\} \\
\mathit{Arcs}(\mathcal{G}') &= \mathit{Arcs}(\mathcal{G}) \cup \{(\alpha, \mathtt{r}(\mathtt{a}, \mathtt{b})), (\alpha, \mathtt{C}(\mathtt{b}))\} \\
\lambda_{\mathcal{G}'}(\alpha, \mathtt{r}(\mathtt{a}, \mathtt{b})) &= \lambda_{\mathcal{G}'}(\alpha, \mathtt{C}(\mathtt{b})) = \Longrightarrow_{\exists}^{\mathtt{K}} \\
\lambda_{\mathcal{G}'}(e) &= \lambda_{\mathcal{G}}(e) \text{ pour } e \in \mathit{Arcs}(\mathcal{G})
\end{aligned}$$

Sous la condition (ii-b), la règle renvoie $\mathcal{G}'$ défini par

$$\begin{aligned}
\mathit{Nœuds}(\mathcal{G}') &= \mathit{Nœuds}(\mathcal{G}) \\
\mathit{Arcs}(\mathcal{G}') &= \mathit{Arcs}(\mathcal{G}) \cup \{(\alpha, \beta_1), (\alpha, \beta_2)\} \\
\lambda_{\mathcal{G}'}(\alpha, \beta_1) &= \lambda_{\mathcal{G}'}(\alpha, \beta_2) = \Longrightarrow_{\exists}^{\mathtt{K}} \\
\lambda_{\mathcal{G}'}(e) &= \lambda_{\mathcal{G}}(e) \text{ pour } e \in \mathit{Arcs}(\mathcal{G})
\end{aligned}$$

FIGURE 2.3 – Les règles de transformation $\Longrightarrow_{\sqcup}$, $\Longrightarrow_{\forall}$, et $\Longrightarrow_{\exists}^{\mathtt{K}}$.

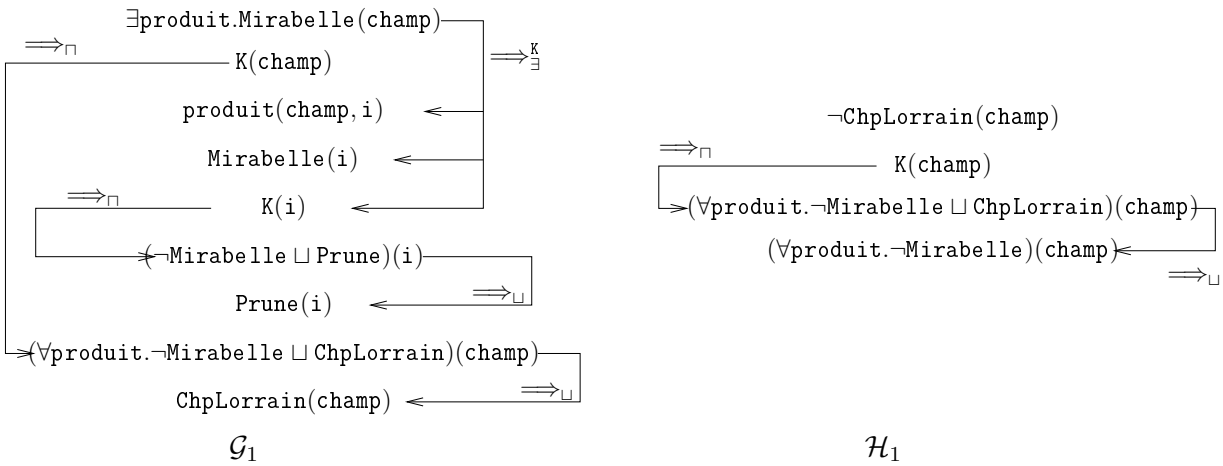


FIGURE 2.4 – AGraph $\mathcal{G}_1$ et $\mathcal{H}_1$ générés respectivement à partir de $\mathcal{A}_\psi$ et $\mathcal{A}_\mu$.

(ii) $\mathcal{G} \neq \mathcal{G}'$ avec $\mathcal{G}'$ défini par

$$\begin{aligned}
\mathit{Nœuds}(\mathcal{G}') &= \mathit{Nœuds}(\mathcal{G}) \cup \{\Box\pm\mathtt{A}(\mathtt{a})\} \\
\mathit{Arcs}(\mathcal{G}') &= \mathit{Arcs}(\mathcal{G}) \cup \{(\mathtt{A}(\mathtt{a}), \Box\pm\mathtt{A}(\mathtt{a})), ((\neg\mathtt{A})(\mathtt{a}), \Box\pm\mathtt{A}(\mathtt{a}))\} \\
\lambda_{\mathcal{G}'}(\mathtt{A}(\mathtt{a}), \Box\pm\mathtt{A}(\mathtt{a})) &= \lambda_{\mathcal{G}'}((\neg\mathtt{A})(\mathtt{a}), \Box\pm\mathtt{A}(\mathtt{a})) = \Longrightarrow_{\Box} \\
\lambda_{\mathcal{G}'}(e) &= \lambda_{\mathcal{G}}(e) \text{ pour } e \in \mathit{Arcs}(\mathcal{G})
\end{aligned}$$

Sous ces conditions, la règle renvoie $\mathcal{G}'$.

L'étape suivante de l'algorithme consiste à appliquer la méthode des tableaux sur chaque $\mathcal{G}_i \cup \mathcal{H}_j$, pour tout i et j, $1 \leqslant i \leqslant m$, $1 \leqslant j \leqslant n$, en utilisant les règles de transformation $\Longrightarrow_{\sqcap}$, $\Longrightarrow_{\sqcup}$, $\Longrightarrow_{\forall}$, $\Longrightarrow_{\exists}^{\mathtt{K}}$ et $\Longrightarrow_{\Box}$. À la différence de la méthode des tableaux présentée plus haut où il était inutile d'appliquer les règles sur les ABox fermées (ou les AGraph fermés), ici, lorsqu'une règle est applicable sur un AGraph contenant une assertion de conflit, elle est appliquée. Plusieurs conflits peuvent donc être générés dans le même AGraph.

Remarque 2.5.4. *Si une assertion de conflit $\Box\pm\mathtt{A}(\mathtt{a})$ est générée, alors ce conflit est la conséquence d'assertions venant à la fois de $\mathcal{G}_i$ et de $\mathcal{H}_j$, autrement, ce conflit aurait été généré à l'étape précédente de l'algorithme (puisque ces deux AGraph sont ouverts et complets).*

Exemple 2.5.6. *La figure 2.5 représente le AGraph obtenu en appliquant la méthode des tableaux sur $\mathcal{G}_1 \cup \mathcal{H}_1$ où $\mathcal{G}_1$ et $\mathcal{H}_1$ sont les AGraph obtenus dans l'exemple 2.5.5. Un conflit apparaît dans l'unique branche obtenue, il concerne les assertions $\mathtt{ChpLorrain}(\mathtt{champ})$ et $\neg\mathtt{ChpLorrain}(\mathtt{champ})$.*

Réparer les assertions de conflit. L'étape précédente a produit un ensemble non vide d'AGraph $\{\mathcal{Q}_k\}_{1 \leqslant k \leqslant p}$. L'étape de réparation consiste à réparer chacun de ces AGraph $\mathcal{Q}_k$ et à

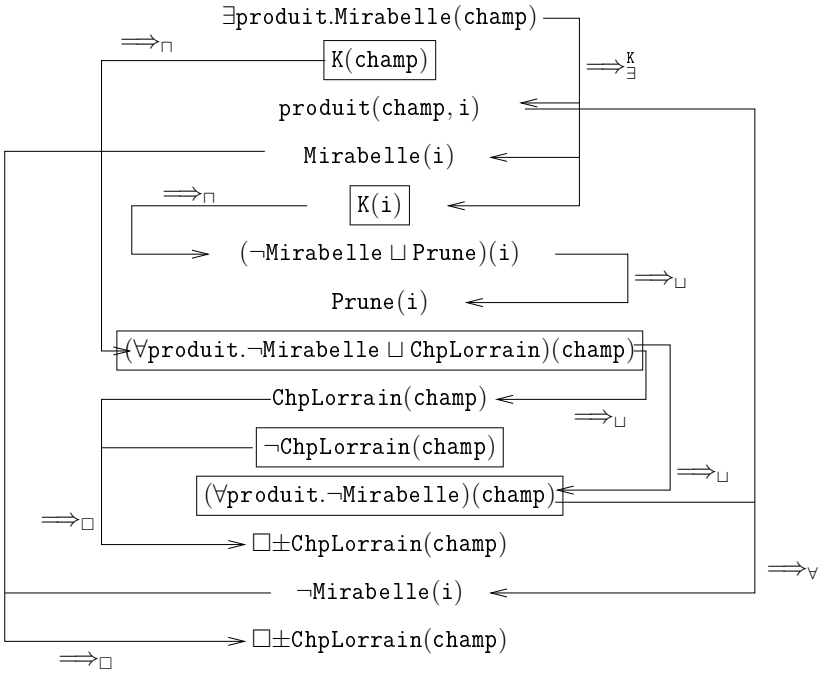


FIGURE 2.5 – AGraph obtenu après application des règles $\Longrightarrow.$, y compris $\Longrightarrow_{\Box}$, sur l'union des AGraph $\mathcal{G}_1$ et $\mathcal{H}_1$ obtenus dans l'exemple 2.5.5. Les formules impliquées par $\mathcal{A}_\mu$ et $\mathcal{T}$ sont encadrées pour symboliser le fait qu'elles ne seront pas remises en cause lors de la révision.

ne conserver que ceux qui minimisent le coût de réparation[4]. Soit $\mathcal{Q}_k$ un AGraph obtenu en développant l'algorithme des tableaux sur $\mathcal{G}_i \cup \mathcal{H}_j$. Si $\mathcal{Q}_k$ ne contient aucune assertion de conflit, cela implique que $\mathcal{G}_i \cup \mathcal{H}_j$ est satisfiable et donc que $\mathcal{A}_\psi \cup \mathcal{A}_\mu$ l'est aussi : il n'y a pas d'adaptation nécessaire. Si $\mathcal{Q}_k$ contient $\delta \geqslant 1$ assertions de conflit, alors l'une d'entre elles est choisie et sa réparation produit un ensemble d'AGraph réparés $\mathcal{Q}'_k$ contenant $\delta - 1$ conflits. La réparation est ensuite reprise sur $\mathcal{Q}'_k$, jusqu'à ce qu'il n'y ait plus de conflit. Le coût de la réparation globale est la somme des coûts de chaque réparation. Dans la suite, nous verrons comment est réparé un conflit de $\mathcal{Q}_k$.

Le principe de la réparation d'un conflit consiste à supprimer des assertions de $\mathcal{Q}_k$ de sorte à empêcher que les conflit soient générés à nouveau par l'application des règles. Ainsi, la réparation de tous les conflits doit produire des AGraph satisfiables (ce qui est une conséquence de la complétude de l'algorithme des tableaux sur $\mathcal{ALC}$). Pour cela, on suit le principe suivant, exprimé comme une règle d'inférence :

$$\frac{\varphi \models \beta \qquad \beta \text{ doit être supprimé}}{\varphi \text{ doit être supprimé}} \tag{2.12}$$

où β est une assertion et φ est un ensemble minimal d'assertions tel que $\varphi \models \beta$ (φ doit être

4. Dans notre prototype, cela a été amélioré en éliminant les réparations en cours dont le coût dépasse le minimum en cours.

interprété comme la conjonction de ses formules). Supprimer φ revient à oublier une des assertions $\alpha \in \varphi$: lorsque $\mathrm{card}(\varphi) \geqslant 2$, il y a plusieurs manières de supprimer φ, et donc, plusieurs AGraph $\mathcal{Q}'_k$ peuvent être obtenus à partir de $\mathcal{Q}_k$. La relation $\models$ reliant φ et β est représentée par les arcs de $\mathcal{Q}_k$. Donc, suivant (2.12), la suppression d'assertions est propagée suivant les arcs (α, β), de β vers α.

Soit $\beta = \Box\pm\mathtt{A}(\mathtt{a})$, le conflit de $\mathcal{Q}_k$ qu'il faut supprimer. Soient $\alpha^+ = \mathtt{A}(\mathtt{a})$ et $\alpha^- = \neg\mathtt{A}(\mathtt{a})$. α^+ ou α^-, l'un des deux au moins, doit être supprimé. $\mathcal{H}_j$ étant un AGraph ouvert et complet, soit $\alpha^+ \notin \mathcal{H}_j$, soit $\alpha^- \notin \mathcal{H}_j$ (cf. remarque 2.5.4). Trois situations sont possibles :

- Si $\alpha^+ \in \mathcal{H}_j$, alors α^+ ne peut être supprimé : c'est une assertion générée à partir de $\mathcal{A}_\mu$. Donc α^- doit être supprimée.
- Si $\alpha^- \in \mathcal{H}_j$, alors α^+ doit être supprimé.
- Si $\alpha^+ \notin \mathcal{H}_j$ et $\alpha^- \notin \mathcal{H}_j$, le choix de l'assertion à supprimer repose sur la minimisation du coût. Si $\mathtt{coût}(\mathtt{A}) < \mathtt{coût}(\neg\mathtt{A})$, α^+ doit être supprimée. Si $\mathtt{coût}(\mathtt{A}) > \mathtt{coût}(\neg\mathtt{A})$, α^- doit être supprimée. Si $\mathtt{coût}(\mathtt{A}) = \mathtt{coût}(\neg\mathtt{A})$, deux AGraph sont générés : l'un en supprimant α^+, l'autre en supprimant α^-.

Si une assertion β doit être supprimée, la propagation de la suppression suivant un arc (α, β) tel que $\lambda_{\mathcal{G}}(\alpha, \beta) \in \{\Longrightarrow_\sqcap, \Longrightarrow_\sqcup, \Longrightarrow_\exists^{\mathtt{K}}\}$ consiste à supprimer α (et à propager la suppression depuis α).

Soit β une assertion à supprimer qui a été déduite par la règle $\Longrightarrow_\forall$. Il existe alors deux assertions telles que $\lambda_{\mathcal{G}}(\alpha_1, \beta) = \lambda_{\mathcal{G}'}(\alpha_2, \beta) = \Longrightarrow_\forall$. Dans cette situation, deux AGraph sont générés, l'un fondé sur la suppression de α_1, l'autre sur la suppression de α_2 (lorsque α_1 ou α_2 est dans $\mathcal{H}_j$, un seul AGraph est généré).

Remarque 2.5.5. *La condition (iii) de la règle $\Longrightarrow_\exists^{\mathtt{K}}$ (règle du* set-blocking*) peut « cacher » des conflits. Cette condition bloque l'application de la règle $\Longrightarrow_\exists^{\mathtt{K}}$ sur un nœuds $(\exists\ \mathtt{r}\ .\mathcal{C})(\mathtt{a})$ d'un AGraph $\mathcal{G}$ lorsqu'il existe une instance $\mathtt{c}$ telle que $\{\mathcal{C} \mid \mathcal{C}(\mathtt{a}) \in \mathtt{Nœuds}(\mathcal{G})\} \subseteq \{\mathcal{C} \mid \mathcal{C}(\mathtt{c}) \in \mathtt{Nœuds}(\mathcal{G})\}$. Cette règle se justifie par le fait que si un conflit découle de $\{\mathcal{C}(\mathtt{a}) \mid \mathcal{C}(\mathtt{a}) \in \mathtt{Nœuds}(\mathcal{G})\}$, alors le même conflit découle des nœuds correspondant dans $\{\mathcal{C}(\mathtt{c}) \mid \mathcal{C}(\mathtt{c}) \in \mathtt{Nœuds}(\mathcal{G})\}$, il est donc suffisant de vérifier la cohérence de $\{\mathcal{C}(\mathtt{c}) \mid \mathcal{C}(\mathtt{c}) \in \mathtt{Nœuds}(\mathcal{G})\}$. Si un conflit est généré par $\{\mathcal{C}(\mathtt{c}) \mid \mathcal{C}(\mathtt{c}) \in \mathtt{Nœuds}(\mathcal{G})\}$ et que l'ensemble des formules de $\{\mathcal{C}(\mathtt{c}) \mid \mathcal{C}(\mathtt{c}) \in \mathtt{Nœuds}(\mathcal{G})\}$ impliqués dans le conflit correspondent à des formules de $\{\mathcal{C}(\mathtt{a}) \mid \mathcal{C}(\mathtt{a}) \in \mathtt{Nœuds}(\mathcal{G})\}$, alors des corrections sont nécessaires dans $\{\mathcal{C}(\mathtt{a}) \mid \mathcal{C}(\mathtt{a}) \in \mathtt{Nœuds}(\mathcal{G})\}$ aussi, et elles correspondent à celles sur $\{\mathcal{C}(\mathtt{c}) \mid \mathcal{C}(\mathtt{c}) \in \mathtt{Nœuds}(\mathcal{G})\}$.*

À la fin du processus de réparation, on obtient un ensemble non vide d'AGraph $\{\mathcal{R}_\ell\}_{1 \leqslant \ell \leqslant q}$ sans conflit. Seuls ceux dont le coût de réparation est minimal sont gardés. Soit $\mathcal{A}_\ell = \mathtt{Nœuds}(\mathcal{R}_\ell)$. Le résultat de la réparation est $\mathcal{D} = \{\mathcal{A}_\ell\}_\ell$.

Exemple 2.5.7. *La figure 2.6 montre les deux réparations possibles au conflit obtenu dans l'exemple 2.5.6 :*

- *La première réparation (à gauche) consiste à remettre en cause le fait que* `champ` *soit un champ lorrain et que des mirabelles y soient produites, en revanche, on conserve le fait que des prunes y sont produites.*

– *La deuxième réparation (à droite) consiste à remettre aussi en cause le fait que* `champ` *soit un champ lorrain et que* `i` *y soit produit.*

Le coût de la première réparation est $\mathtt{coût}(\mathtt{ChpLorrain}) + \mathtt{coût}(\mathtt{Mirabelle})$*, celui de la deuxième est* $\mathtt{coût}(\mathtt{ChpLorrain}) + \mathtt{coût}(\neg\mathtt{Mirabelle})$*. La première réparation est plus intéressante, ce choix correspond à la condition suivante sur* $\mathtt{coût}$ *:* $\mathtt{coût}(\mathtt{Mirabelle}) < \mathtt{coût}(\neg\mathtt{Mirabelle})$.

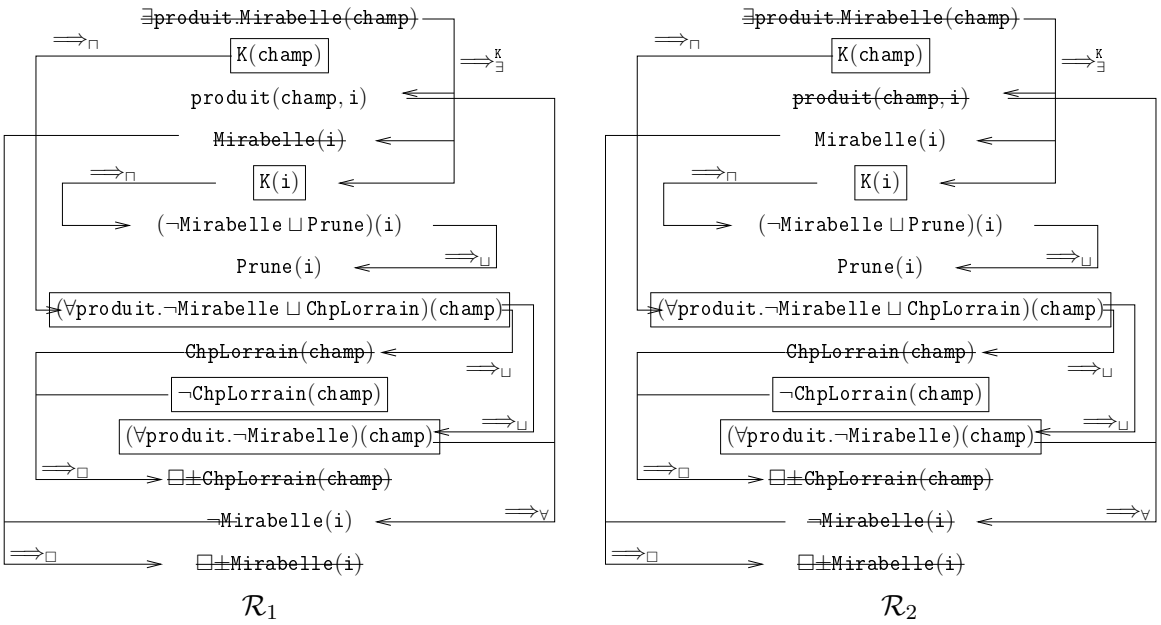


FIGURE 2.6 – AGraph obtenu après application des règles $\Longrightarrow_{\cdot}$, y compris $\Longrightarrow_{\Box}$, sur l'union des AGraph $\mathcal{G}_1$ et $\mathcal{H}_1$ obtenus dans l'exemple 2.5.5. Les formules impliquées par $\mathcal{A}_\mu$ et $\mathcal{T}$ sont encadrées pour symboliser le fait qu'elles ne seront pas remises en cause lors de la révision.

Simplifier la disjonction d'ABox $\mathcal{D}$. Si $\mathcal{A}, \mathcal{B} \in \mathcal{D}$ vérifient $\mathcal{A} \models \mathcal{B}$, alors les disjonctions d'ABox $\mathcal{D}$ et $\mathcal{D} \setminus \{\mathcal{A}\}$ sont équivalentes. Cela sert à simplifier $\mathcal{D}$ en supprimant de tels $\mathcal{A}$([5]). Après cette simplification, chaque $\mathcal{A} \in \mathcal{D}$ est réécrit pour enlever les instances `i` introduites par l'algorithme des tableaux.

D'abord, les instances `i` qui ne sont reliées à aucune instance nommée, ni directement, ni indirectement, par des assertions $\mathtt{r}(\mathtt{a}, \mathtt{b})$ sont supprimées, ce qui signifie que des assertions avec de tels `i` sont supprimées (cela peut arriver à cause de l'étape de réparation qui « déconnecte » `i` des instances nommées).

Exemple 2.5.8. *L'unique instance introduite lors du développement de l'exemple 2.5.6 est* `i`*. Elle est reliée à l'instance* `champ` *par la relation* `produit`*. La formule* $\mathtt{produit}(\mathtt{champ}, i)$ *est supprimée dans la réparation* $\mathcal{R}_2$ *(voir figure 2.6),* `i` *n'est donc reliée à aucune autre instance et elle est*

5. Dans nos tests, nous nous sommes servis de conditions nécessaires à $\mathcal{A} \models \mathcal{B}$ portant sur l'inclusion ensembliste, avec ou sans renommage d'instance introduite. Cela a entraîné une réduction conséquente de la taille de $\mathcal{D}$, ce qui incite à penser que l'algorithme présenté ici peut être grandement amélioré, en évitant la génération d'ABox inutiles.

supprimée ainsi que toutes les formules qui la contiennent. On obtient l'ABox $\mathcal{A}_2$ (en omettant les formules redondantes) :

$$\mathcal{A}_2 = \left\{ \begin{array}{ll} \neg \mathit{ChpLorrain}(\mathit{champ}), & \forall \mathit{produit}.\neg \mathit{Mirabelle}(\mathit{champ}), \\ K(\mathit{champ}) & \end{array} \right\}$$

Ensuite, une étape de « dé-skolemisation » est effectuée. Les instances ont été introduites séquentiellement (au plus une par application de règle), soit $\mathtt{i}$ la dernière instance introduite encore présente dans l'ABox. Comme pour toute instance introduite, on peut montrer qu'il existe une unique instance $\mathtt{a}$ et un seul rôle $\mathtt{r}$ tels que $\mathtt{r}(\mathtt{a},\mathtt{i})$ soit dans l'ABox. Soit $\mathtt{C}$ le concept obtenu par conjonction de tous les concepts $\mathtt{D}$ tels que $\mathtt{D}(\mathtt{i})$ soit dans l'ABox. On substitue les formules $\mathtt{r}(\mathtt{a},\mathtt{i})$ et $\mathtt{D}(\mathtt{i})$ par la formule $(\exists \mathtt{r}.\mathtt{C})(\mathtt{a})$. Par exemple, l'ensemble $\{\mathtt{r}(\mathtt{a},\mathtt{i}_1), \mathtt{A}(\mathtt{i}_1), \mathtt{s}(\mathtt{i}_1,\mathtt{i}_2), \neg\mathtt{B}(\mathtt{i}_2)\}$ est remplacé par $\{(\exists \mathtt{r}.(\mathtt{A} \sqcap \exists \mathtt{s}.\neg\mathtt{B}))(\mathtt{a})\}$. De plus, puisque l'on a choisi $\mathtt{i}$ comme étant la dernière introduite parmi les instances restantes, il n'existe par d'autre instance $\mathtt{j}$ et de rôle $\mathtt{s}$ tels que $\mathtt{s}(\mathtt{i},\mathtt{j})$ soit dans l'ABox (les formules $\mathtt{s}(\mathtt{i},\mathtt{j})$ sont ajoutées par l'algorithme lors de l'application de la règle $\Longrightarrow^{\mathtt{K}}_{\exists}$ sur un nœud $(\exists \mathtt{r}.\mathtt{C})(\mathtt{i})$, qui introduit l'instance $\mathtt{j}$, l'instance $\mathtt{i}$ ne peut donc pas avoir été introduite après $\mathtt{j}$). Ainsi, il ne reste plus de formule faisant référence à $\mathtt{i}$, cette procédure est ensuite répétée pour les instances introduites restantes. L'algorithme renvoie la valeur finale de $\mathcal{D}$.

Exemple 2.5.9. *L'instance i est toujours reliée à l'instance champ dans la la réparation $\mathcal{R}_1$ par la relation $\mathit{produit}$ (voir figure 2.6). Les concepts D tels que $D(i)$ est dans $\mathcal{R}_1$ sont K, $\mathit{Prune} \sqcup \neg\mathit{Mirabelle}$, Prune et $\neg\mathit{Mirabelle}$, Pour simplifier, on ignore K et $\mathit{Prune} \sqcup \neg\mathit{Mirabelle}$ qui sont précisés par les autres. La « dé-skolémisation » de l'instance i consiste alors à :*

$$\begin{array}{c} \textit{remplacer} \quad \left\{ \begin{array}{l} \mathit{produit}(\mathit{champ}, i), \\ \mathit{Prune}(i), \neg\mathit{Mirabelle})(i) \end{array} \right\} \\ \textit{par} \quad (\exists \mathit{produit}.(\mathit{Prune} \sqcup \neg\mathit{Mirabelle}))(\mathit{champ}) \end{array}$$

On obtient alors l'ABox $\mathcal{A}_1$ (les formules redondantes sont omises) :

$$\mathcal{A}_1 = \left\{ \begin{array}{ll} \neg \mathit{ChpLorrain}(\mathit{champ}), & \forall \mathit{produit}.\neg \mathit{Mirabelle}(\mathit{champ}), \\ \exists \mathit{produit}.(\mathit{Prune} \sqcap \neg\mathit{Mirabelle})(\mathit{champ}), & K(\mathit{champ}) \end{array} \right\}$$

Avec un coût tel que $\mathit{coût}(\mathit{ChpLorrain}) < \mathit{coût}(\neg\mathit{Mirabelle})$, $\mathcal{D} = \{\mathcal{A}_1\}$ et donc :

$$\Psi \dotplus M \quad \textit{est équivalent à} \quad \{\exists \mathit{produit}.(\mathit{Prune})(\mathit{champ})\} \cup M$$

2.5.4 Propriétés de l'algorithme

Proposition 2.5.1. *Si $\Longrightarrow^{\cdot}_{\exists}$ n'est appliquée sur un AGraph que lorsqu'aucune autre règle n'est applicable, l'algorithme des tableaux sur les AGraph termine et il est correct et complet. La répa-*

ration termine aussi, ainsi l'algorithme de révision dans son entier termine.

Démonstration. Sous condition de n'appliquer la règle $\longrightarrow_{\exists}^{\cdot}$ que quand aucune autre règle $\longrightarrow_{\cdot}$ n'est applicable, l'algorithme des tableaux termine [Baader *et al.*, 2003], seul un nombre fini de ABox est donc généré et chaque ABox est finie. Donc, d'après la remarque 2.5.3, l'application des règles $\Longrightarrow_{\cdot}$ sur $\mathcal{A}_{\psi}$ et $\mathcal{A}_{\mu}$ ($\Longrightarrow_{\exists}^{\cdot}$ étant appliquée en dernier recours) ne génère qu'un nombre fini de AGraph et chaque AGraph a un nombre de nœuds finis. Dans un AGraph, le nombre de nœuds étant finis, le nombre d'arêtes est fini aussi, donc, comme chaque règle ajoute au moins une arête, l'application des règles $\Longrightarrow_{\cdot}$ sur $\mathcal{A}_{\psi}$ et $\mathcal{A}_{\mu}$ termine.

L'étape de génération des confits entre $\mathcal{G}_i$ et $\mathcal{H}_j$ termine aussi. En effet, les nœuds $\Box\pm\mathtt{A}(\mathtt{a})$ générés par la règle $\Longrightarrow_{\Box}$ n'interfèrent pas avec l'application des autres règles $\Longrightarrow_{\cdot}$. Les AGraph générés par l'ensemble des règles $\Longrightarrow_{\cdot}$, y compris $\Longrightarrow_{\Box}$, correspondent donc aux AGraph générés par les règles $\Longrightarrow_{\cdot}$, sans $\Longrightarrow_{\Box}$, avec des nœuds $\Box\pm\mathtt{A}(\mathtt{a})$ supplémentaires. Pour chaque AGraph, le nombre de nœuds de la forme $\Box\pm\mathtt{A}(\mathtt{a})$ générés est fini car majoré par le nombre de nœuds de la forme $\mathtt{A}(\mathtt{a})$. Ainsi, mis à part l'ajout d'un nombre fini de nœuds $\mathtt{A}(\mathtt{a})$ par AGraph, la règle $\Longrightarrow_{\Box}$ ne change pas le résultat de l'algorithme des tableaux sur les AGraph. Il reste encore à montrer que le fait de ne pas arrêter le développement d'un AGraph dans lequel un conflit a été découvert n'empêche pas la terminaison. Comme précédemment, la remarque 2.5.3 nous permet de dire que les AGraph obtenus par application des règles $\Longrightarrow_{\cdot}$ correspondent aux ABox obtenues par application des règles $\longrightarrow_{\cdot}$ correspondantes sur les nœuds des AGraph. Or le fait qu'une ABox contenant un conflit ne soit plus développée n'intervient pas dans la preuve de la terminaison de l'algorithme des tableaux sur ABox [Baader *et al.*, 2003]. Ainsi l'étape de génération des conflits entre $\mathcal{G}_i$ et $\mathcal{H}_j$ termine aussi.

La correction et la complétude de l'algorithme des tableaux sur les AGraph découle de celle de l'algorithme des tableaux sur les ABox [Baader *et al.*, 2003] et de la remarque 2.5.3.

Chaque étape de réparation supprime un nœud, la réparation d'un AGraph termine donc, il n'y a qu'un nombre fini d'AGraph à réparer, ce qui implique que l'étape de réparation finit également. Il est simple de voir que les autres étapes de l'algorithme terminent aussi. Ce qui permet de conclure que l'algorithme de révision termine. □

Remarque 2.5.6. *Certaines conditions d'application des règles sur un AGraph ne sont utiles que pour la terminaison, elles ne sont nécessaires ni pour la correction ni pour la complétude. Par exemple, si $\Longrightarrow_{\sqcap}$ ou $\Longrightarrow_{\forall}$ sont appliquées sur un nœud de $\mathcal{G}$ alors que leurs conditions (ii) respectives sont fausses, $\mathcal{G}$ reste inchangé.*

Proposition 2.5.2. *L'AGraph $\mathcal{R}$ obtenu par réparation d'un $\mathcal{Q}_k$ est consistent.*

Démonstration. $\mathcal{Q}_k$ est un tableau complet, donc aucune règle ne peut lui être appliquée. L'étape de réparation supprime certains nœuds de $\mathcal{Q}_k$. La seule règle qui puisse de nouveau être appliquée après la suppression des ces nœuds (et donc les nœuds à partir desquels ils ont été générés) est $\Longrightarrow_{\exists}^{\mathtt{K}}$. Suivant la remarque 2.5.5, la réparation garantit que l'application de cette règle ne

générerait aucun conflit. Donc $\mathcal{R}$ peut être incomplet, mais il s'agit d'un sous-graphe d'un AGraph complet et dépourvu de conflit, ce qui entraîne sa cohérence. □

Proposition 2.5.3. *L'opérateur de révision défini par l'algorithme satisfait les postulats* (G1) *à* (G3)*, mais il ne satisfait ni* (G4) *ni* (G5)*.*

Démonstration.

(G1) : Soient $\Psi = \mathcal{T} \cup \mathcal{A}_\psi$ et $\mathrm{M} = \mathcal{T} \cup \mathcal{A}_\mu$ deux BC d'$\mathcal{ALC}$ partageant la même TBox $\mathcal{T}$, et $\phi \in \mathrm{M}$ une formule de M. On veut montrer que $\Psi \dot{+} \mathrm{M} \models \phi$. Lorsque $m = 0$ ou $n = 0$, $\Psi \dot{+} \mathrm{M} = \mathrm{M}$ et on a bien $\Psi \dot{+} \mathrm{M} \models \phi$. Supposons maintenant que $m > 0$ et $n > 0$. Pour tout $\mathcal{R} \in \mathcal{D}$, $\mathcal{R}$ est obtenu par réparation d'un $\mathcal{Q}_k$ issu du développement d'un certain $\mathcal{G}_i \cup \mathcal{H}_j$. Aucune conséquence de $\mathcal{H}_j$ ni de $\top \sqsubseteq \mathtt{K}$ n'a été supprimée lors de la réparation de Γ, donc $R \models \{\top \sqsubseteq \mathtt{K}\} \cup \mathcal{H}_j$. Soit $\mathcal{D}$ la disjonction d'ABox obtenue à la fin de l'algorithme. Pour formaliser la suite du raisonnement, on assimile un AGraph à l'ABox constituée des formules de ses nœuds. D'après la remarque 2.5.2, M est équivalent (au renommage près des instances introduites) à $\{\mathcal{H}_j \cup \mathcal{T}\}_i$. Or $\Psi \dot{+} \mathrm{M} = \{R \cup \mathcal{T} \mid R \in \mathcal{D}\} \models \{\mathcal{H}_j \cup \mathcal{T}\}_i$, donc $\Psi \dot{+} \mathrm{M} \models \phi$.

(G2) : Supposons que $\mathrm{Mod}(\Psi) \cap \mathrm{Mod}(\mathrm{M}) \neq \emptyset$, c'est-à-dire que $\mathrm{Mod}(\mathcal{T}) \cap \mathrm{Mod}(\mathcal{A}_\psi) \cap \mathrm{Mod}(\mathcal{A}_\mu) \neq \emptyset$. $\mathcal{A}_\psi$ et $\mathcal{A}_\mu$ étant chacun consistant modulo $\mathcal{T}$, $m > 0$ et $n > 0$. Les $\mathcal{Q}_k$ sont obtenus à partir de $\mathcal{A}_\psi \cup \mathcal{A}_\mu$ par application des règles $\Longrightarrow.$, sans tenir compte des conditions sur l'ordre d'application. Cependant, d'après la remarque 2.5.6, la correction et la complétude restent vérifiées. Par correction, comme $\mathcal{A}_\psi \cup \mathcal{A}_\mu$ est consistant, il y a des AGraph libres de conflit parmi $\{\mathcal{Q}_k\}_{1 \leqslant k \leqslant p}$. Soit $\mathcal{D}$ la disjonction d'ABox obtenue à la fin de l'étape de réparation. Les $\mathcal{Q}_k$ libres de conflit n'ont pas besoin de réparation, leur coût de réparation est donc de 0 et comme $\mathtt{coût}(\ell) > 0$, les ABox de $\mathcal{D}$ correspondent aux AGraph $\mathcal{Q}_k$ libres de conflit. Ainsi $\mathrm{Mod}(\Psi \dot{+} \mathrm{M}) = \bigcup_{\mathcal{A} \in \mathcal{D}} \mathrm{Mod}(\mathcal{A}) \cap \mathrm{Mod}(\mathcal{T})$. D'après la remarque 2.5.3, $\mathcal{D}$ est obtenu par application des règles $\longrightarrow.$ dans le même ordre sur $\mathcal{A}_\psi \cup \mathcal{A}_\mu$. Or d'après la remarque 2.5.2, la disjonction de ces ABox est équivalente (au renommage près des instances introduites) à $\mathcal{A}_\psi \cup \mathcal{A}_\mu$: $\mathrm{Mod}(\mathcal{A}_\psi \cup \mathcal{A}_\mu) = \bigcup_{\mathcal{A} \in \mathcal{D}} \mathrm{Mod}(\mathcal{A})$. Donc $\mathrm{Mod}(\Psi \dot{+} \mathrm{M}) = \mathrm{Mod}(\mathcal{A}_\psi \cup \mathcal{A}_\mu) \cap \mathrm{Mod}(\mathcal{T}) = \mathrm{Mod}(\Psi \cup \mathrm{M})$.

(G3) : Supposons que M soit cohérent. Si Ψ est incohérent, $m = 0$ et $\mathcal{D} = \mathcal{A}_\mu$, donc $\Psi \dot{+} \mathrm{M} = \mathrm{M}$ qui est cohérent. Supposons maintenant que Ψ aussi soit cohérent. On a alors $m > 0$ et $n > 0$.

D'après la propriété 2.5.2, pour tout $\mathcal{Q}_k$, l'AGraph R obtenu par réparation de $\mathcal{Q}_k$ est consistent modulo $\mathcal{T}$. Ainsi pour toute ABox $\mathcal{A} \in \mathcal{D}$, où $\mathcal{D}$ est la disjonction d'ABox obtenue à la fin de l'étape de réparation, $\mathcal{A}$ est satisfiable modulo $\mathcal{T}$. Donc $\mathcal{D}$ est aussi satisfiable modulo $\mathcal{T}$, l'étape de « dé-skolémisation » n'a pas d'influence sur la satisfiabilité de $\mathcal{D}$ et $\mathrm{Mod}(\Psi \dot{+} \mathrm{M}) = \bigcup_{\mathcal{A} \in \mathcal{D}} \mathrm{Mod}(\mathcal{A}) \cap \mathrm{Mod}(\mathcal{T}) \neq \emptyset$.

(G4) n'est pas satisfait : Soient $\Psi_1 = \{\mathtt{r}(\mathtt{a}, \mathtt{b}), \mathtt{A}(\mathtt{b})\}$, $\Psi_2 = \Psi_1 \cup \{f\}$ avec $f = \exists \mathtt{r}.(\mathtt{A} \sqcup \mathtt{B})(\mathtt{a})$, $\mathrm{M} = \{(\forall \mathtt{r}.\neg \mathtt{A})(a)\}$ ($\mathcal{T} = \emptyset$), et $\mathtt{coût}$ tel que $\mathtt{coût}(\neg \mathtt{A}) < \mathtt{coût}(\mathtt{A})$. Ψ_1 est équivalente à Ψ_2 car $\Psi_1 \models f$. Mais $\Psi_1 \dot{+} \mathrm{M}$ (qui est équivalente à $\mathrm{M} \cup \{\mathtt{A}(\mathtt{b})\}$) n'est pas équivalente à $\Psi_2 \dot{+} \mathrm{M}$ (qui est équivalente à $\mathrm{M} \cup \{f, \mathtt{A}(\mathtt{b}), \exists \mathtt{r}.(\neg \mathtt{A} \sqcap \mathtt{B})(\mathtt{a})\}$). En effet $\mathrm{M} \not\models \exists \mathtt{r}.(\neg \mathtt{A} \sqcap \mathtt{B})(\mathtt{a})$.

(G5) n'est pas satisfait : Considérons $\Psi = \{\forall \mathtt{r}.\mathtt{A}(\mathtt{a})\}$, $\mathrm{M}_1 = \{\mathtt{r}(\mathtt{a},\mathtt{b}), \neg\mathtt{A}(\mathtt{b})\}$, et $\mathrm{M}_2 = \{\mathtt{r}(\mathtt{a},\mathtt{c})\}$. $(\Psi \dot{+} \mathrm{M}_1) \cup \mathrm{M}_2$ est équivalent à $\{\mathtt{r}(\mathtt{a},\mathtt{b}), \neg\mathtt{A}(\mathtt{b}), \mathtt{r}(\mathtt{a},\mathtt{c})\}$ alors que $\Psi \dot{+} (\mathrm{M}_1 \cup \mathrm{M}_2)$ est équivalent à $\{\mathtt{r}(\mathtt{a},\mathtt{b}), \neg\mathtt{A}(\mathtt{b}), \mathtt{r}(\mathtt{a},\mathtt{c}), \mathtt{A}(\mathtt{c})\}$. □

2.5.5 Autres approches de la révision sur des logiques de descriptions

[Qi *et al.*, 2008] propose un opérateur de révision de TBox inspiré par des travaux sur le débogage d'ontologies [Kalyanpur *et al.*, 2006], [Schlobach et Cornet, 2003], dont le but est de calculer les ensembles minimaux de formules à enlever d'une TBox pour restaurer sa cohérence. Pour déboguer $\mathcal{T}$, on calcule d'abord ses MIPS (*Minimal Incoherence Preserving Sub-TBox*[6] [Schlobach et Cornet, 2003]) : ce sont les sous-ensembles de $\mathcal{T}$ minimaux pour l'inclusion parmi ceux qui entraînent l'insatisfiabilité d'au moins un concept atomique. Les formules à supprimer sont ensuite obtenues par l'algorithme des *Hitting Sets* [Reiter, 1987] sur les MIPS, cet algorithme calcule les ensembles minimaux de formules qui contiennent au moins une formule par MIPS. [Kalyanpur *et al.*, 2007] donne deux algorithmes pour calculer les MIPS, un algorithme de type « boîte noire » qui interroge un raisonneur externe, et un algorithme de type « boîte transparente » qui est similaire à l'algorithme des tableaux sur AGraph présenté en section 2.5.3 puisqu'il s'appuie sur un algorithme des tableaux sur ABox qui conserve les traces des déductions effectuées. [Qi *et al.*, 2008] étend la notion de MIPS de $\mathcal{T}$ par la notion de MIPS de $\mathcal{T}$ modulo une autre TBox $\mathcal{T}_0$. La révision de $\mathcal{T}$ par $\mathcal{T}_0$ est alors réduit au problème du débogage de la TBox $\mathcal{T}$ modulo $\mathcal{T}_0$. Cet opérateur de révision vise à maintenir la satisfiabilité des concepts atomiques et ne satisfait donc pas (G2). Cependant, on peut facilement modifier cet algorithme pour qu'il calcule une révision visant à maintenir la consistance de la BC (et qui satisfait donc (G2)) en considérant les sous-ensembles de $\mathcal{T}$ minimaux pour l'inclusion parmi ceux qui sont satisfiables, au lieu des MIPS.

Un opérateur de révision syntaxique plus fin est proposé dans [Qi *et al.*, 2006] sur la LD $\mathcal{ALCO}$[7]. Il vise à maintenir la consistance des BC et accepte des ABox non vides (un opérateur similaire sur $\mathcal{ALCN}$[8] est donné dans [Meyer *et al.*, 2005]). Au lieu de supprimer les formules impliquées dans une inconsistance, elles sont affaiblies ici en appliquant des règles de réécriture. Ces règles utilisent le constructeur de concepts « *one of* » de $\mathcal{ALCO}$ pour introduire des exceptions aux formules pour quelques instances, un coût égal au nombre d'exceptions introduites est attribué à chaque règle. Lors de la révision de Ψ par M, les règles d'affaiblissement sont ap-

6. Dans la littérature du débogage d'ontologie, le terme « incohérence » a un sens différent de celui que nous utilisons dans cette thèse, une BC est dite « incohérente » si elle admet un concept atomique insatisfiable, c'est-à-dire qu'il existe un concept atomique qui est interprété comme l'ensemble vide dans tout modèle de la BC. Le terme « inconsistance » est utilisée pour désigner l'absence de modèles.

7. La LD $\mathcal{ALCO}$ est une extension de $\mathcal{ALC}$ qui permet de définir des énumérations d'instances. Le constructeur de concepts « *one of* » (noté $\mathcal{O}$) permet de définir les concepts $\{\mathtt{a}_1, \ldots \mathtt{a}_n\}$ où $\mathtt{a}_1, \ldots \mathtt{a}_n$ sont des instances nommées, l'image de ce concept par une interprétation I est $\{{\mathtt{a}_1}^I, \ldots {\mathtt{a}_n}^I\}$.

8. La LD $\mathcal{ALCN}$ est une extension de $\mathcal{ALC}$. Le constructeur de concepts « *number restriction* » (noté $\mathcal{N}$) permet de définir les concepts $\forall_n \mathtt{r}$ et $\exists_n \mathtt{r}$ où $n \in \mathbb{N}$ et $\mathtt{r}$ est un rôle. L'image de ces concept par une interprétation I sont respectivement $\{x \in \Delta_I \mid \mathrm{card}(\{y \in \Delta_I \mid (x,y) \in \mathtt{r}^I\}) \geqslant n\}$ et $\{x \in \Delta_I \mid \mathrm{card}\{y \in \Delta_I \mid (x,y) \in \mathtt{r}^I\}) \leqslant n\}$. Le concept $\forall_n \mathtt{r}$ est équivalent au concept $\exists_{n+1} \mathtt{r}$.

pliquées sur les formules de Ψ jusqu'à ce qu'elles deviennent consistantes avec M, le coût associé au résultat est égal à la somme des coûts des règles appliquées. Le résultat de la révision est constitué de la disjonction des BC obtenues pour un coût minimal. Nous considérons que cet opérateur de révision fait de l'apprentissage « par cœur » de l'interprétation des concepts puisque cet apprentissage consiste à ajouter des exceptions aux formules. Pourtant, cet opérateur reste grossier concernant les instances puisque des formules en conflit à propos d'une instance sont simplement supprimées.

À notre connaissance, le seul opérateur de révision sémantique sur LD est présenté dans [Qi et Du, 2009]. Il est défini à l'aide d'une « distance » entre interprétations partageant le même domaine, la révision d'une BC Ψ par M est déterminée par l'ensemble des modèles de M qui minimisent la distance avec les modèles de Ψ. La distance entre deux interprétations dont les images des rôles sont identiques est définie comme le nombre de concepts atomiques dont l'image change d'une interprétation à l'autre. Lorsque l'interprétation d'un rôle change, la distance est alors maximale (elle est égale au nombre total de concepts atomiques). L'opérateur de révision obtenu satisfait les postulats (G1) à (G6). Cependant il n'est pas garanti que l'ensemble des interprétations obtenues puisse être exprimé dans le même formalisme. En effet, le calcul de la révision suivant cet opérateur est réduit au problème d'oubli de concept [Wang *et al.*, 2008], pour lequel il est montré que les résultats restent exprimables dans le même formalisme que des TBox exprimées dans la LD *DL-Lite*[9], mais ce n'est plus vrai si l'on ajoute une ABox non vide. [Wang *et al.*, 2009] montre que ce résultat est faux pour $\mathcal{ALC}$ même avec une ABox vide. [Qi et Du, 2009] donne aussi une variante de cet opérateur de révision dont le but est de préserver la satisfiabilité des concepts atomiques. Elle est obtenue en se restreignant aux modèles de M dont l'image des concepts atomiques est non vide.

2.6 Perspectives concernant la révision en $\mathcal{ALC}$

L'algorithme présenté en section 2.5.3 est dans un état préliminaire et nous avons relevé quelques exemples en section 2.6.1 pour lesquels les résultats obtenus ne correspondent pas à ce qui était attendu. En particulier, plusieurs postulats de la révision ne sont pas vérifiés par l'opérateur de révision défini par cet algorithme. Cela s'explique par le fait que la sémantique de cet opérateur n'a pas été définie. Nous donnons en section 2.6.2 une piste pour faire correspondre l'algorithme avec un opérateur de révision défini sémantiquement.

L'algorithme tel que défini en section 2.5.3 effectue la révision entre ABox modulo une TBox, nous présentons en section 2.6.3 une piste pour étendre cet algorithme à la révision de bases de

9. Le langage de *DL-Lite* est constitué de concepts de la forme $\mathtt{A}$, $\top$, $\bot$, $\neg\mathtt{C}$, $\mathtt{C} \sqcap \mathtt{D}$ ou $\geqslant_n \mathtt{r}$, où $\mathtt{A}$ est un concept atomique, $\mathtt{C}$ et $\mathtt{D}$ des concepts quelconques, $n \in \mathbb{N}$ et $\mathtt{r}$ un rôle. Un concept de la forme $\geqslant_n \mathtt{r}$ représente l'ensemble des instances reliées à au moins n autres instances par $\mathtt{r}$, son image par une interprétation I vaut $\geqslant_n \mathtt{r}^I = \{u \in \Delta_I \mid \text{card}\{v \in \Delta_I \mid (u,v) \in \mathtt{r}^I\} \geqslant n\}$. La sémantique des autres formes de concepts est identique à celle de $\mathcal{ALC}$. Les rôles sont de la forme $\mathtt{p}$ et $\mathtt{p}^-$ où $\mathtt{p}$ est un rôle atomique et $\mathtt{p}^-$ est qualifié de *rôle inverse de* $\mathtt{p}$, son image par une interprétation I vaut $\mathtt{p}^{-I} = \{(v,u) \in \Delta_I \times \Delta_I \mid (u,v) \in \mathtt{p}^I\}$.

connaissances en $\mathcal{ALC}$ dont les TBox peuvent être différentes.

Dans le but d'obtenir une implantation efficace de cet algorithme, nous envisageons d'exploiter un moteur d'inférences existant, nous présentons cette perspective en section 2.6.4.

Une autre perspective que nous ne détaillons pas ici concerne l'extension à des logiques de descriptions plus expressives en particulier en ajoutant des domaines concrets [Baader et Hanschke, 1991] dans le but d'intégrer les travaux de la section 2.4.

2.6.1 Exemples problématiques

Dépendance à la syntaxe

L'exemple proposé dans la preuve de la propriété 2.5.3 (page 57, à propos du postulat (G4)) montre que le résultat de la révision effectuée par l'algorithme dépend de la syntaxe. On considère les BC suivantes :

$$\Psi_1 = \{\mathtt{r}(\mathtt{a},\mathtt{b}), \mathtt{A}(\mathtt{b})\} \qquad\qquad \mathrm{M} = \{(\forall \mathtt{r}.\neg\mathtt{A})(a)\}$$
$$\Psi_2 = \Psi_1 \cup \{f\} \qquad \text{avec} \quad f = \exists \mathtt{r}.(\mathtt{A} \sqcup \mathtt{B})(\mathtt{a})$$

$\Psi_1 \models f$ donc Ψ_1 est équivalent à Ψ_2 pourtant lorsque $\mathtt{coût}(\neg\mathtt{A}) < \mathtt{coût}(\mathtt{A})$, la révision par l'algorithme présenté en section 2.5.3 donne :

$$\Psi_1 \dot{+} \mathrm{M} = \mathrm{M} \cup \{\mathtt{A}(\mathtt{b})\}$$
$$\Psi_2 \dot{+} \mathrm{M} = \mathrm{M} \cup \{\mathtt{A}(\mathtt{b}), \exists \mathtt{r}.(\neg\mathtt{A} \sqcap \mathtt{B})(\mathtt{a})\}$$

$\Psi_2 \dot{+} \mathrm{M}$ n'est pas équivalent à $\Psi_2 \dot{+} \mathrm{M}$.

La suppression des formules f de l'ABox révisée $\mathcal{A}_\psi$ telles que $\mathcal{A}_\psi \setminus \{f\} \models f$ est un prétraitement qui permet d'éviter ce type de contre-exemple. Nous n'avons pas trouvé d'exemple montrant la dépendance à la syntaxe que ce prétraitement ne résoudrait pas, mais nous n'avons pas non plus de preuve qu'elle assure l'indépendance à la syntaxe. En section 2.6.2, nous proposons quelques pistes pour obtenir un algorithme correspondant à un opérateur de révision défini grâce à une distance entre interprétations. Cet opérateur de révision étant défini sémantiquement, le résultat de l'algorithme ne dépendra pas de la syntaxe.

Certaines réparations attendues ne sont pas obtenues

On considère une variante de l'exemple 2.5.1 (page 41) :

$$\mathcal{A}_\psi = \{\mathtt{ChpMirabelle}(\mathtt{champ}), \mathtt{exploite}(\mathtt{léo}, \mathtt{champ})\}$$
$$\mathcal{A}_\mu = \{(\forall \mathtt{exploite}.\neg\mathtt{ChpLorrain})(\mathtt{léo})\}$$
$$\mathcal{T} = \{\mathtt{ChpMirabelle} \sqsubseteq \mathtt{ChpLorrain}\}$$

Il y a un conflit entre $\mathcal{A}_\psi$ et $\mathcal{A}_\mu$ puisque le fait que `champ` produise des mirabelles entraîne le fait qu'il s'agit d'un champ lorrain, or `léo` n'exploite pas de champ lorrain. On s'attend à deux révisions possibles :

- Remettre en cause le fait que le champ produise des mirabelles.
- Remettre en cause le fait que le champ soit exploité par Paul.

Or comme nous allons le voir, seule la deuxième option est proposée (quelle que soit la fonction `coût`).

La figure 2.7 montre le développement de $\mathcal{A}_\psi$. Bien que `ChpLorrain(champ)` soit une conséquence à la fois de $(\neg$`ChpMirabelle`$\sqcup$`ChpLorrain`$)$`(champ)` et de `ChpMirabelle(champ)`, aucun arc n'est créé dans $\mathcal{G}_1$ entre les nœuds `ChpMirabelle(champ)` et $\neg$`ChpLorrain(champ)`.

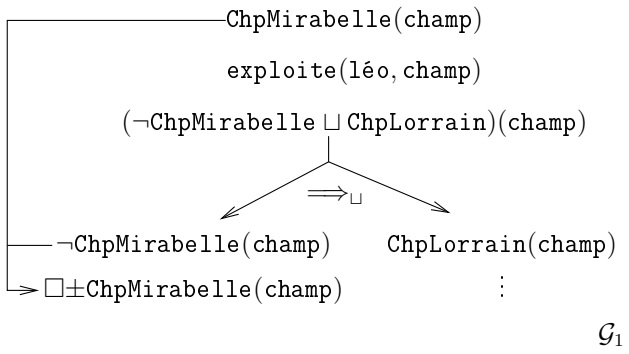


FIGURE 2.7 – Développement de $\mathcal{A}_\psi$. Pour simplifier, `K(léo)` n'est pas développé. Un seul AGraph $\mathcal{G}_i$ ouvert est généré.

Pour simplifier, on ne développe pas `K(léo)`), on a donc $\mathcal{H}_1 = \mathcal{A}_\mu$. Le développement de $\mathcal{G}_1 \cup \mathcal{H}_1$ génère un conflit entre les nœuds `ChpLorrain(champ)` et $\neg$`ChpLorrain(champ)` (figure 2.8).

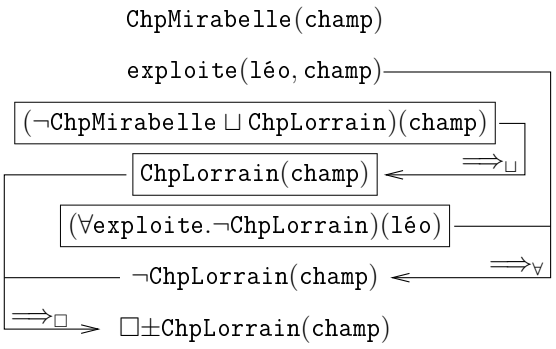


FIGURE 2.8 – Génération des conflits entre $\mathcal{G}_1$ et $\mathcal{H}_1 = \mathcal{A}_\mu$.

Une seule réparation est proposée (figure 2.9), elle consiste à supprimer le nœud `exploite(léo, champ)` qui entraîne $\neg$`ChpLorrain(champ)` avec $\mathcal{A}_\mu$.

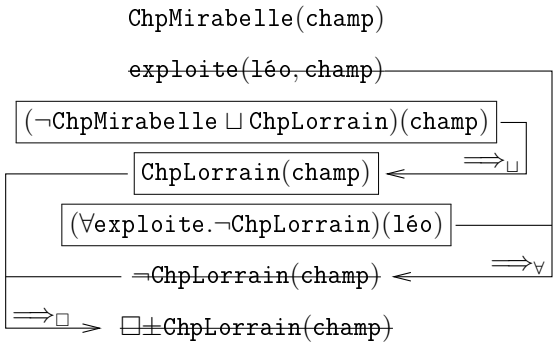


FIGURE 2.9 – Génération des conflits entre $\mathcal{G}_1$ et $\mathcal{H}_1 = \mathcal{A}_\mu$.

L'autre réparation attendue consiste à supprimer le nœud ChpMirabelle(champ) qui entraîne ChpLorrain(champ) avec $\mathcal{T}$. Elle n'est pas proposée puisqu'il n'y a pas d'arc entre ChpMirabelle(champ) et ChpLorrain(champ). Le fait que ChpLorrain(champ) soit une conséquence de ChpMirabelle(champ) et de (¬ChpMirabelle⊔ChpLorrain)(champ) apparaît dans le développement des tableaux par le fait que ChpMirabelle(champ) provoque un conflit dans la branche de droite. Un nœud $\mathtt{C}_i(\mathtt{a})$ issu de $\mathtt{D}(\mathtt{a}) = (\mathtt{C}_1 \sqcup \ldots \sqcup \mathtt{C}_k)(\mathtt{a})$ est conséquence de $\mathtt{D}(\mathtt{a})$ et des nœuds provoquant des conflits dans les autres branches issues de $\mathtt{D}(\mathtt{a})$.

Question 1. *Lors du développement de $\mathcal{A}_\psi$, on considère un nœud $\mathcal{C}_i(\mathtt{a})$ généré à partir de $(\mathcal{C}_1 \sqcup \ldots \sqcup \mathcal{C}_k)(\mathtt{a})$ par application de la règle $\Longrightarrow_\sqcup$ et tel que toutes les autres branches générées à partir de $(\mathcal{C}_1 \sqcup \ldots \sqcup \mathcal{C}_k)(\mathtt{a})$ sont fermées. Est-il suffisant d'ajouter des arêtes $\Longrightarrow_\sqcup$ entre les causes des conflits dans les autres branches issues de $(\mathcal{C}_1 \sqcup \ldots \sqcup \mathcal{C}_k)(\mathtt{a})$ et $\mathcal{C}_i(\mathtt{a})$ pour résoudre le problème ?*

Remarque 2.6.1. *Lorsqu'un nœud N doit être supprimé, tous les nœuds reliés à N par des arcs $\Longrightarrow_\sqcap$ et $\Longrightarrow_\exists^K$ doivent être supprimés aussi. En revanche, il n'est pas nécessaire de supprimer tous les nœuds reliés à N par $\Longrightarrow_\forall$, il suffit de supprimer un nœud par couple $\mathtt{r}(\mathtt{a},\mathtt{b})$, $(\forall \mathtt{r}.\mathcal{C})(\mathtt{a})$ à partir desquels à été généré N. Il peut y avoir plus d'un couple de ce type entraînant N, par exemple avec $\mathcal{A}_\psi = \{\mathtt{r}(\mathtt{a},\mathtt{b}), \mathtt{r}(\mathtt{c},\mathtt{b}), \neg\mathcal{C}(\mathtt{b})\}$ et $\mathcal{A}_\mu = \{(\forall \mathtt{r}.\mathcal{C})(\mathtt{a}), (\forall \mathtt{r}.\mathcal{C})(\mathtt{c})\}$, quatre arcs $\Longrightarrow_\forall$ pointent sur le nœud $\mathcal{C}(\mathtt{b})$. Il y a donc un peu de travail à faire pour identifier les possibilités de réparation possibles après la suppression du nœud N.*

Ces considérations se compliquent si l'on ajoute des arcs $\Longrightarrow_\sqcup$ comme proposé plus haut. Pour supprimer un nœud $\mathcal{C}_{i_0}(\mathtt{a})$ issu de $(\mathcal{C}_1 \sqcup \ldots \sqcup \mathcal{C}_k)(\mathtt{a})$ sur lequel pointent plusieurs arcs $\Longrightarrow_\sqcup$, il faut supprimer assez de nœuds à l'origine de ces arcs pour qu'il n'y ait plus de conflit dans une des branches issues de l'un des $\mathcal{C}_i(\mathtt{a})$. Comment exprimer ces contraintes ?

2.6.2 Vers une définition sémantique

On cherche à définir un opérateur de révision à l'aide d'une « distance » entre interprétations, comme présenté en section 2.3 dans le cadre propositionnel. C'est-à-dire que l'on veut caractériser

la révision d'une BC $\Psi = \mathcal{T} \cup \mathcal{A}_\psi$ par une BC $\mathrm{M} = \mathcal{T} \cup \mathcal{A}_\mu$ par les modèles de M les plus proches de ceux de Ψ selon la « distance » d ($\mathcal{A}_\psi$ et $\mathcal{A}_\mu$ sont des ABox et $\mathcal{T} = \{\top \sqsubseteq \mathtt{K}\}$). Cette section est une perspective de recherche avec des pistes de travail.

On note $\mathcal{C}$ l'ensemble des concepts atomiques et $\mathcal{R}$ l'ensemble des rôles employés dans Ψ et M. On notera aussi instances($\mathcal{A}$) l'ensemble des instances apparaissant dans une ABox $\mathcal{A}$.

Distance entre interprétations

On supposera ici que les interprétations sont définies pour un ensemble d'instances $\mathcal{I}nst$ comprenant les instances apparaissant dans Ψ et M. Il est possible d'étendre toute interprétation définie pour un sous-ensemble de $\mathcal{I}nst$ en associant une valeur arbitraire de son domaine aux instances manquantes.

La définition d'une distance entre interprétations d'une logique $\mathcal{ALC}$ sur le vocabulaire donné par $\mathcal{C}$, $\mathcal{R}$ et $\mathcal{I}nst$ est plus compliquée qu'en logique propositionnelle. En particulier, parce qu'elles ne partagent pas toutes le même domaine d'interprétation et parce que ce domaine peut être infini. Heureusement, [Baader *et al.*, 2003] montre que toute BC cohérente en $\mathcal{ALC}$ admet un modèle fini. On peut donc se contenter de comparer les interprétations finies.

Si la réponse à la question suivante est « oui », alors on peut se contenter de ne comparer que les interprétations qui partagent le même domaine et dont l'interprétation des instances coïncide.

Question 2. *Si Ψ et M sont cohérentes, admettent-elles des modèles finis, respectivement I et J, tels que $\Delta_I = \Delta_J$, et pour toute instance $\mathtt{a} \in \mathcal{I}nst$, $\mathtt{a}^I = \mathtt{a}^J$?*

On considère dans la suite une « distance » entre les interprétations qui prend la valeur $+\infty$ lorsque les interprétations n'ont pas le même domaine ou ne coïncident pas pour l'interprétation des instances.

Exemple 2.6.1. *On définit la « distance » d telle que pour tout couple d'interprétations (I, J) tel que* instances(I) = instances(J), *$\Delta_I = \Delta_J$ et pour tout $\mathtt{a} \in$* instances(I), *$\mathtt{a}^I = \mathtt{a}^J$:*

$$d(I, J) = \sum_{\mathtt{A} \in \mathcal{C}} card(\mathtt{A}^I \triangle \mathtt{A}^J) + \sum_{\mathtt{r} \in \mathcal{R}} card(\mathtt{r}^I \triangle \mathtt{r}^J)$$

et $d(I, J) = +\infty$ sinon, où pour deux ensembles U et V, $U \triangle V$ est la différence symétrique entre U et V : $U \triangle V = (U \setminus V) \cup (V \setminus U)$.

Définition d'un opérateur de révision

L'assignation fidèle engendrée par une telle « distance » d ne vérifie par (A4′), c'est-à-dire que lors de la révision de Ψ par M, l'ensemble des modèles de M minimisant la distance avec les modèles de Ψ ne peut pas être représenté par une BC de $\mathcal{ALC}$. En effet, puisque la distance ne compare que des interprétations finies, l'ensemble d'interprétations retenu ne contient que des interprétations finies, or toute BC satisfiable de $\mathcal{ALC}$ admet des modèles infinis.

Une solution est de définir $\Psi \dot{+} \mathrm{M}$ comme une BC dont les modèles couvrent les modèle retenus par la minimisation de la distance et qui a le moins possible de modèles supplémentaires.

Question 3. *Existe-t-il une BC Φ telle que* $\mathrm{Mod}(\Phi)$ *soit minimal pour l'inclusion parmi les modèles de BC contenant* $\underset{d(\mathrm{Mod}(\Psi),\cdot)}{\mathrm{Min}} \mathrm{Mod}(M)$ *?*

Calcul de cet opérateur de révision

Ψ et M ont une infinité de modèles, comment les comparer pour retenir les modèles de M qui minimisent la distance avec les modèles de Ψ ? La remarque 2.5.2 (page 45) établit une équivalence de satisfiabilité, modulo une TBox $\mathcal{T}$, entre une ABox $\mathcal{A}_0$ et les ABox $\mathcal{A}_1, \ldots, \mathcal{A}_n$ obtenues par application de l'algorithme des tableaux sur $\mathcal{A}_0$.

Question 4. *Peut-on établir une correspondance entre les modèles de $\mathcal{A}_0$ et les modèles des $\mathcal{A}_i$?*

Cette correspondance permettrait de partitionner l'ensemble des modèles de Ψ suivant les modèles des ABox $\mathcal{G}_i$ obtenues par application de la règle des tableaux sur $\mathcal{A}_\psi$ (on reprend les notation de la section 2.5.3 en considérant des ABox à la place des AGraph). De même les modèles de M seraient partitionnés selon les modèles des $\mathcal{H}_j$. La proposition 1.2.3 (page 11) permettrait ainsi de réduire le calcul de la minimisation de la distance entre modèles de Ψ et M à la minimisation de la distance entre modèles de $\mathcal{G}_i$ et $\mathcal{H}_j$.

Question 5. *Si* $\mathrm{Mod}(\Psi \dot{+} M)$ *est défini comme suggéré par la question 3, peut-il être calculé à l'aide des* $\mathrm{Mod}(\mathcal{G}_i \dot{+} \mathcal{H}_j)$ *?*

Il nous reste à montrer que les réparations effectuées sur les ABox $\mathcal{Q}_k$ issues du développement de l'algorithme des tableaux sur $\mathcal{G}_i \cup \mathcal{H}_j$ et les différences entre les modèles de $\mathcal{G}_i$ et les modèles de $\mathcal{H}_j$.

Question 6. *On suppose que le développement des tableaux sur $\mathcal{G}_i \cup \mathcal{H}_j$ ne génère qu'une seule ABox $\mathcal{Q}_k$, qui est ensuite réparée. Les implications suivantes sont-elles vraies ?*

- *Si une formule $\mathtt{A}(\mathtt{a})$ est supprimée, alors pour tout modèle I de $\mathcal{G}_i$ et tout modèle J de $\mathcal{H}_j$, $\mathtt{a}^I \in \mathtt{A}^I$ et $\mathtt{a}^J \notin \mathtt{A}^J$.*
- *Si une formule $\neg\mathtt{A}(\mathtt{a})$ est supprimée, alors pour tout modèle I de $\mathcal{G}_i$ et tout modèle J de $\mathcal{H}_j$, $\mathtt{a}^I \notin \mathtt{A}^I$ et $\mathtt{a}^J \in \mathtt{A}^J$.*
- *Si une formule $\mathtt{r}(\mathtt{a}, \mathtt{b})$ est supprimée, alors pour tout modèle I de $\mathcal{G}_i$ et tout modèle J de $\mathcal{H}_j$, $(\mathtt{a}^I, \mathtt{b}^I) \in \mathtt{r}^I$ et $(\mathtt{a}^J, \mathtt{b}^J) \notin \mathtt{r}^J$.*

Comment détecter le fait que tout modèle I de $\mathcal{G}_i$ et pour tout modèle J de $\mathcal{H}_j$, $(\mathtt{a}^I, \mathtt{b}^I) \notin \mathtt{r}^I$ et $(\mathtt{a}^J, \mathtt{b}^J) \in \mathtt{r}^J$?

On peut remarquer que la suppression de deux nœuds $\mathtt{A}(\mathtt{a})$ et $\mathtt{A}(\mathtt{b})$ correspond à un seul changement sur un modèle I tel que $\mathtt{a}^I = \mathtt{b}^I$. Cela a une influence sur le calcul de la distance et donc du résultat de la révision. Cela explique la dépendance à la syntaxe du résultat de la révision de l'exemple présenté en section 2.6.1. suivant l'algorithme présenté en section 2.5.3. Cet exemple est repris dans l'exemple suivant mais avec la « distance » de l'exemple 2.6.1.

Exemple 2.6.2. *On considère les révisions de Ψ par M et de Ψ' par M où :*

$$\Psi = \{r(a, b), A(b)\}$$
$$\Psi' = \{r(a, b), A(b), (\exists r.(A \sqcup B))(a)\}$$
$$M = \{(\forall r.\neg A)(a)\}$$

Ψ est équivalent à Ψ'.

On considère d'abord la révision de Ψ par M. Aucune règle n'est applicable sur ces deux ABox, et une seule ABox est obtenue après union de $\mathcal{G}_1 = \Psi$ et $\mathcal{H}_1 = M$:

$$\mathcal{Q}_1 = \Psi \cup M \cup \{\neg A(b), \Box\pm A(b)\}$$

Deux réparations sont possibles :

$$\mathcal{R}_1 = M \cup \{A(b)\} \qquad \textit{en supprimant } r(a, b)$$
$$\mathcal{R}_2 = M \cup \{r(a, b), \neg A(b)\} \qquad \textit{en supprimant } A(b)$$

Avec la « distance » d de l'exemple 2.6.1, $\Psi \dot{+}^d M$ vaut :

- *$\mathcal{R}_1$ si $\mathtt{coût}(A) > \mathtt{coût}(r)$.*
- *$\mathcal{R}_2$ si $\mathtt{coût}(r) > \mathtt{coût}(A)$*
- *La disjonction des deux si les coûts sont égaux.*

On considère maintenant la révision de Ψ' par M. Le développement de Ψ' génère deux ABox :

$$\mathcal{G}'_1 = \Psi' \cup \{r(a, i), A(i)\}$$
$$\mathcal{G}'_2 = \Psi' \cup \{r(a, i), B(i)\}$$

où i est une nouvelle instance. Aucune règle n'est applicable sur M, on n'a donc qu'une seule $\mathcal{H}_j$: $\mathcal{H}_1 = M$. Les ABox $\mathcal{G}'_1 \cup \mathcal{H}_1$ et $\mathcal{G}'_2 \cup \mathcal{H}_1$ génèrent une seule ABox chacune, respectivement $\mathcal{Q}'_1$ et $\mathcal{Q}'_2$:

$$\mathcal{Q}'_1 = \mathcal{G}'_1 \cup \mathcal{H}_1 \cup \{\neg A(b), \neg A(i), \Box\pm A(b), \Box\pm A(i)\}$$
$$\mathcal{Q}'_2 = \mathcal{G}'_2 \cup \mathcal{H}_1 \cup \{\neg A(b), \neg A(i), \Box\pm A(b)\}$$

Quatre réparations sont possibles pour $\mathcal{Q}'_1$:

$$\mathcal{R}'_1 = \mathcal{H}_1 \cup \{A(b), A(i)\} \qquad \textit{en supprimant } r(a, b) \textit{ et } r(a, i)$$
$$\mathcal{R}'_2 = \mathcal{H}_1 \cup \{A(b), r(a, i), \neg A(i)\} \qquad \textit{en supprimant } r(a, b) \textit{ et } A(i)$$
$$\mathcal{R}'_3 = \mathcal{H}_1 \cup \{r(a, b), \neg A(b), A(i)\} \qquad \textit{en supprimant } A(b) \textit{ et } r(a, i)$$
$$\mathcal{R}'_4 = \mathcal{H}_1 \cup \{r(a, b), \neg A(b), r(a, i), \neg A(i)\} \qquad \textit{en supprimant } A(b) \textit{ et } A(i)$$

Deux réparations sont possibles pour $\mathcal{Q}_2$:

$$\mathcal{R}'_5 = \mathcal{H}_1 \cup \{A(b), \neg A(i), B(i)\} \qquad \textit{en supprimant } r(a,b)$$
$$\mathcal{R}'_6 = \mathcal{H}_1 \cup \{r(a,b), \neg A(b), r(a,i), \neg A(i), B(i)\} \qquad \textit{en supprimant } A(b)$$

Tout modèle I de $\mathcal{R}'_1$ tel que $b^I \neq i^I$ est à une distance d'au moins $2 \cdot \mathtt{coût}(r)$ des modèles de $\mathcal{G}_1$, ceux de $\mathcal{R}'_2$ et $\mathcal{R}'_3$ sont à une distance d'au moins $\mathtt{coût}(A) + \mathtt{coût}(r)$ des modèles de $\mathcal{G}'_1$, et ceux de $\mathcal{R}'_4$ sont à une distance d'au moins $2 \cdot \mathtt{coût}(A)$ des modèles de $\mathcal{G}'_1$. Or il existe un modèle J_1 de $\mathcal{R}'_5$ à distance $\mathtt{coût}(r)$ du modèle I_1 de $\mathcal{G}'_2$, et un modèle J_2 de $\mathcal{R}'_6$ à distance $\mathtt{coût}(A)$ du modèle I_2 de $\mathcal{G}'_2$:

$$d(I_1, J_1) = \mathtt{coût}(r) \quad \textit{avec :} \quad A^{I_1} = A^{J_1} = \{b\} \quad B^{I_1} = B^{J_1} = \{i\} \quad r^{I_1} = \{(a,b)\} \quad r^{J_1} = \emptyset$$
$$d(I_2, J_2) = \mathtt{coût}(A) \quad \textit{avec :} \quad A^{I_2} = \{b\} \quad A^{J_2} = \emptyset \quad B^{I_2} = B^{J_2} = \{i\} \quad r^{I_2} = r^{J_2} = \emptyset$$

Si on ne considère pas les interprétations I telles que $b^I = i^I$, les résultats retenus pour la révision de Ψ' par M sont :

- *$\mathcal{R}'_5$ si $\mathtt{coût}(A) > \mathtt{coût}(r)$, elle est équivalente, après « dé-skolémisation », à $M \cup \{A(b)\}$, on retrouve dans ce cas le résultat de la révision de Ψ par M.*
- *$\mathcal{R}'_4$ si $\mathtt{coût}(r) > \mathtt{coût}(A)$, elle est équivalente, après « dé-skolémisation », à $M \cup \{r(a,b), \neg A(b), (\exists r.\neg A \sqcap B)(a,i)\}$ et n'est pas équivalente au résultat de la révision de Ψ par M.*
- *La disjonction des deux si $\mathtt{coût}(r) = \mathtt{coût}(A)$, là encore le résultat n'est pas équivalent à la révision de Ψ par M.*

Si on considère les I telles que $b^I = i^I$, on retrouve les résultats de la révision de Ψ par M. $\mathcal{R}'_2$, $\mathcal{R}'_3$ et $\mathcal{R}'_5$ n'ont pas de modèle I tel que $b^I = i^I$. Il existe un modèle J_3 de $\mathcal{R}'_1$ à distance $\mathtt{coût}(r)$ du modèle I_3 de $\mathcal{G}'_1$, un modèle J_4 de $\mathcal{R}'_4$ à distance $\mathtt{coût}(A)$ du modèle J_3 de $\mathcal{G}'_1$:

$$d(I_3, J_3) = \mathtt{coût}(r) \quad \textit{avec :} \quad A^{I_3} = A^{J_3} = \{b\} \quad B^{I_3} = B^{J_3} = \emptyset \quad r^{I_3} = \{(a,b)\} \quad r^{J_3} = \emptyset$$
$$d(I_4, J_4) = \mathtt{coût}(A) \quad \textit{avec :} \quad A^{I_4} = \{b\} \quad A^{J_4} = \emptyset \quad B^{I_4} = B^{J_4} = \emptyset \quad r^{I_4} = r^{J_4} = \{(a,b)\}$$

Donc si $\mathtt{coût}(A) < \mathtt{coût}(r)$, $\Psi' \dot{+} M$ est équivalent à $\mathcal{R}'_4 \vee \mathcal{R}'_6$. Or $\mathcal{R}'_4 \subseteq \mathcal{R}'_6$, $\Psi' \dot{+} M$ est donc équivalent à $\mathcal{R}'_4$, soit, après la « dé-skolémisation » de i : $M \cup \{r(a,b), \neg A(b)\}$.

Si $\mathtt{coût}(A) > \mathtt{coût}(r)$, $\Psi' \dot{+} M$ est équivalent à $\mathcal{R}'_1 \vee \mathcal{R}'_5$. Après la « dé-skolémisation » de i, $\mathcal{R}'_1$ devient $M \cup \{A(b)\}$ et $\mathcal{R}'_5$ devient $M \cup \{A(b), (\exists r.(\neg A \sqcap B))(a)\}$ qui est plus spécifique que $M \cup \{A(b)\}$. Donc $\Psi' \dot{+} M$ est équivalent à $M \cup \{A(b)\}$.

Si $\mathtt{coût}(A) = \mathtt{coût}(r)$, $\Psi' \dot{+} M$ est équivalent à la disjonction des deux résultats précédents.

Remarque 2.6.2. *Dans l'exemple, lorsque l'on identifie b et i, les ABox $\mathcal{R}'_2$, $\mathcal{R}'_3$ et $\mathcal{R}'_5$ deviennent insatisfiables. Il est donc nécessaire de réévaluer la satisfiabilité des ABox après avoir identifié des instances en poursuivant l'algorithme des tableaux. L'identification de deux in-*

stances peut provoquer de nouvelles applications de règles. Par exemple, si $r(a, c) \in \mathcal{R}_l$ *et* $(\forall r.C)(b) \in \mathcal{R}_l$*, l'identification des instances* a *et* b *provoque l'application de la règle* $\Longrightarrow_\forall$*. De plus, si* $C = (\exists r.D)$ *le nœud* $(\exists r.D)(c)$ *généré provoque l'application de la règle* $\Longrightarrow_\exists^K$ *et donc l'introduction d'une nouvelle instance.*

Question 7. *L'identification d'instances relance dont un processus de développement des tableaux et de réparation. Peut-on assurer la terminaison ? Le coût de la réparation avant identification des instances donne une borne supérieure pour le coût minimal, est-ce suffisant ?*

2.6.3 Vers la révision d'une BC quelconque (ABox et TBox non vides)

Introduction d'une variable universelle x

On introduit une nouvelle instance x qui servira de variable « universelle », c'est-à-dire que tout ce qui est déduit pour x doit être vrai pour toutes les instances. Ainsi, si on assimile les formules de $\mathcal{ALC}$ à des formules en logique du premier ordre (L1O), $\forall \mathtt{x}\ \mathtt{K}(\mathtt{x})$ sera interprété comme $\top \sqsubseteq \mathtt{K}$.

D'après la remarque 2.5.2, étant donné une ABox $\mathcal{A}$ et une TBox $\mathcal{T} = \{\top \sqsubseteq \mathtt{K}\}$, la disjonction d'ABox $\mathcal{D}$ obtenue après application des règles $\longrightarrow$. est équivalente (au renommage près des instances introduites) à $\mathcal{A}$ modulo $\mathcal{T}$. En prenant les notations de la logique du premier ordre :

$$\mathcal{A} \cup \mathcal{T} \quad \text{est équivalent à} \quad \bigvee_{\mathcal{B} \in \mathcal{D}} \mathcal{B} \wedge \mathcal{T}$$

Question 8. *Soit* $\mathcal{A}^K(x) = \mathcal{A}^K \cup \{K(x)\}$ *et* $\mathcal{D}(x)$ *la disjonction d'ABox obtenue après l'application des règles* $\longrightarrow$. *sur* $\mathcal{A}^K(x)$*.* $\mathcal{D}(x)$ *est équivalente à* $\mathcal{A}^K(x)$ *modulo* $\mathcal{T}$ *(*x *étant traitée comme une instance normale). La propriété suivante est-elle vraie ?*

$$\mathcal{A} \cup \mathcal{T} \quad \textit{est équivalent à} \quad \forall x \bigvee_{\mathcal{B}(x) \in \mathcal{D}(x)} \mathcal{B}(x)$$

Générer les conflits entre $\mathcal{A}_\psi \cup \mathcal{T}_\psi$ et $\mathcal{A}_\mu \cup \mathcal{T}_\mu$ On considère la révision de la BC $\Psi = \mathcal{A}_\psi \cup \mathcal{T}_\psi$ par $\mathrm{M} = \mathcal{A}_\mu \cup \mathcal{T}_\mu$.

On note $\mathcal{A}_\psi^{\mathtt{K}_\psi}(\mathtt{x}) = \mathcal{A}_\psi^{\mathtt{K}_\psi} \cup \{\mathtt{K}_\psi(\mathtt{x})\}$ et $\mathcal{A}_\mu^{\mathtt{K}_\mu}(\mathtt{x}) = \mathcal{A}_\mu^{\mathtt{K}_\mu} \cup \{\mathtt{K}_\mu(\mathtt{x})\}$.

On génère les conflits entre $\mathcal{A}_\psi^{\mathtt{K}_\psi}(\mathtt{x})$ et $\mathcal{A}_\mu^{\mathtt{K}_\mu}(\mathtt{x})$:

- On développe de $\mathcal{A}_\psi^{\mathtt{K}_\psi}(\mathtt{x})$ avec les règles $\Longrightarrow_\sqcap$, $\Longrightarrow_\sqcup$, $\Longrightarrow_\exists^{\mathtt{K}_\psi}$, $\Longrightarrow_\forall$, et on retient l'ensemble $\{\mathcal{G}_i(\mathtt{x})\}_{1 \leqslant i \leqslant m}$ des AGraph sans conflit. Si la propriété proposée en question 8 est vraie, Ψ est équivalent à $\forall x \bigvee_i \mathcal{G}_i(\mathtt{x})$.
- On développe de $\mathcal{A}_\mu^{\mathtt{K}_\mu}(\mathtt{x})$ avec les règles $\Longrightarrow_\sqcap$, $\Longrightarrow_\sqcup$, $\Longrightarrow_\exists^{\mathtt{K}_\mu}$, $\Longrightarrow_\forall$, et on retient l'ensemble $\{\mathcal{H}_j(\mathtt{x})\}_{1 \leqslant j \leqslant n}$ des AGraph sans conflit. Suivant la propriété proposée dans la question 8, M est équivalent à $\forall x \bigvee_j \mathcal{H}_j(\mathtt{x})$.

- On ajoute dans chaque AGraph $\mathcal{G}_i(\mathtt{x})$ les $\mathtt{K}_\mu(\mathtt{a})$ pour toute instance $\mathtt{a}$ apparaissant dans $\mathcal{G}_i(\mathtt{x})$.
- On ajoute dans chaque AGraph $\mathcal{H}_j(\mathtt{x})$ les $\mathtt{K}_\psi(\mathtt{a})$ pour toute instance $\mathtt{a}$ apparaissant dans $\mathcal{H}_j(\mathtt{x})$.
- On développe de $\mathcal{G}_i(\mathtt{x}) \cup \mathcal{H}_j(\mathtt{x})$ avec les règles $\Longrightarrow_\sqcap$, $\Longrightarrow_\sqcup$, $\Longrightarrow_\exists^{\mathtt{K}_\psi \sqcap \mathtt{K}_\mu}$, $\Longrightarrow_\forall$, et $\Longrightarrow_\Box$. Ce qui engendre un ensemble $\{\mathcal{Q}_k(\mathtt{x})\}_{1 \leqslant k \leqslant p}$ d'AGraph pouvant contenir des conflits.

À ce stade, si au moins l'un des AGraph $\mathcal{Q}(\mathtt{x})$ ne contient pas de conflit, alors $\Psi \cup \mathrm{M}$ est satisfiable. En effet l'ensemble $\{\mathcal{Q}_k(\mathtt{x})\}_{1 \leqslant k \leqslant p}$ est obtenu à partir de $\mathcal{A}_\psi \cup \mathcal{A}_\mu \cup \{(\mathtt{K}_\psi \sqcap \mathtt{K}_\mu)(\mathtt{x})\}$ en appliquant les règles $\Longrightarrow$. (sans tenir compte des conditions assurant la terminaison), donc d'après la complétude de l'algorithme des tableaux [Baader *et al.*, 2003] et la remarque 2.5.3, $\mathcal{A}_\psi \cup \mathcal{A}_\mu \cup \{(\mathtt{K}_\psi \sqcap \mathtt{K}_\mu)(\mathtt{x})\}$ est satisfiable modulo $\mathtt{K}_\psi \sqcap \mathtt{K}_\mu$. Et donc $\mathcal{A}_\psi \cup \mathcal{A}_\mu$ est satisfiable modulo $\mathtt{K}_\psi \sqcap \mathtt{K}_\mu$, ce qui est équivalent au fait que $\Psi \cup \mathrm{M}$ est satisfiable. Dans ce cas, suivant la propriété proposée dans la question 8, $\Psi \dotplus \mathrm{M}$ est équivalent à $\forall x \bigvee_k \mathcal{Q}_k(\mathtt{x})$

Dans le cas contraire, il faut réparer des conflits

Réparation des conflits

Question 9. *Étant donné un ensemble d'AGraph $\{\mathcal{Q}_k(\mathtt{x})\}_{1 \leqslant k \leqslant p}$ sur lequel aucune règles $\Longrightarrow$. n'est applicable, $\forall x \bigvee_k \mathcal{R}_k(\mathtt{x})$ est-il satisfiable ssi il existe k_0 tel que :*

1. *$\mathcal{R}_{k_0}$ est libre de conflit.*
2. *Pour toute instance $\mathtt{a}$ de $\mathcal{Q}_{k_0}(\mathtt{x})$ il existe k tel que :*

$$\{\mathcal{C} \mid \mathcal{C}(\mathtt{x}) \in \mathcal{Q}_k(\mathtt{x})\} \subseteq \{\mathcal{C} \mid \mathcal{C}(\mathtt{a}) \in \mathcal{Q}_{k_0}(\mathtt{x})\}$$

On peut montrer récursivement que la condition *(2)* est satisfaite par tous les élément de l'ensemble d'AGraph $\{\mathcal{Q}_k(\mathtt{x})\}_{1 \leqslant k \leqslant p}$ obtenu à la fin de l'étape précédente. Par contre, la suppression d'un nœud $\mathtt{D}(\mathtt{a}) \in \mathcal{R}_{k_0}$ peut remettre en cause cette propriété. Pour la préserver il peut être nécessaire de supprimer un nœud $\mathtt{C}(\mathtt{x})$ dans un AGraph $\mathcal{Q}_k$ (par exemple, si avant la suppression de $\mathtt{D}(\mathtt{a})$, on avait $\{\mathtt{C} \mid \mathtt{C}(\mathtt{x}) \in \mathcal{Q}_k(\mathtt{x})\} \subseteq \{\mathtt{C} \mid \mathtt{C}(\mathtt{a}) \in \mathcal{Q}_{k_0}(\mathtt{x})\}$, alors la suppression de $\mathtt{D}(\mathtt{x})$ dans ce $\mathcal{Q}_k$ convient).

Remarque 2.6.3. *La condition* (2) *rend les réparations complexes, puisqu'il faut modifier plusieurs AGraph simultanément contrairement aux réparations effectuées lors de la révision d'ABox (section 2.5.3). Pour simplifier cette étape de réparation on peut remplacer la condition* (2) *par une condition plus forte :*

2'. Pour toute instance $\mathtt{a}$ de $\mathcal{Q}_{k_0}(\mathtt{x})$, $\{\mathcal{C} \mid \mathcal{C}(\mathtt{x}) \in \mathcal{Q}_{k_0}(\mathtt{x})\} \subseteq \{\mathcal{C} \mid \mathcal{C}(\mathtt{a}) \in \mathcal{Q}_{k_0}(\mathtt{x})\}$.

Cette condition n'est pas toujours vérifiée par l'ensemble d'AGraph $\{\mathcal{Q}_k(\mathtt{x})\}_{1 \leqslant k \leqslant p}$ générés à partir de $\mathcal{A}_\psi^{\mathtt{K}_\psi}(\mathtt{x})$ et $\mathcal{A}_\mu^{\mathtt{K}_\mu}(\mathtt{x})$ peuvent ne pas respecter (2'), y compris lorsque $\Psi \cup M$ est cohérente.

Par exemple avec :

$$\Psi = \{\top \sqsubseteq A \sqcup B\}$$
$$M = \{(A \sqcap \neg B)(a), \neg A \sqcap B)(b)\}$$

Pour respecter la condition (2'), des nœuds de $\{\mathcal{Q}_k(x)\}_{1 \leqslant k \leqslant p}$ *doivent être supprimés, ce qui est en contradiction avec* (G*2*).

2.6.4 Exploitation d'un autre moteur d'inférences existant

Le moteur d'inférences PELLET [Sirin *et al.*, 2007], [Pellet, 2011] permet d'évaluer la satisfiabilité de BC exprimées dans les standards OWL [W3C OWL Working Group, 2009] qui correspondent à des LD plus expressives que $\mathcal{ALC}$. On peut donc l'exploiter pour évaluer la satisfiabilité de BC en $\mathcal{ALC}$. D'après [Sirin *et al.*, 2007], PELLET dispose d'un système de « traçage » des déductions effectuées par le moteur d'inférences, ce système est aussi mentionné par [Kalyanpur *et al.*, 2007] qui s'en sert comme outil de débogage d'ontologie.

Un stage de master en cours (celui de Julien Stévenot) a pour objectif de déterminer si ce système de « traçage » permet de reconstituer les AGraph utilisés pour la révision dans l'algorithme présenté en section 2.5.3. Cela permettrait d'exploiter PELLET pour implanter un système de révision. L'avantage de cette solution serait de bénéficier des nombreuses optimisations dont bénéficie PELLET. Ces optimisations rendent les temps de calculs raisonnables en pratique [Horrocks, 1997], bien que la complexité théorique des inférences sur les logiques de descriptions expressives est importante (ExpTime-complet pour $\mathcal{ALC}$). Cependant ces optimisations risquent d'interférer avec l'algorithme de révision, et il sera peut-être nécessaire d'en désactiver quelques unes.

2.7 Fusion contrainte

Supposons qu'un agent dispose de connaissances sur le monde qu'il considère comme étant irréfutables : ce sont ses contraintes d'intégrité (CI). Cet agent reçoit de plusieurs sources des connaissances sur le monde. La conjonction de ces connaissances ne donne pas toujours une BC cohérente avec les CI de l'agent. Il lui faut donc remettre en cause en partie les connaissances reçues afin d'obtenir une BC cohérente avec les CI. L'exemple de fusion présenté en section 2.1.2 suit le même schéma : plusieurs personnes fournissent des connaissances à propos des performances de cyclistes lors d'une course, dont ils cherchent à tirer une BC cohérente avec les connaissances diffusées par la télévision (qui jouent le rôle de CI).

Plusieurs opérateurs permettent de fusionner ces sources de connaissances pour obtenir un résultat cohérent avec les CI. Comme pour la révision des connaissances, des postulats ont été établis pour caractériser ce type d'opérateurs, nous les présentons en section 2.7.1. Puis nous verrons quelques exemples d'opérateurs de fusion contrainte en section 2.7.2

2.7.1 Définition

[Konieczny, 1999] définit un opérateur de fusion contrainte de connaissances comme une fonction $\triangle$ qui prend en entrée un multi-ensemble $\mathcal{B}$ de BC, et une BC IC, appelée contraintes d'intégrité, et qui renvoie une BC $\triangle_{IC}(\mathcal{B})$ qui est le résultat de la fusion des connaissances issues de $\mathcal{B}$ sous contrainte IC. $\mathcal{B}$ représente les connaissances à fusionner, chaque BC de ce multi-ensemble correspond à une source de connaissances.

Définition 2.7.1. *Intuitivement, un multi-ensemble est un ensemble auquel un élément peut appartenir plusieurs fois. Formellement, étant donné un ensemble $\mathcal{U}$, un multi-ensemble $\mathcal{M}$ d'éléments de $\mathcal{U}$ est une fonction de $\mathcal{U}$ dans les entiers naturels $\mathbb{N}$ qui associe à chaque élément de $\mathcal{U}$ la multiplicité de son appartenance à $\mathcal{M}$.*

Lorsque l'on aura besoin de représenter un multi-ensemble, on utilisera la notation similaire à celle des ensembles en ne faisant apparaître que les éléments dont l'ordre de multiplicité est non nul, et en répétant les autres éléments suivant leur ordre de multiplicité. Par exemple, $\{1, 1, 3\}$ représente le multi-ensemble d'entiers contenant 1 deux fois et 3 une fois, l'ordre de multiplicité des autres éléments étant 0.

L'union de deux multi-ensembles $\mathcal{M}_1$ et $\mathcal{M}_2$, notée $\mathcal{M}_1 \cup \mathcal{M}_2$, est un multi-ensemble obtenu en additionnant l'ordre de multiplicité des éléments. Par exemple, $\{1, 1, 3\} \cup \{0, 3, 5\} = \{0, 1, 1, 3, 3, 5\}$.

Pour un multi-ensemble $\mathcal{B}$ de BC, on note $\mathrm{Mods}(\mathcal{B}) = \{\mathrm{Mod}(\Psi) \mid \Psi \in \mathcal{B}\}$ le multi-ensemble formé des ensembles de modèles respectifs des différentes BC de $\mathcal{B}$, en respectant les multiplicités. En particulier, si $\mathcal{B}$ contient deux BC Ψ_1 et Ψ_2 avec les multiplicités respectives n_1 et n_2 et telles que $\mathrm{Mod}(\Psi_1) = \mathrm{Mod}(\Psi_2)$, et qu'aucune autre BC de $\mathcal{B}$ n'a les mêmes modèles, alors $\mathrm{Mod}(\Psi_1)$ aura un ordre de multiplicité de $n_1 + n_2$ dans $\mathrm{Mods}(\mathcal{B})$.

Les connaissances à fusionner sont données dans un multi-ensemble car une connaissance qui est apportée par deux sources aura plus d'importance que si elle n'est apportée que par une source.

[Konieczny, 1999] propose une caractérisation des opérateurs de fusion contrainte par un ensemble de postulats inspirés des postulats AGM de la révision.

Postulats de la fusion contrainte

[Konieczny, 1999] traite de la fusion contrainte en logique propositionnelle finie. Nous proposons ici une reformulation des postulats de [Konieczny, 1999] correspondant à la reformulation

des postulats AGM présentée en section 2.3 :

($\triangle$-0) $\mathrm{Mod}(\triangle_{IC}(\mathcal{B})) \subseteq \mathrm{Mod}(IC)$

($\triangle$-1) Si $\mathrm{Mod}(IC) \neq \emptyset$ alors $\mathrm{Mod}(\triangle_{IC}(\mathcal{B})) \neq \emptyset$

($\triangle$-2) Si $\bigcap_{\Psi \in \mathcal{B}} \mathrm{Mod}(\Psi) \cap \mathrm{Mod}(IC) \neq \emptyset$ alors

$\mathrm{Mod}(\triangle_{IC}(\mathcal{B})) = \bigcap_{\Psi \in \mathcal{B}} \mathrm{Mod}(\Psi) \cap \mathrm{Mod}(IC)$

($\triangle$-3) Si $\mathrm{Mod}(IC_1) = \mathrm{Mod}(IC_2)$ et $\mathrm{Mods}(\mathcal{B}_1) = \mathrm{Mods}(\mathcal{B}_2)$ alors

$\mathrm{Mod}(\triangle_{IC_1}(\mathcal{B}_1)) = \mathrm{Mod}(\triangle_{IC_2}(\mathcal{B}_2))$

($\triangle$-4) Si $\mathrm{Mod}(\Psi_1) \subseteq \mathrm{Mod}(IC)$ et $\mathrm{Mod}(\Psi_2) \subseteq \mathrm{Mod}(IC)$ alors

$\mathrm{Mod}(\triangle_{IC}(\{\Psi_1, \Psi_2\})) \cap \mathrm{Mod}(\Psi_1) \neq \emptyset$ ssi

$\mathrm{Mod}(\triangle_{IC}(\{\Psi_1, \Psi_2\})) \cap \mathrm{Mod}(\Psi_2) \neq \emptyset$

($\triangle$-5) $\mathrm{Mod}(\triangle_{IC}(\mathcal{B}_1)) \cap \mathrm{Mod}(\triangle_{IC}(\mathcal{B}_2)) \subseteq \mathrm{Mod}(\triangle_{IC}(\mathcal{B}_1 \cup \mathcal{B}_2))$

($\triangle$-6) Si $\mathrm{Mod}(\triangle_{IC}(\mathcal{B}_1)) \cap \mathrm{Mod}(\triangle_{IC}(\mathcal{B}_2)) \neq \emptyset$ alors

$\mathrm{Mod}(\triangle_{IC}(\mathcal{B}_1 \cup \mathcal{B}_2)) \subseteq \mathrm{Mod}(\triangle_{IC}(\mathcal{B}_1)) \cap \mathrm{Mod}(\triangle_{IC}(\mathcal{B}_2))$

($\triangle$-7) $\mathrm{Mod}(\triangle_{IC_1}(\mathcal{B})) \cap \mathrm{Mod}(IC_2) \subseteq \mathrm{Mod}(\triangle_{IC_1 \cap IC_2}(\mathcal{B}))$

($\triangle$-8) Si $\mathrm{Mod}(\triangle_{IC_1}(\mathcal{B})) \cap \mathrm{Mod}(IC_2) \neq \emptyset$ alors

$\mathrm{Mod}(\triangle_{IC_1 \cup IC_2}(\mathcal{B})) \subseteq \mathrm{Mod}(\triangle_{IC_1}(\mathcal{B}))$

où IC, IC_1, IC_2, Ψ_1 et Ψ_2 sont des BC telles que $\mathrm{Mod}(\Psi_1) \neq \emptyset$, $\mathrm{Mod}(\Psi_2) \neq \emptyset$ et où $\mathcal{B}$, $\mathcal{B}_1$, et $\mathcal{B}_2$ sont trois multi-ensembles de BC cohérentes.

Remarque 2.7.1. ($\triangle$-8) *n'est pas exactement symétrique de* ($\triangle$-7), $\mathrm{Mod}(IC_2)$ *a été omis de la partie droite de l'inclusion, cela n'a pas d'importance car* $\mathrm{Mod}(\triangle_{IC_1 \cup IC_2}(\mathcal{B})) \subseteq \mathrm{Mod}(IC_2)$ *est conséquence de* ($\triangle$-0) *puisque* $\mathrm{Mod}(IC_1 \cup IC_2) = \mathrm{Mod}(IC_1) \cap \mathrm{Mod}(IC_2) \subseteq \mathrm{Mod}(IC_2)$. *Cette formulation du postulat correspond à celle de [Konieczny, 1999].*

($\triangle$-0) établit que le résultat de la fusion doit satisfaire les contraintes d'intégrité. Ce postulat correspond à (G1) dans le cas de la révision.

($\triangle$-1) établit que si IC est cohérent, alors la fusion des connaissances $\mathcal{B}$ doit donner un résultat cohérent. Ce postulat correspond à (G3) dans le cas de la révision.

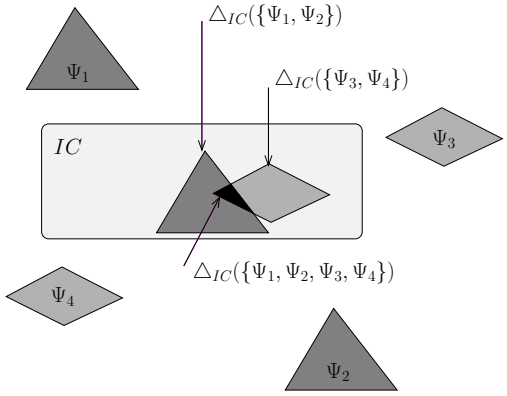
($\triangle$-2) établit que si les connaissances $\mathcal{B}$ sont cohérentes ensemble avec les contraintes d'intégrité IC, la fusion consiste à regrouper ces connaissances, c'est-à-dire à faire l'union des BC : $\bigcap_{\Psi \in \mathcal{B}} \mathrm{Mod}(\Psi) \cap \mathrm{Mod}(IC) = \mathrm{Mod}\left(\bigcup_{\Psi \in \mathcal{B}} \Psi \cup IC\right)$. Intuitivement, ce postulat signifie que s'il n'est pas nécessaire de remettre en cause les connaissances, on ne le fait pas, il correspond au postulat (G2) dans le cas de la révision.

($\triangle$-3) établit l'indépendance à la syntaxe de la fusion. Il correspond au postulat (G4) de la révision.

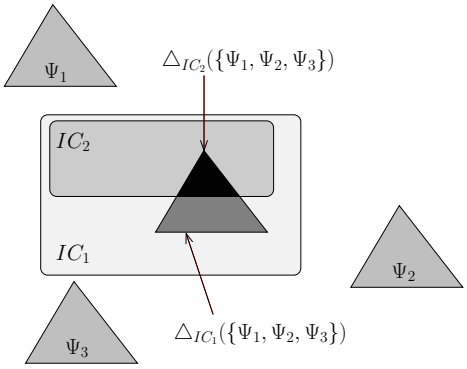
($\triangle$-4) assure l'équité entre les sources de connaissances : si deux BC contenues dans $\mathcal{B}$ sont cohérentes séparément avec les contraintes d'intégrité, alors soit la fusion retient les deux BC, soit elle remet en cause les deux.

Les postulats ($\triangle$-5) à ($\triangle$-8) caractérisent la minimalité des changements apportés aux connaissances de $\mathcal{B}$.

($\triangle$-5) et ($\triangle$-6) établissent que si la fusion de deux multi-ensembles de BC donnent des solutions cohérentes entre elles, alors la combinaison de leur union correspond à la conjonction de ces deux solutions.



($\triangle$-7) et ($\triangle$-8) établissent que si la fusion de $\mathcal{B}$ sous les contraintes IC_1 est cohérente avec les contraintes IC_2, alors la combinaison de $\mathcal{B}$ sous les contraintes $IC_1 \cup IC_2$ est égale à la conjonction de IC_2 et la combinaison de $\mathcal{B}$ sous les contraintes IC_1. Ces postulats correspondent respectivement aux postulats (G5) et (G6) de la révision.



Lien entre la fusion contrainte et la révision des connaissances

La fusion contrainte de connaissances peut être vue comme une généralisation de la révision dans le sens où, étant donné un opérateur de fusion $\triangle$, l'opération $\dot{+}$ définie pour deux bases de connaissances Ψ et M par :

$$\Psi \dot{+} \mathrm{M} = \triangle_{\mathrm{M}}(\{\Psi\}) \tag{2.13}$$

vérifie les postulats de la révision (G·), vus en section 2.3.

2.7.2 Exemple d'opérateur de fusion contrainte

L'opérateur présenté dans cette section est inspiré de l'un des opérateurs présentés dans [Konieczny, 1999]. Il est défini par la minimisation d'une distance entre les modèles des différentes BC. Dans [Konieczny, 1999], cet opérateur est défini en logique propositionnelle, nous étendons cette définition à toute logique telle que définie en préliminaires.

Étant donné une « distance » d sur $\mathcal{U}$, on définit la fonction d^{Σ} qui, à tout couple $(\mathcal{M}, y)$ où $y \in \mathcal{U}$ et $\mathcal{M}$ est un multi-ensemble fini de sous ensembles de $\mathcal{U}$, associe un réel positif $d^{\Sigma}(\mathcal{M}, y)$

tel que :

$$d^{\Sigma}(\mathcal{M}, y) = \sum_{A \in \mathcal{M}} d(A, y)$$

L'opérateur de fusion contrainte $\triangle^{d,\Sigma}$ est défini grâce à d^{Σ} par ses modèles, pour une BC $IC \in 2^{\mathcal{U}}$, et un multi-ensemble fini $\mathcal{B}$ de BC :

$$\begin{aligned} \mathrm{Mod}(\triangle^{d,\Sigma}_{IC}(\mathcal{B})) &= \operatorname*{Min}_{d^{\Sigma}(\mathrm{Mods}(\mathcal{B}),\cdot)} \mathrm{Mod}(IC) \\ &= \{y \in IC \mid d^{\Sigma}(\mathrm{Mods}(\mathcal{B}), y) = d^{\Sigma}(\mathrm{Mods}(\mathcal{B}), IC)\} \end{aligned} \tag{2.14}$$

où $d^{\Sigma}(\mathcal{B}, IC) = \inf_{y \in IC} d^{\Sigma}(\mathcal{B}, y)$.

Proposition 2.7.1. *$\triangle^{d,\Sigma}$ satisfait les postulats* ($\triangle$-0)*,* ($\triangle$-3)*,* ($\triangle$-5)*,* ($\triangle$-6)*,* ($\triangle$-7) *et* ($\triangle$-8)*.*

Démonstration. **($\triangle$-0)** La satisfaction du postulat ($\triangle$-0) est une conséquence directe de la définition de $\triangle^{d,\Sigma}_{IC}(\mathcal{B})$.

($\triangle$-3) La satisfaction de ce postulat est aussi une conséquence de la définition de $\triangle^{d,\Sigma}_{IC}(\mathcal{B})$, celui-ci étant défini à partir des modèles des différentes BC.

($\triangle$-5) et ($\triangle$-6) Ces postulats sont satisfaits lorsque $\mathrm{Mod}(\triangle^{d,\Sigma}_{IC}(\mathcal{B}_1)) \cap \mathrm{Mod}(\triangle^{d,\Sigma}_{IC}(\mathcal{B}_2)) = \emptyset$. Lorsque ce n'est pas le cas, on considère un élément y de $\mathrm{Mod}(\triangle^{d,\Sigma}_{IC}(\mathcal{B}_1)) \cap \mathrm{Mod}(\triangle^{d,\Sigma}_{IC}(\mathcal{B}_2))$,

$$\begin{aligned} & d^{\Sigma}(\mathrm{Mods}(\mathcal{B}_1), \mathrm{Mod}(IC)) + d^{\Sigma}(\mathrm{Mods}(\mathcal{B}_2), \mathrm{Mod}(IC)) \\ &\qquad = d^{\Sigma}(\mathrm{Mods}(\mathcal{B}_1), y) + d^{\Sigma}(\mathrm{Mods}(\mathcal{B}_2), y) \\ &\qquad \geqslant \inf_{z \in \mathrm{Mod}(IC)} \big(d^{\Sigma}(\mathrm{Mods}(\mathcal{B}_1), z) + d^{\Sigma}(\mathrm{Mods}(\mathcal{B}_2), z)\big) \\ &\qquad \geqslant d^{\Sigma}(\mathrm{Mods}(\mathcal{B}_1) \cup \mathrm{Mods}(\mathcal{B}_2), \mathrm{Mod}(IC)) \end{aligned}$$

Cependant

$$\begin{aligned} & d^{\Sigma}(\mathrm{Mods}(\mathcal{B}_1), \mathrm{Mod}(IC)) + d^{\Sigma}(\mathrm{Mods}(\mathcal{B}_2), \mathrm{Mod}(IC)) \\ &\qquad = \inf_{z \in \mathrm{Mod}(IC)} d^{\Sigma}(\mathrm{Mods}(\mathcal{B}_1), z) + \inf_{z \in \mathrm{Mod}(IC)} d^{\Sigma}(\mathrm{Mods}(\mathcal{B}_2), z) \\ &\qquad \leqslant \inf_{z \in \mathrm{Mod}(IC)} \big(d^{\Sigma}(\mathrm{Mods}(\mathcal{B}_1), z) + d^{\Sigma}(\mathrm{Mods}(\mathcal{B}_2), z)\big) \\ &\qquad \leqslant d^{\Sigma}(\mathrm{Mods}(\mathcal{B}_1 \cup \mathcal{B}_2), \mathrm{Mod}(IC)) \end{aligned}$$

Toutes les inégalités sont donc des égalités, en particulier la borne inférieure de $d^{\Sigma}(\mathrm{Mods}(\mathcal{B}_1), z) + d^{\Sigma}(\mathrm{Mods}(\mathcal{B}_2), z)$ pour $z \in IC$ est $d^{\Sigma}(\mathrm{Mods}(\mathcal{B}_1 \cup \mathcal{B}_2), \mathrm{Mod}(IC))$, et comme pour $i = 1, 2$ $d^{\Sigma}(\mathrm{Mods}(\mathcal{B}_i), z) \geqslant d^{\Sigma}(\mathrm{Mods}(\mathcal{B}_i), \mathrm{Mod}(IC))$ cette borne est at-

teinte lorsque $d^{\Sigma}(\mathrm{Mods}(\mathcal{B}_i), z) = d^{\Sigma}(\mathrm{Mods}(\mathcal{B}_i), \mathrm{Mod}(IC))$. Donc $z \in \triangle^{d,\Sigma}_{IC}(\mathcal{B}_1 \cup \mathcal{B}_2)$ ssi $z \in \triangle^{d,\Sigma}_{IC}(\mathcal{B}_1) \cap \triangle^{d,\Sigma}_{IC}(\mathcal{B}_2)$.

($\triangle$-7) et ($\triangle$-8) Comme précédemment, lorsque $\mathrm{Mod}(\triangle^{d,\Sigma}_{IC_1}(\mathcal{B})) \cap \mathrm{Mod}(IC_2) = \emptyset$, les postulats sont satisfaits.

Lorsque ce n'est pas le cas, on considère un élément y de $\mathrm{Mod}(\triangle^{d,\Sigma}_{IC_1}(\mathcal{B})) \cap \mathrm{Mod}(IC_2)$.

$$\begin{aligned}
d^{\Sigma}(\mathrm{Mods}(\mathcal{B}), y) &= d^{\Sigma}(\mathrm{Mods}(\mathcal{B}), \mathrm{Mod}(IC_1)) \\
&= \inf_{z \in \mathrm{Mod}(IC_1)} d^{\Sigma}(\mathrm{Mods}(\mathcal{B}), z) \\
&\leqslant \inf_{z \in \mathrm{Mod}(IC_1) \cap \mathrm{Mod}(IC_2)} d^{\Sigma}(\mathrm{Mods}(\mathcal{B}), z) \\
&\leqslant d^{\Sigma}(\mathrm{Mods}(\mathcal{B}), \mathrm{Mod}(IC_1) \cap \mathrm{Mod}(IC_2)) \\
&\leqslant d^{\Sigma}(\mathrm{Mods}(\mathcal{B}), y)
\end{aligned}$$

Donc

$$\begin{aligned}
d^{\Sigma}(\mathrm{Mods}(\mathcal{B}), \mathrm{Mod}(IC_1)) &= d^{\Sigma}(\mathrm{Mods}(\mathcal{B}), y) \\
&= d^{\Sigma}(\mathrm{Mods}(\mathcal{B}), \mathrm{Mod}(IC_1) \cap \mathrm{Mod}(IC_2)) \\
&= d^{\Sigma}(\mathrm{Mods}(\mathcal{B}), \mathrm{Mod}(IC_1 \cup IC_2))
\end{aligned}$$

Et

$$\begin{aligned}
\mathrm{Mod}(\triangle^{d,\Sigma}_{IC_1}(\mathcal{B}_1)) \cap \mathrm{Mod}(IC_2) &= \{y \in \mathrm{Mod}(IC_1 \cup IC_2) \mid d^{\Sigma}(\mathrm{Mods}(\mathcal{B}), y) \\
&= d^{\Sigma}(\mathrm{Mods}(\mathcal{B}), \mathrm{Mod}(IC_1))\} \\
&= \mathrm{Mod}(\triangle^{d,\Sigma}_{IC_1 \cup IC_2}(\mathcal{B}_1))
\end{aligned}$$

□

L'opérateur $\triangle^{d,\Sigma}$ ne satisfait pas toujours les postulats ($\triangle$-1), ($\triangle$-2) et ($\triangle$-4). Comme pour le postulat (G2), le postulat ($\triangle$-2) est vérifié si les modèles des BC forment des ensembles fermés pour d, c'est-à-dire que pour une BC Ψ, $\overline{\mathrm{Mod}(\Psi)} = \mathrm{Mod}(\Psi)$. Pour que les postulats ($\triangle$-1) et ($\triangle$-4) soient vérifiés, il faut que la distance entre les modèles de deux BC soit atteinte, c'est-à-dire que pour deux BC cohérentes Ψ et M, il existe $u \in \mathrm{Mod}(\Psi)$ et $v \in \mathrm{Mod}(\mathrm{M})$ tel que $d(\mathrm{Mod}(\Psi), \mathrm{Mod}(\mathrm{M})) = d(u, v)$. Comme pour la révision (satisfaction du postulat (G3)), cela peut être assuré en faisant des hypothèses sur les ensembles de modèles et sur d, par exemple que d prenne des valeurs discrètes, comme c'est le cas en logique propositionnelle. Une autre possibilité serait de considérer la la fusion floue, comme pour la révision en section 2.3.4.

Proposition 2.7.2. *Si pour tout $\Psi \in \mathcal{B}$, $\mathrm{Mod}(\Psi)$ est fermé pour d, alors $\triangle^{d,\Sigma}$ satisfait le postulat ($\triangle$-2).*

Démonstration. Supposons que $\bigcap_{\Psi\in\mathcal{B}} \mathrm{Mod}(\Psi) \cap \mathrm{Mod}(IC) \neq \emptyset$.

Soit $y \in \bigcap_{\Psi\in\mathcal{B}} \mathrm{Mod}(\Psi) \cap \mathrm{Mod}(IC)$, $y \in \mathrm{Mod}(\Psi)$ et donc $d(\mathrm{Mod}(\Psi), y) = 0$ pour tout $\Psi \in \mathcal{B}$ donc $d^{\Sigma}(\mathrm{Mods}(\mathcal{B}), y) = 0$. Puisque $y \in \mathrm{Mod}(IC)$, $d^{\Sigma}(\mathrm{Mods}(\mathcal{B}), \mathrm{Mod}(IC)) = 0$ et $y \in \triangle^{d,\Sigma}_{IC}(\mathcal{B})$.

Réciproquement, puisque $d^{\Sigma}(\mathrm{Mods}(\mathcal{B}), \mathrm{Mod}(IC)) = 0$, pour $y \in \triangle^{d,\Sigma}_{IC}(\mathcal{B})$ $d^{\Sigma}(\mathrm{Mods}(\mathcal{B}), y) = 0$ et donc $d(\mathrm{Mod}(\Psi), y) = 0$ pour tout $\Psi \in \mathcal{B}$ ($d(\mathrm{Mod}(\Psi), y) \geqslant 0$ pour tout Ψ). Puisque l'on suppose que $\mathrm{Mod}(\Psi)$ est fermé cela implique que $y \in \mathrm{Mod}(\Psi)$, et donc que $y \in \bigcap_{\Psi\in\mathcal{B}} \mathrm{Mod}(\Psi)$ et par définition de $\triangle^{d,\Sigma}_{IC}(\mathcal{B})$ on obtient que $y \in IC$. □

Proposition 2.7.3. *Si la fonction $y \mapsto d^{\Sigma}(\mathrm{Mods}(\mathcal{B}), y)$ admet un minimum sur $\mathrm{Mod}(IC)$, alors $\triangle^{d,\Sigma}$ satisfait le postulat* ($\triangle$-1).

Démonstration. Si cette fonction admet un minimum sur $\mathrm{Mod}(IC)$, $\underset{d^{\Sigma}(\mathrm{Mods}(\mathcal{B}),\cdot)}{\mathrm{Min}} \mathrm{Mod}(IC) \neq \emptyset$, donc d'après la définition de $\triangle^{d,\Sigma}$ (équation (2.14), page73), $\mathrm{Mod}(\triangle^{d,\Sigma}_{IC}(\mathcal{B})) \neq \emptyset$. □

Proposition 2.7.4. *Si d est symétrique, c'est-à-dire que pour tout $x, y \in \mathcal{U}$, $d(x, y) = d(y, x)$, et les distances entre modèles sont atteintes, alors $\triangle^{d,\Sigma}$ satisfait aussi le postulat* ($\triangle$-4).

Démonstration. On suppose ici que d est symétrique et on considère trois BC : IC, Ψ_1 et Ψ_2 telles que $\mathrm{Mod}(\Psi_1) \neq \emptyset$, $\mathrm{Mod}(\Psi_2) \neq \emptyset$, $\mathrm{Mod}(\Psi_1) \subseteq \mathrm{Mod}(IC)$ et $\mathrm{Mod}(\Psi_2) \subseteq \mathrm{Mod}(IC)$.

Si $\mathrm{Mod}(\triangle^{d,\Sigma}_{IC}(\{\Psi_1, \Psi_2\})) \cap \mathrm{Mod}(\Psi_1) \neq \emptyset$, soit u_1 un élément de cet ensemble.

$$\begin{aligned} d^{\Sigma}(\mathrm{Mods}(\{\Psi_1, \Psi_2\}), \mathrm{Mod}(IC)) &= d^{\Sigma}(\mathrm{Mods}(\{\Psi_1, \Psi_2\}), u_1) \\ &= d(\mathrm{Mod}(\Psi_1), u_1) + d(\mathrm{Mod}(\Psi_2), u_1) \end{aligned}$$

comme $u_1 \in \mathrm{Mod}(\Psi_1)$, $d(\mathrm{Mod}(\Psi_1), u_1) = 0$, donc $d^{\Sigma}(\mathrm{Mods}(\{\Psi_1, \Psi_2\}), \mathrm{Mod}(IC)) = d(\mathrm{Mod}(\Psi_2), u_1)$. Puisque l'on suppose que les distances entre ensembles de modèles sont atteintes et que $\mathrm{Mod}(\Psi_2) \neq \emptyset$, il existe $u_2 \in \mathrm{Mod}(\Psi_2)$ tel que $d(u_2, u_1) = d(\mathrm{Mod}(\Psi_2), u_1) = d^{\Sigma}(\mathrm{Mods}(\{\Psi_1, \Psi_2\}), \mathrm{Mod}(IC))$.

$$\begin{aligned} d^{\Sigma}(\mathrm{Mods}(\{\Psi_1, \Psi_2\}), \mathrm{Mod}(IC)) &= d(u_2, u_1) \\ &= d(u_1, u_2) \\ &\geqslant d(\mathrm{Mod}(\Psi_1), u_2) = d^{\Sigma}(\mathrm{Mods}(\{\Psi_1, \Psi_2\}), u_2) \end{aligned}$$

comme $u_2 \in \mathrm{Mod}(IC)$, $d^{\Sigma}(\mathrm{Mods}(\{\Psi_1, \Psi_2\}), u_2) \geqslant d^{\Sigma}(\mathrm{Mods}(\{\Psi_1, \Psi_2\}), \mathrm{Mod}(IC))$, il y a donc égalité et $u_2 \in \mathrm{Mod}(\triangle^{d,\Sigma}_{IC}(\{\Psi_1, \Psi_2\}))$ ce qui entraîne que $\mathrm{Mod}(\triangle^{d,\Sigma}_{IC}(\{\Psi_1, \Psi_2\})) \cap \mathrm{Mod}(\Psi_2) \neq \emptyset$.

L'autre implication est symétrique. □

2.7.3 Calcul de la fusion

Fusion dans un formalisme numérique affine

Dans [Cojan et Lieber, 2009a], nous étendons l'opérateur de révision défini en section 2.4.2, à un opérateur de fusion contrainte dont le calcul est aussi réductible à un problème de program-

mation linéaire. Pour cela on suppose que :

- $\mathcal{U} = I_1 \times \ldots \times I_n$ où I_i est un intervalle de $\mathbb{Z}$ ou de $\mathbb{R}$.
- Pour $x, y \in \mathcal{U}$ $d(x,y) = \sum_{i=1}^{n} w_i |y_i - x_i|$.
- IC est constitué d'un nombre finie de contraintes linéaires.
- $\mathcal{B} = \{\Psi_1, \ldots, \Psi_p\}$ et chaque Ψ_j est aussi constitué d'un nombre fini de contraintes linéaires.

Suivant la définition de $\triangle^{d,\Sigma}$ (équation (2.14), page 73), le calcul de $\mathrm{Mod}(\triangle^{d,\Sigma}_{IC}(\mathcal{B}))$ se réduit à la minimisation de la fonction :

$$(y, x^1, \ldots, x^p) \mapsto \sum_{j=1}^{p} \sum_{i=1}^{n} w |y_i - x_i^j|$$

pour $(y, x^1, \ldots, x^p) \in \mathrm{Mod}(IC) \times \mathrm{Mod}(\Psi_1) \times \ldots \times \mathrm{Mod}(\Psi_p)$. Ce problème de minimisation est réductible à un problème de programmation linéaire par la technique présentée dans l'exemple 2.4.1 (page 36).

Comme nous l'avons vu en section 2.4.2 (page 38), les ensembles considérés dans le formalisme plus général donné dans [Schwind, 2010] peuvent être décomposés en unions finies de simplexes. De la même manière que pour le calcul de la minimisation de d entre disjonctions de simplexes peut être effectué à partir du calcul de la minimisation entre simplexes, le calcul de la minimisation de d^{Σ} entre des disjonctions de simplexes peut être effectue à partir du calcul de la minimisation sur des simplexes. Le calcul de la fusion contrainte dans le formalisme utilisé par [Schwind, 2010] est alors réduit à un problème de programmation linéaire.

Remarque 2.7.2. *Le calcul de la fusion contrainte peut aussi être réduite à un problème de programmation linéaire pour des opérateurs $\triangle^{d,\Sigma}$ définis grâce à d'autres distances, par exemple, celle de l'exemple 2.4.2 (page 36).*

Fusion dans $\mathcal{ALC}$

L'algorithme de calcul de la révision de BC exprimées dans $\mathcal{ALC}$ est encore dans un état préliminaire, son extension à la fusion est à l'état de perspective. Cependant le principe de l'algorithme s'étend facilement à la fusion.

Dans le cas de la révision d'une BC Ψ par une autre BC M, le principe de l'algorithme est de développer l'algorithme des tableaux sur la BC $\Psi \cup \mathrm{M}$. Si aucun conflit n'est détecté, d'après la complétude de l'algorithme des tableaux $\Psi \cup \mathrm{M}$ est cohérent, et en accord avec le postulat (G2) on peut renvoyer $\Psi \dotplus \mathrm{M} = \Psi \cup \mathrm{M}$. Si des conflits sont détectés, ceux-ci sont « réparés » en remettant en cause en partie Ψ. Le choix des réparations est guidé par le coût des modifications apportées à Ψ.

Dans le cas de la fusion d'un multi-ensemble $\mathcal{B}$ de BC sous contraintes d'une BC IC, on développe l'algorithme des tableaux sur la BC $\left(\bigcup_{\Psi \in \mathcal{B}} \Psi\right) \cup IC$. Si aucun conflit n'est détecté, toujours d'après la complétude de l'algorithme des tableaux on sait que cette BC est cohérente en en accord avec le postulat ($\triangle$-2), on peut poser $\triangle_{\mathcal{B}}(IC) = \left(\bigcup_{\Psi \in \mathcal{B}} \Psi\right) \cup IC$. Sinon, les conflits

détectés sont « réparés » de la même manière que pour la révision, en remettant en cause en partie certains $\Psi \in \mathcal{B}$. Le choix des réparations est guidée par le coût des modifications apportées aux différents $\Psi \in \mathcal{B}$, la fonction d'agrégation fournit un coût global pour les réparations apportées à l'ensemble des Ψ.

2.7.4 Choix d'une fonction d'agrégation

[Konieczny, 1999] propose d'autres opérateurs de fusion définis grâce à des distances. Ces opérateurs sont définis à partir d'une « distance » sur l'espace des interprétations et d'une fonction d'agrégation qui agrège les distances vis-à-vis des différentes sources. Pour l'opérateur $\triangle^{d,\Sigma}$, puisque $d^{\Sigma}(\mathcal{M}, \cdot)$ est définie comme la somme des $d(A, \cdot)$ pour $A \in \mathcal{M}$, la fonction d'agrégation choisie est $(x_1, \ldots, x_k) \mapsto \sum_{i=1}^{k} x_i$. D'autres fonctions d'agrégation sont proposées dans [Konieczny, 1999], le choix d'une distance et d'une fonction d'agrégation détermine le comportement de l'opérateur de fusion.

Remarque 2.7.3. *[Konieczny* et al.*, 2004] donne une forme plus générale d'opérateur de fusion défini à l'aide d'une « distance » sur l'espace des interprétations. Elle fait intervenir deux fonctions d'agrégations. La première est qualifiée d'« intra-source », elle agrège la distance vis-à-vis des différentes formules d'une BC, ce qui permet de définir des opérateurs de fusion syntaxiques. Étant donné que l'on ne distingue pas les modèles des différentes formules, on l'ignore ici. La fonction d'agrégation qui nous intéresse est la seconde, qualifiée d'« inter-source » puisqu'elle elle agrège les distances vis-à-vis des différentes sources.*

[Konieczny, 1999] établit des propriétés pour plusieurs opérateurs définis de cette manière. Comme nous le verrons en section 3.4.5, ces propriétés ne correspondent pas à des comportements attendus pour le RÀPC. Une perspective serait de préciser les propriétés attendues d'un opérateur de fusion pour le RÀPC et de définir des opérateurs qui satisfont ces propriétés.

Et dans la perspective du calcul, il serait intéressant d'établir quelles associations entre une distance et une fonction d'agrégation engendrent un opérateur de fusion dont le calcul soit réductible à un problème de programmation linéaire.

Chapitre 3

L'adaptation par révision

Le raisonnement à partir de cas (RÀPC) est un mode de raisonnement issu des sciences cognitives [Riesbeck et Schank, 1989]. Il s'appuie sur l'observation du fait que lorsque nous sommes confrontés à un problème nous nous inspirons parfois d'expériences acquises lors de confrontations à des problèmes similaires. Le RÀPC modélise le raisonnement consistant à exploiter les connaissances constituées par l'expérience de résolutions de problèmes pour résoudre de nouveaux problèmes. Le terme « cas » fait référence à ce type d'expérience.

Afin de situer la question de l'adaptation nous proposons une présentation générale du RÀPC en section 3.1. Une grande diversité de formes d'adaptation sont décrites dans la littérature sans qu'il y ait de cadre formel commun pour leur définition. C'est l'objectif de la définition de l'adaptation par la révision en section 3.2. Nous exprimons en section 3.3 plusieurs approches d'adaptation sous forme d'adaptation par la révision. En section 3.4, nous généralisons l'adaptation par révision en définissant une approche de combinaison de cas à l'aide d'un opérateur de fusion contrainte de connaissances.

3.1 Préliminaires

3.1.1 Le RÀPC : principes généraux

Le principe du RÀPC est d'exploiter l'expérience de la résolution de problèmes particuliers pour résoudre de nouveaux problèmes du même domaine. Cette expérience est représentée par une *base de cas*, dont les éléments, les *cas sources*, donnent la description d'un problème et de sa résolution, c'est-à-dire d'une solution et parfois d'informations complémentaires, comme une description du raisonnement ayant abouti à cette solution [Kolodner, 1993]. L'objectif est de proposer une solution au nouveau problème, qui peut aussi être accompagnée de justifications comme pour les cas sources. Le problème à résoudre est aussi représenté par un cas, le *cas cible* dont on ne connaît pas la solution.

La résolution d'un cas cible par un système de RÀPC peut être décomposée en deux étapes principales : la *remémoration* d'un (ou plusieurs) cas sources jugés utiles pour la résolution du cas

cible, et l'*adaptation* du cas source remémoré afin d'obtenir une solution convenant au cas cible. Une solution au cas cible peut aussi être construite à partir de plusieurs cas sources, on parle alors de *combinaison de cas* (nous présentons cette opération en section 3.4). Suivant les applications, d'autres opérations peuvent être effectuées [Riesbeck et Schank, 1989], [Aamodt et Plaza, 1994], par exemple la solution proposée peut être *corrigée* (révisée selon le vocabulaire de la communauté RÀPC) si, après évaluation de la solution, des modifications sont demandées. Si la solution obtenue est jugée intéressante, le cas obtenu peut être *mémorisé*, c'est-à-dire ajouté à la base de cas.

On peut distinguer deux utilisations du RÀPC : en tant qu'heuristique permettant d'accélérer la résolution de problèmes, ou en tant que méthode de résolution hypothétique en l'absence de méthode sûre pour obtenir une solution. Cette distinction est reprise de celle qui est faite entre analogie *heuristique* et analogie *recours* [Coulon *et al.*, 1990]. L'utilisation heuristique du RÀPC (RÀPC *heuristique*) concerne les domaines pour lesquels on dispose de connaissances complètes pour identifier les solutions à un problèmes. Cependant, en l'absence de méthode efficace pour calculer ces solutions, la résolution d'un problème peut être accélérée en partant d'une solution connue pour un problème proche. La base de cas est alors une sorte d'« échantillonnage » de solutions à des problèmes rencontrés. Le RÀPC heuristique peut aussi servir à obtenir une solution non optimale si le calcul d'une solution optimale est trop coûteuse. Le second type d'utilisation du RÀPC, que nous appellerons RÀPC *hypothétique* concerne les domaines où l'on ne dispose pas de connaissances suffisantes pour obtenir une solution sûre. Par exemple lorsqu'il y a une part de subjectif dans l'évaluation des solutions : en cuisine, il n'y a pas de connaissances exactes permettant de prédire si un plat plaira, en revanche les recettes apportent l'expérience de plats déjà préparés. L'objectif du RÀPC dans cette situation est de proposer des solutions candidates, les meilleures possibles, en fonction des connaissances disponibles, en particulier la connaissance de solutions validées pour des problèmes déjà rencontrés. La remémoration est nécessaire pour les utilisations heuristiques et hypothétiques, l'adaptation n'est pas essentielle pour l'utilisation heuristique, puisqu'une méthode exacte prend le relais pour le calcul d'une solution. Pour une utilisation hypothétique, en revanche, l'adaptation est nécessaire pour obtenir des solutions candidates et la qualité de ces solutions dépend des capacités d'adaptation du système.

Nous nous intéressons dans cette thèse au RÀPC hypothétique et, particulièrement, à l'adaptation.

3.1.2 Connaissances exploitées par le RÀPC

Types de connaissances exploitées

Le RÀPC exploite principalement les expériences représentées par les cas sources. Cependant d'autres connaissances complémentaires peuvent intervenir lors de la résolution d'un problème. En nous appuyant sur [Richter, 1998], nous identifions quatre principaux types de connaissances

exploitées dans différentes étapes du RÀPC : la base de cas, les connaissances du domaine, les connaissances d'adaptation et la similarité entre cas.

Base de cas. En plus des connaissances apportées par les cas individuellement, la base de cas peut contenir d'autres connaissances à travers son organisation.

Les cas peuvent être hiérarchisés par niveau d'abstraction [Branting et Aha, 1995], un cas particulier traitant d'un problème précis peut être abstrait en un cas moins précis mais couvrant davantage de problèmes. Cette structure est utile pour la remémoration : on cherche d'abord un cas abstrait proche du problème cible puis on affine la solution en cherchant parmi les cas concrets couverts par le cas abstrait retenu. Les cas abstraits permettent aussi d'exprimer des connaissances complémentaires aux cas (mesure de similarité et connaissances guidant l'adaptation) à un niveau plus général [Bergmann et Wilke, 1996].

Les cas peuvent aussi être organisés de façon modulaire, un cas complexe pouvant être décomposé en plusieurs sous-cas apportant chacun une solution à une partie du problème [Smyth *et al.*, 2001]. La réutilisation de plusieurs solutions partielles fait appel à la combinaison de cas : il faut adapter ces solutions partielles au cas cible mais aussi les rendre compatibles entre elles. Nous étudierons ces questions en section 3.4.

Connaissances du domaine (`CD`). Elles établissent des propriétés générales du domaine, indépendamment des situations particulières. Typiquement, elles constituent une ontologie du domaine reliant les concepts représentés. Par exemple, une ontologie de la cuisine établira que le poulet et le bœuf sont des viandes et qu'une recette végétarienne est une recette ne contenant pas de viande. Ces connaissances sont complémentaires aux cas, elles complètent leur description en impliquant des connaissances supplémentaires les concernant. Par exemple, les caractéristiques diététiques d'une recette peuvent être déduites de la liste des ingrédients qui y figurent et des connaissances du domaine qui donnent la composition diététique des différents ingrédients.

Les connaissances du domaine guident aussi la résolution d'un cas source en établissant des contraintes que doivent vérifier les solutions candidates. Par exemple, si on demande une recette végétarienne, les solutions proposées ne devront pas contenir de viande, et il faudra adapter les recettes remémorées si elles en contiennent. Dans le cadre de l'adaptation, on qualifie ces connaissances de *statiques* car elles contraignent le résultat de l'adaptation, indépendamment du changement apporté au cas source.

Connaissances d'adaptation. Les connaissances d'adaptation traitent des modifications à apporter à un cas source. On qualifie ces connaissances de connaissances *dynamiques* car elles portent sur les variations entre cas et guident les modifications à apporter au cas source pour obtenir une solution au cas cible. Différents types de connaissances d'adaptation sont décrites dans la littérature, nous en présentons quelques unes en section 3.1.4.

En l'absence de méthode exacte de résolution, ce sont ces connaissances d'adaptation qui permettent d'obtenir des solutions candidates au problème posé. Ces connaissances sont plus

ou moins fiables dans le sens où il n'est pas certain que la solution candidate obtenue après leur application à une solution d'un cas source soit satisfaisante. La notion d'*effort d'adaptation* représente la confiance accordée aux modifications indiquées par une connaissance d'adaptation. Lorsque cet effort est mesuré par une valeur numérique, on parlera de *coût d'adaptation*. Lors de l'adaptation d'un cas source, on cherchera à minimiser l'effort d'adaptation afin de maximiser les chances d'obtenir une solution candidate satisfaisante.

Similarité entre cas. L'évaluation de la similarité entre cas est utilisée lors de la remémoration. Le cas source remémoré est celui qui est le plus similaire au cas cible. Lorsqu'elle est suivie par l'adaptation, le but de la remémoration est de fournir le cas source qui pourra le mieux être adapté, c'est-à-dire pour lequel l'adaptation fournira la solution candidate ayant le plus de chances d'être satisfaisante. La similarité doit donc refléter la capacité d'adaptation des cas sources vis-à-vis du cas cible. C'est le principe de la remémoration guidée par l'adaptation [Smyth et Keane, 1998].

La similarité entre cas est donc déduite de l'effort d'adaptation associée aux connaissances d'adaptation. Pour des raisons d'efficacité, la similarité peut n'être qu'une approximation de la capacité d'adaptation, et par exemple n'être exprimée qu'au niveau des problèmes.

3.1.3 Représentation des connaissances

On se place dans un domaine dans lequel on applique le RÀPC pour la résolution de problèmes. On note $\mathcal{U}_{\texttt{pb}}$ l'ensemble des problèmes pouvant être rencontrés dans ce domaine, et $\mathcal{U}_{\texttt{sol}}$ l'ensemble des solutions qui peuvent y être apportées. Un cas associant une solution $\texttt{sol} \in \mathcal{U}_{\texttt{sol}}$ à un problème $\texttt{pb} \in \mathcal{U}_{\texttt{pb}}$ représente l'élément $(\texttt{pb}, \texttt{sol}) \in \mathcal{U} = \mathcal{U}_{\texttt{pb}} \times \mathcal{U}_{\texttt{sol}}$. Un tel cas sera dit ponctuel. Plus généralement, un cas traite d'un ensemble de problèmes auxquels il associe des solutions, il représente un ensemble $\texttt{Cas} \subseteq \mathcal{U}$, c'est-à-dire l'ensemble de couples $(\texttt{pb}, \texttt{sol}) \in \texttt{Cas}$. On peut supposer que $\texttt{Cas}$ peut être mis sous la forme $\texttt{Cas} = \texttt{Pb} \times \texttt{Sol}$ où $\texttt{Pb} \subseteq \mathcal{U}_{\texttt{pb}}$ est un ensemble de problèmes et $\texttt{Sol} \subseteq \mathcal{U}_{\texttt{sol}}$ est un ensemble de solutions. Un tel cas signifie que pour tout problème $\texttt{pb} \in \texttt{Pb}$ et toute solution $\texttt{sol} \in \texttt{Sol}$, $\texttt{sol}$ est une solution à $\texttt{pb}$.

Remarque 3.1.1. *Souvent, une hypothèse différente est faite pour la signification d'un cas* $\texttt{Cas} = \texttt{Pb} \times \texttt{Sol}$ *non ponctuel : « pour tout problème* $\texttt{pb} \in \texttt{Pb}$*,* $\texttt{Sol}$ *contient une solution à* $\texttt{pb}$ *». Le fait que* $\texttt{Cas}$ *associe plusieurs éléments de* $\mathcal{U}_{\texttt{sol}}$ *à un problème* $\texttt{pb} \in \texttt{Pb}$ *correspond à une incertitude sur ce qui pourrait être une solution possible de* $\texttt{pb}$*. Nous verrons en section 3.2.4, que cette hypothèse correspond à un critère de changement minimal pour l'adaptation légèrement différent.*

Les connaissances du domaine précisent les connaissances apportées par les cas. Elles correspondent à une restriction des couples problèmes-solutions possibles et peuvent donc être représentées par un sous-ensemble $\texttt{CD}$ de $\mathcal{U}$: $(\texttt{pb}, \texttt{sol}) \notin \texttt{CD}$ signifie soit que le problème $\texttt{pb}$ ou la solution $\texttt{sol}$ sont impossibles soit que l'affirmation « $\texttt{pb}$ a pour solution $\texttt{sol}$ » est fausse. Comme vu plus haut, la similarité entre cas est calculée à partir de l'effort d'adaptation associé

aux transformations représentées par les connaissances d'adaptation. En section 3.3 nous montrons comment certains types de connaissances d'adaptation peuvent être caractérisées par une « distance » sur $\mathcal{U}$, suivant le principe de la remémoration guidée par l'adaptation, la similarité entre cas est aussi calculée à partie de cette « distance ».

Remarque 3.1.2. *La distinction de l'ensemble des problèmes de l'ensemble des solutions n'est pas toujours pertinente, elle suppose que le problème traité soit déterminé pour tout cas source. Dans certains domaines cela n'a pas de sens. Par exemple dans le cadre du projet* TAAABLE *dont le but est de proposer des recettes de cuisine en réponse à des requêtes, les cas sources correspondent à des recettes de cuisine et ne comportent pas de partie problème. En effet, les requêtes consistent en des contraintes sur la recette désirée, par exemple une liste d'ingrédients à utiliser ou au contraire à proscrire, des régimes à respecter, etc. La réponse à une requête consiste à décrire une recette satisfaisant ces contraintes. Les requêtes sont donc en quelque sorte des recettes non abouties et elles sont exprimées dans le même formalisme. De plus, une recette peut répondre à plusieurs requêtes et on peut supposer que la satisfaction qu'elle procure ne dépend pas de la requête posée. Il est donc inutile dans cette application de distinguer une partie problème d'une partie solution, un cas cible représente une requête, soit une description de recette incomplète et le but du* RÀPC *est de compléter cette description en s'inspirant des recettes de la base de cas.*

Formalismes de représentation des connaissances utilisés en RÀPC

[Kolodner, 1993] relève plusieurs formalismes de représentation courants en RÀPC : les représentations attributs-valeurs simples, les représentations type « objet » et les représentations logiques. On détaille dans cette section ces types de représentation.

Cette liste n'est pas exhaustive, par exemple le RÀPC textuel (*textual case-based reasoning*) [Weber *et al.*, 2005] exploite des cas représentés textuellement.

Formalisme attributs-valeurs simples. Les formalismes attributs-valeurs simples, présentés en section 2.4.1, permettent de représenter une situation ou l'état d'un système par les valeurs d'un ensemble de paramètres (les attributs). Les problèmes et leurs solutions sont décrits par des n-uplets de valeurs, résoudre un cas consiste à trouver les valeurs manquantes. C'est le formalisme utilisé pour le raisonnement sur les quantités d'ingrédients dans TAAABLE, section 4.2, où les cas sont des recettes de cuisine et chaque attribut correspond à la quantité d'un type d'ingrédient (la préparation est représentée dans TAAABLE par un autre formalisme). Les connaissances du domaine sont exprimées par des contraintes numériques sur les valeurs des attributs, par exemple la quantité de viande est égale à la somme des quantités des différents ingrédients classés comme viande. Ce formalisme permet de représenter des connaissances numériques. L'expressivité est cependant limitée, le nombre fixe d'attributs le rend peu pratique pour représenter des domaines complexes.

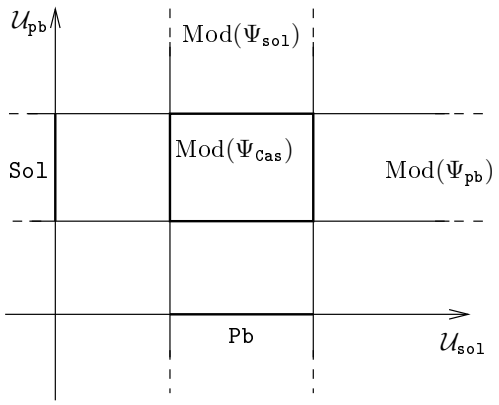
Formalismes objet. Les formalismes objets permettent de structurer la représentation des connaissances. Un cas est représenté par un objet dont les attributs peuvent être des valeurs simples ou d'autres objets. Cela permet d'établir des relations entre objets et de décrire des cas complexes. Les connaissances du domaine décrivent des classes d'objets et leurs propriétés. Des relations de généralisation-spécialisation entre classes permettent d'introduire plusieurs niveaux de généralisation dans la base de cas. Les MOP présentés dans [Riesbeck et Schank, 1989] et les *Frames* dont [Kolodner, 1993] fait mention sont proches des formalismes objets.

Formalismes logiques. Dans les formalismes logiques, les cas et les connaissances du domaine sont représentées par des bases de connaissances exprimées dans un langage $\mathcal{L}$ dont les interprétations sont les éléments de $\mathcal{U}$. Les problèmes sont décrits dans un langage $\mathcal{L}_{\texttt{pb}} \subseteq \mathcal{L}$, les solutions sont décrites dans un langage $\mathcal{L}_{\texttt{sol}} \subseteq \mathcal{L}$. Un cas est une BC $\Psi_{\texttt{Cas}} = \Psi_{\texttt{pb}} \cup \Psi_{\texttt{sol}}$ où $\Psi_{\texttt{pb}}$ et $\Psi_{\texttt{sol}}$ sont des bases de connaissances exprimées respectivement dans $\mathcal{L}_{\texttt{pb}}$ et $\mathcal{L}_{\texttt{sol}}$. Les modèles de ces BC sont des éléments de $\mathcal{U}$, $\Psi_{\texttt{pb}}$ représente un ensemble de problèmes $\texttt{Pb} \subseteq \mathcal{U}_{\texttt{pb}}$ sans spécifier de solution, $\Psi_{\texttt{sol}}$ représente un ensemble de solutions $\texttt{Sol} \subseteq \mathcal{U}_{\texttt{sol}}$ sans spécifier de problèmes, le cas représenté par $\Psi_{\texttt{Cas}}$ associe les solutions $\texttt{Sol}$ aux problèmes $\texttt{Pb}$:

$$\mathrm{Mod}(\Psi_{\texttt{pb}}) = \texttt{Pb} \times \mathcal{U}_{\texttt{sol}}$$
$$\mathrm{Mod}(\Psi_{\texttt{sol}}) = \mathcal{U}_{\texttt{pb}} \times \texttt{Sol}$$
$$\mathrm{Mod}(\Psi_{\texttt{Cas}}) = \mathrm{Mod}(\Psi_{\texttt{pb}} \cup \Psi_{\texttt{sol}})$$

Or $\mathrm{Mod}(\Psi_{\texttt{pb}} \cup \Psi_{\texttt{sol}}) = \mathrm{Mod}(\Psi_{\texttt{pb}}) \cap \mathrm{Mod}(\Psi_{\texttt{sol}})$ (équation (1.1), page 6), donc

$$\mathrm{Mod}(\Psi_{\texttt{Cas}}) = \texttt{Pb} \times \texttt{Sol} = \texttt{Cas}$$



Les connaissances du domaine peuvent exprimer des dépendances entre problèmes et solutions, $\mathcal{L}$ peut donc contenir des formules φ telles que $\mathrm{Mod}(\varphi)$ ne peut pas se mettre sous la forme $\texttt{Pb} \times \texttt{Sol}$ avec $\texttt{Pb} \subseteq \mathcal{U}_{\texttt{pb}}$ et $\texttt{Sol} \subseteq \mathcal{U}_{\texttt{sol}}$. On peut donc avoir $\mathcal{L}_{\texttt{pb}} \cup \mathcal{L}_{\texttt{sol}} \subsetneq \mathcal{L}$.

Remarque 3.1.3. *Comme nous avons vu à la remarque 3.1.2, il n'est pas toujours nécessaire de distinguer l'expression des problèmes des solutions. Les langages de représentation des problèmes et des solutions sont identiques, c'est-à-dire que $\mathcal{L}_{pb} = \mathcal{L}_{sol}$. Les problèmes résolus par un cas source sont implicites.*

Les implications logiques permettent d'exprimer des relations de généralisation-spécialisation entre cas. Pour deux cas représentés par $\Psi_{\texttt{Cas}_1}$ et $\Psi_{\texttt{Cas}_2}$, si $\Psi_{\texttt{Cas}_1} \models \Psi_{\texttt{Cas}_2}$, c'est-à-dire $\mathrm{Mod}(\Psi_{\texttt{Cas}_1}) \subseteq \mathrm{Mod}(\Psi_{\texttt{Cas}_2})$, cela signifie que $\Psi_{\texttt{Cas}_1}$ est plus spécifique que $\Psi_{\texttt{Cas}_2}$, il associe moins de solutions (ou une solution plus précise) à moins de problèmes (des problèmes plus particuliers). L'intérêt

de représenter les connaissances dans un formalisme logique est de bénéficier d'outils de raisonnement, notamment de révision que nous utiliserons en section 3.2 pour définir une approche de l'adaptation.

Nous n'accorderons pas d'importance à la syntaxe, nous assimilerons donc une BC à ses modèles, un cas correspond à un ensemble $\mathtt{Cas} \subseteq \mathcal{U}$ et les connaissances du domaine à un ensemble $\mathtt{CD} \subseteq \mathcal{U}$. Certaines approches de l'adaptation, que nous verrons plus loin, manipulent des cas ponctuels, c'est-à-dire des cas qui correspondent à un singleton. Nous noterons en minuscule un tel cas (de même pour son problème et sa solution), et nous l'assimilerons à un élément de $\mathcal{U}$: $\mathtt{cas} = (\mathtt{pb}, \mathtt{sol}) \in \mathcal{U}$ avec $\mathtt{pb} \in \mathcal{U}_{\mathtt{pb}}$ et $\mathtt{sol} \in \mathcal{U}_{\mathtt{sol}}$.

Nous avons présenté en section 2.4.1 les formalismes attributs-valeurs simples comme des formalismes logiques, les travaux présentés dans ce chapitre s'appliqueront donc aussi dans ce cadre. Nous nous sommes intéressés en section 2.5 aux logiques de descriptions. Comme les formalismes objet, elles établissent des classes d'instances et des relations entre ces instances. Il est possible d'exprimer des connaissances numériques dans certaines logiques de descriptions par l'introduction de *domaines concrets* [Baader et Hanschke, 1991]. De nombreuses ontologies décrivant différents domaines sont disponibles dans ces logiques, notamment dans le cadre du web sémantique et peuvent donc être exploitées par un système de RÀPC fonctionnant dans ce formalisme.

3.1.4 L'adaptation en RÀPC

Habituellement, l'adaptation est présentée en distinguant l'expression des problèmes de celle des solutions. Étant donné le cas cible $\mathtt{Cible} = \mathtt{Pb_c} \times \mathcal{U}_{\mathtt{sol}}$ avec $\mathtt{Pb_c} \subseteq \mathcal{U}_{\mathtt{pb}}$, l'étape de remémoration doit proposer un cas source que nous notons $\mathtt{Source} = \mathtt{Pb_s} \times \mathtt{Sol_s}$ avec $\mathtt{Pb_s} \subseteq \mathcal{U}_{\mathtt{pb}}$ et $\mathtt{Sol_s} \subseteq \mathcal{U}_{\mathtt{sol}}$. Le but de l'adaptation est de proposer une solution candidate $\mathtt{Sol_c}$ au problème $\mathtt{Pb_c}$. Pour être retenue, cette solution doit être cohérente avec les connaissances dont dispose le système sur le problème posé, c'est-à-dire avec le contexte cible : $\mathtt{CD} \cap \mathtt{Pb_c}$. L'adaptation est donc une opération qui prend en paramètres des connaissances du domaine $\mathtt{CD}$, un cas source, c'est-à-dire un couple $(\mathtt{Pb_s}, \mathtt{Sol_s})$ et un problème $\mathtt{Pb_c}$ et qui génère une solution :

$$(\mathtt{CD}, \mathtt{Pb_s}, \mathtt{Sol_s}, \mathtt{Pb_c}) \longmapsto \mathtt{Sol_c}$$

telle que $\mathtt{CD} \cap (\mathtt{Pb_c} \times \mathtt{Sol_c})$ soit cohérent. Le cas obtenu est noté $\mathtt{CibleComplétée} = \mathtt{Pb_c} \times \mathtt{Sol_c}$.

Remarque 3.1.4. *Dans le cadre de la remarque 3.1.3, où l'on ne fait pas la distinction entre description des problèmes et des solutions, c'est-à-dire que $\mathcal{L}_{pb} = \mathcal{L}_{sol}$, l'adaptation consiste à préciser le cas* `Cible` *en* `CibleComplétée` *tel que* `CibleComplétée` $\subseteq$ `Cible` *et que* `CD` $\cap$ `CibleComplétée` $\neq \emptyset$. *Dans ce cadre, l'adaptation est donc une opération qui prend en paramètre des connaissances du domaine* `CD`*, un cas source* `Source` *et un cas cible* `Cible` *et*

génère un cas `CibleComplétée` *:*

$$(\mathtt{CD}, \mathtt{Source}, \mathtt{Cible}) \longmapsto \mathtt{CibleComplétée}$$

L'adaptation est souvent développée de façon *ad hoc* pour les systèmes de RÀPC suivant des connaissances d'adaptation propres au domaine d'application. Quelques formes d'adaptation ont été définies dans la littérature et se retrouvent régulièrement dans les systèmes de RÀPC.

Adaptation par généralisation/spécialisation. Pour cette adaptation, on suppose que les cas sont organisés en hiérarchie suivant des relations de généralisation spécialisation. L'adaptation de `Source` pour résoudre `Cible` consiste alors à généraliser `Source` en `Source'` consistant avec `Cible` ($\mathtt{Source} \subseteq \mathtt{Source}'$ et $\mathtt{Source}' \cap \mathtt{Cible} \neq \emptyset$), puis à spécialiser `Source'` par `Cible` ($\mathtt{CibleComplétée} = \mathtt{Source}' \cap \mathtt{Cible}$). L'effort d'adaptation est liée à l'importance de la généralisation de `Source` et à la spécialisation par `Cible`. Les coûts de généralisation et de spécialisation peuvent être calculés différemment. Typiquement, la généralisation, qui fait perdre des caractéristiques importantes de la solution est plus coûteuse que la spécialisation qui est en fait une déduction en général (de coût nul).

Adaptation par règles d'adaptation [Melis *et al.*, 1998]. Les connaissances d'adaptation sont des règles d'adaptation exprimées sur $\mathcal{U}$, c'est-à-dire avec des problèmes et des solutions ponctuelles. Une règle d'adaptation est un couple $(\mathtt{r}, \mathcal{A}_{\mathtt{r}})$ où $\mathtt{r}$ est une relation entre problèmes ponctuels et où $\mathcal{A}_{\mathtt{r}}$ est une fonction qui prend en paramètres deux problèmes reliés par $\mathtt{r}$, dont on connaît une solution pour l'un des deux, et qui renvoie une solution candidate pour l'autre problème. Cette règle d'adaptation s'interprète de la façon suivante : si les problèmes $\mathtt{pb}^{\mathtt{c}}$ et $\mathtt{pb}^{\mathtt{s}}$ sont reliés par $\mathtt{r}$ et que $\mathtt{sol}^{\mathtt{s}}$ est une solution au problème $\mathtt{pb}^{\mathtt{s}}$, alors $\mathcal{A}_{\mathtt{r}}(\mathtt{pb}^{\mathtt{s}}, \mathtt{sol}^{\mathtt{s}}, \mathtt{pb}^{\mathtt{c}})$ résout probablement $\mathtt{pb}^{\mathtt{c}}$.

Les règles de reformulation peuvent être composées, une première règle sert à résoudre un premier problème intermédiaire à partir du cas source, puis la règle suivante adapte la solution obtenue pour ce problème intermédiaire pour un deuxième problème et ainsi de suite jusqu'à la résolution du cas cible.

La présentation de cette approche d'adaptation est développée en section 3.3.1.

Adaptation différentielle [Fuchs *et al.*, 2000]. Cette adaptation s'applique aux formalismes attributs-valeurs simples, on suppose donc que $\mathcal{U}_{\mathtt{pb}} = \mathcal{U}^1_{\mathtt{pb}} \times \ldots \times \mathcal{U}^n_{\mathtt{pb}}$ et $\mathcal{U}_{\mathtt{sol}} = \mathcal{U}^1_{\mathtt{sol}} \times \ldots \times \mathcal{U}^m_{\mathtt{sol}}$. La résolution de problèmes est assimilée ici à l'évaluation d'une fonction $\mathtt{sol}$ de $\mathcal{U}_{\mathtt{pb}}$ dans $\mathcal{U}_{\mathtt{sol}}$, où pour tout problème $\mathtt{pb} \in \mathcal{U}_{\mathtt{pb}}$, $\mathtt{sol}(\mathtt{pb}) \in \mathcal{U}_{\mathtt{sol}}$ est une solution de $\mathtt{pb}$.

L'adaptation différentielle repose sur une analogie avec l'approximation de Taylor-Young à l'ordre 1 : pour une fonction f de $\mathbb{R}^n$ dans $\mathbb{R}^m$ dérivable en un point $x = (x_1, \ldots, x_n) \in \mathbb{R}^n$ et

pour un point $y = (y_1, \ldots, y_n) \in \mathbb{R}^n$ proche de x

$$f_j(y) \simeq f_j(x) + \sum_i \frac{\partial f_j}{\partial x_i}(x)(y_i - x_i)$$

où $\frac{\partial f_j}{\partial x_i}(x)$ est la dérivée partielle en x de la composante j de f selon sa $i^{\text{ème}}$ variable.

Chaque cas source $(\mathtt{pb^s}, \mathtt{sol}(\mathtt{pb^s})) \in \mathcal{U}$ est accompagné de dépendances établissant des corrélations entre les variations d'un problème autour de $\mathtt{pb^s}$ et les variations de sa solution autour de $\mathtt{sol}(\mathtt{pb^s})$. La dépendance entre la composante i des problèmes et la composante j des solutions associée au cas source $(\mathtt{pb^s}, \mathtt{sol}(\mathtt{pb^s}))$ correspond pour `sol` aux dérivées partielles $\frac{\partial f_j}{\partial x_i}(x)$ de f au point x. Une approximation de $\mathtt{sol}(\mathtt{pb^c})$ est calculée en répercutant sur $\mathtt{sol}(\mathtt{pb^s})$ les variations induites par les variations entre $\mathtt{pb^c}$ et $\mathtt{pb^s}$.

La présentation de cette approche est développée en section 3.3.2.

Adaptation par révision (+-adaptation) [Lieber, 2007]. Nous défendons l'idée que les approches de l'adaptation utilisées pour le RÀPC hypothétique, dont les approches présentées plus haut, suivent le principe de la minimisation de l'effort d'adaptation sous contrainte de cohérence avec le contexte cible, c'est-à-dire $\mathtt{CD} \cap \mathtt{Cible}$. Cette hypothèse correspond à l'un des critères énoncés par [Kolodner, 1993] pour le choix d'une adaptation : une adaptation simple est préférable à une adaptation plus complexe [10].

Dans la section suivante (section 3.2) nous étudions une approche de l'adaptation définie comme une révision par le contexte du cas cible de l'expérience de résolution de problème apportée par le cas source. Ce parallèle entre adaptation et révision est motivé par le fait que ces deux opérations sont guidées par la minimisation de modifications à apporter à des connaissances pour les rendre cohérentes avec un nouveau contexte.

Un objectif de cette approche de l'adaptation est d'unifier plusieurs autres approches par une définition formelle sur laquelle peuvent être appliqués les résultats et de la révision. Nous montrons en section 3.3 que certaines approches de l'adaptation présentées plus haut peuvent être mises sous forme d'adaptation par révision.

3.2 Définition de l'adaptation par révision

On considère ici l'adaptation d'un cas `Source` pour résoudre un cas `Cible`. On ne suppose pas de distinction entre problèmes et solutions, on se place donc dans le cadre de la remarque 3.1.4 : on cherche à préciser `Cible` en $\mathtt{CibleComplétée} \subseteq \mathtt{Cible}$ tel que $\mathtt{CD} \cap \mathtt{CibleComplétée} \neq \emptyset$.

10. L'autre critère principal de [Kolodner, 1993] pour le choix d'une adaptation donne la préférence aux adaptations spécifiques (et donc plus ciblées pour le problème d'adaptation posé) sur les adaptations plus générales. Cela peut se traduire par un effort d'adaptation plus faible pour une adaptation spécifique que pour une adaptation générale. Une adaptation spécifique est donc préférée sous condition d'être applicable au problème d'adaptation posé.

Nous avons vu que pour le RÀPC hypothétique, l'adaptation était souvent guidée par la minimisation d'un effort d'adaptation. L'effort d'adaptation correspond à une estimation, compte tenu des connaissances d'adaptation disponibles, du risque d'obtenir une solution non satisfaisante après adaptation d'un cas source.

Ce changement minimal du cas source lors de l'adaptation peut être comparé au changement minimal des connaissances pour la révision. [Lieber, 2007] s'appuie sur ce parallèle pour proposer une approche de l'adaptation définie comme la révision des connaissances apportées par `Source` par le contexte du cas `Cible`. Les connaissances sont à considérer modulo les connaissances du domaine, `CibleComplétée` est obtenu par révision du contexte source : $\texttt{CD} \cap \texttt{Source}$ par le contexte cible : $\texttt{CD} \cap \texttt{Cible}$.

$$\texttt{CibleComplétée} = (\texttt{CD} \cap \texttt{Source}) \dot{+} (\texttt{CD} \cap \texttt{Cible}) \tag{3.1}$$

L'opérateur de révision choisi doit produire un `CibleComplétée` qui minimise l'effort d'adaptation, il dépend donc des connaissances d'adaptation disponibles (voir les opérateurs proposés en section 3.3). Des propriétés générales attendues de l'adaptation peuvent cependant être mis en relation avec les postulats de l'adaptation vus en section 2.3.

Remarque 3.2.1. *On peut cependant remarquer que dans les situations habituellement traitées par la révision, les connaissances révisées sont moins précises que les connaissances par lesquelles elles sont révisées (typiquement ce sont des faits qui provoquent la révision de la représentation générale d'un domaine). Pour l'adaptation, en revanche, le cas source représente une expérience particulière de résolution de problèmes et il est plus précis que le cas cible que l'on cherche à spécifier.*

3.2.1 Propriétés de l'adaptation par révision

Deux propriétés sont requises pour le résultat `CibleComplétée` de l'adaptation d'un cas `Source` pour résoudre un cas `Cible`. `CibleComplétée` doit répondre au problème posé par `Cible`, il doit donc satisfaire les contraintes exprimées par `Cible`, soit $\texttt{CibleComplétée} \subseteq \texttt{Cible}$. Le postulat (G1) établit une condition plus forte : $\texttt{CibleComplétée} \subseteq \texttt{CD} \cap \texttt{Cible}$ où `CD` sont les connaissances du domaine. La condition supplémentaire $\texttt{CibleComplétée} \subseteq \texttt{CD}$ n'est pas contraignante puisque les cas sont considérés modulo `CD`.

Pour que le résultat de l'adaptation ait une signification, il doit être cohérent avec `CD`. Suivant la propriété précédente, si `Cible` n'est pas cohérent avec les connaissances du domaine, c'est-à-dire si $\texttt{CD} \cap \texttt{Cible} = \emptyset$, `CibleComplétée` peut ne pas être cohérent non plus. En revanche lorsque $\texttt{CD} \cap \texttt{Cible}$ est cohérent, on souhaite que l'adaptation produise un résultat utile et donc que $\texttt{CD} \cap \texttt{CibleComplétée}$ soit cohérent. Si $\texttt{CD} \cap \texttt{Cible}$ est cohérent, (G3) établit que `CibleComplétée` l'est aussi, ce qui implique, s'il vérifie (G1), qu'il est cohérent avec `CD`.

Selon [Katsuno et Mendelzon, 1991b], les postulats (G5) et (G6) expriment la minimalité du changement. Ces postulats caractérisent la minimisation d'un pré-ordre total (propositions 1.2.1

et 1.2.2, page 9). Selon [Katsuno et Mendelzon, 1991b], dans le cadre de la révision, ce pré-ordre correspond à la préférence pour des changement minimaux de connaissances. Dans le cadre de l'adaptation, il correspond à la préférence pour les adaptations d'effort minimal.

L'effort d'adaptation minimal correspond à la situation pour laquelle aucune adaptation n'est nécessaire, c'est-à-dire lorsque $\mathtt{CD} \cap \mathtt{Source}$ est cohérent avec $\mathtt{CD} \cap \mathtt{Cible}$ et donc que les solutions proposées par `Source` sont applicables à `Cible`. `Source` est alors simplement spécialisé en `Cible` en $\mathtt{CibleComplétée} = (\mathtt{CD} \cap \mathtt{Source}) \cap (\mathtt{CD} \cap \mathtt{Cible})$, ce qui correspond au postulat (G2). Lorsque le cas source est ponctuel, c'est-à-dire que $\mathtt{CD} \cap \mathtt{Source} = \mathtt{srce}$ donc $\mathtt{CD} \cap \mathtt{Source}$ est cohérent avec $\mathtt{CD} \cap \mathtt{Cible}$ ssi $\mathtt{srce} \in \mathtt{CD} \cap \mathtt{Cible}$, ce qui donne comme résultat $\mathtt{CibleComplétée} = \{\mathtt{srce}\} = \mathtt{CD} \cap \mathtt{Source}$, il s'agit alors d'une *adaptation par copie*.

3.2.2 Adaptation conservatrice

Dans [Lieber, 2007] et [Cojan et Lieber, 2008a], l'adaptation par révision est appelée *conservative adaptation* (adaptation conservatrice) Ce qualificatif fait référence au comportement de l'adaptation par copie et correction de la $\dot{+}$-adaptation pour des opérateurs définis grâce à des distances simples. En effet, si l'on distingue la description des problèmes de celle des solutions et que la distance d s'écrit sous la forme d'une somme d'une distance sur $\mathcal{U}_{\mathtt{pb}}$ et d'une distance sur $\mathcal{U}_{\mathtt{sol}}$, donc sans un terme qui caractérise le lien problème-solution, alors la $\dot{+}^d$-adaptation se ramène à une copie de la solution du cas source, corrigée si nécessaire pour rétablir la cohérence avec les connaissances du domaine. C'est le cas, par exemple, avec la distance définie dans l'exemple 2.4.1 (page 36).

Proposition 3.2.1. *En l'absence de connaissances du domaine ($\mathtt{CD} = \mathcal{U}$), si on a une séparation problème-solution des cas ($\mathcal{U} = \mathcal{U}_{pb} \times \mathcal{U}_{sol}$, $\mathtt{Cible} = \{pb^c\} \times \mathcal{U}_{sol}$, $\mathtt{Source} = Pb_s \times Sol_s$), si la distance d sur $\mathcal{U}$ s'écrit comme la somme d'une distance d_{pb} sur $\mathcal{U}_{pb}$ et d'une distance d_{sol} sur $\mathcal{U}_{sol}$, i.e. si pour tout $u_1 = (x_1, y_1), u_2 = (x_2, y_2) \in \mathcal{U}$,*

$$d(u_1, u_2) = d_{pb}(x_1, x_2) + d_{sol}(y_1, y_2)$$

et enfin si Sol_s est un fermé de $(\mathcal{U}_{sol}, d_{sol})$ ($\{y \in \mathcal{U}_{sol} \mid d_{sol}(Sol_s, y) = 0\} = Sol_s$) alors la $\dot{+}^d$-adaptation de $\mathtt{Source}$ pour $\mathtt{Cible}$ consiste en une copie de la solution du cas source :

$$\mathtt{CibleComplétée} = \{pb^c\} \times Sol_s$$

Démonstration. Soit $u_{\mathtt{c}} = (x_{\mathtt{c}}, y_{\mathtt{c}}) \in \mathtt{CibleComplétée}$. Puisque $\mathtt{CibleComplétée} \subseteq \mathtt{Cible} =$

$\{\mathtt{pb^c}\} \times \mathcal{U}_{\mathtt{sol}}$, $x_{\mathtt{c}} = \mathtt{pb^c}$, et par définition de la $\dot{+}^d$-adaptation, $u_{\mathtt{c}}$ minimise $d(\mathtt{Source}, \cdot)$. Or :

$$\begin{aligned} d(\mathtt{Source}, u_{\mathtt{c}}) &= \inf_{u_{\mathtt{s}} \in \mathtt{Source}} d(u_{\mathtt{s}}, u_{\mathtt{c}}) \\ &= \inf_{(x_{\mathtt{s}}, y_{\mathtt{s}}) \in \mathtt{Pb_s} \times \mathtt{Sol_s}} \left[d_{\mathtt{pb}}(x_{\mathtt{s}}, \mathtt{pb^c}) + d_{\mathtt{sol}}(y_{\mathtt{s}}, y_{\mathtt{c}}) \right] \\ &= \underbrace{\left[\inf_{x_{\mathtt{s}} \in \mathtt{Pb_s}} d_{\mathtt{pb}}(x_{\mathtt{s}}, \mathtt{pb^c}) \right]}_{d_{\mathtt{pb}}(\mathtt{Pb_s}, \mathtt{pb^c})} + \underbrace{\left[\inf_{y_{\mathtt{s}} \in \mathtt{Sol_s}} d_{\mathtt{sol}}(y_{\mathtt{s}}, y_{\mathtt{c}}) \right]}_{d_{\mathtt{sol}}(\mathtt{Sol_s}, y_{\mathtt{c}})} \end{aligned}$$

Donc $u_{\mathtt{c}} \in \mathtt{CibleComplétée}$ si $y_{\mathtt{c}}$ minimise $d_{\mathtt{sol}}(\mathtt{Sol_s}, y_{\mathtt{c}})$, ce qui revient à dire que $d(\mathtt{Sol_s}, y_{\mathtt{c}}) = 0$ et donc $y_{\mathtt{c}} \in \mathtt{Sol_s}$. □

On montre en section 3.3 que l'adaptation par révision peut avoir un comportement plus sophistiqué. C'est pourquoi nous utilisons le terme d'adaptation conservatrice pour des adaptations par révision suivant ce comportement de copie et correction. Ce comportement correspond à l'adaptation par la révision suivant des opérateurs de révision orthogonaux :

Définition 3.2.1. *Un opérateur de révision $\dot{+}$ sur $\mathcal{U}$ est dit* orthogonal *(par rapport à la décomposition problème-solution de $\mathcal{U}$) s'il existe un opérateur de révision $\dot{+}_{pb}$ sur $\mathcal{U}_{pb}$ et un opérateur de révision $\dot{+}_{sol}$ sur $\mathcal{U}_{sol}$ vérifiant :*

$$\begin{gathered} si \quad P_1, P_2 \in 2^{\mathcal{U}_{pb}} \quad et \quad S_1, S_2 \in 2^{\mathcal{U}_{sol}}, \quad alors \\ (P_1 \times S_1) \dot{+} (P_2 \times S_2) = (P_1 \dot{+}_{pb} P_2) \times (S_1 \dot{+}_{sol} S_2) \end{gathered}$$

Cet adjectif est justifié plus loin dans l'exemple 3.3.3 (page 99).

Définition 3.2.2. *Une adaptation est dite conservatrice s'il s'agit d'une $\dot{+}$-adaptation où $\dot{+}$ est un opérateur de révision orthogonal selon la décomposition problème-solution de $\mathcal{U}$. .*

Effet de compensation de l'adaptation conservatrice

L'adaptation conservatrice n'est cependant pas inintéressante. Certains comportements peuvent être obtenus en guidant la réparation par les connaissances du domaine. En particulier, l'effet de compensation est beaucoup utilisé pour l'adaptation des quantités d'ingrédients dans le système TAAABLE.

Par exemple, supposons que l'on veuille adapter une recette de tarte aux pommes pour obtenir une recette de tarte aux poires (parce qu'on a des poires mais pas de pommes). Les recettes sont représentées dans un formalisme attributs-valeurs simples comme présenté en section 2.4.2, avec un attribut par ingrédient : `pâte`, `pomme`, `poire` et `piridion` (famille de fruits dont font partie les pommes et les poires), `saccharose` (sucre en poudre) et `sucre` (somme du sucre ajouté à la recette sous forme de saccharose et du sucre contenu dans les autres ingrédients) La valeur associée à un attribut est un réel qui représente la masse en grammes dans la recette

de l'ingrédient correspondant à l'attribut. On considère donc l'adaptation du cas `Source` pour résoudre `Cible` en prenant compte les connaissances du domaine `CD` :

$$\begin{aligned}
\texttt{Source} &= \mathrm{Mod}(\{\texttt{pâte} = 100,\ \texttt{pomme} = 300,\ \texttt{poire} = 0,\ \texttt{saccharose} = 30\}) \\
\texttt{Cible} &= \mathrm{Mod}(\{\texttt{pomme} = 0\}) \\
\texttt{CD} &= \mathrm{Mod}\left(\left\{\begin{array}{l} \texttt{piridion} = \texttt{pomme} + \texttt{poire}, \\ \texttt{sucre} = \texttt{saccharose} + 0.05 \cdot \texttt{pomme} + 0.1 \cdot \texttt{poire} \end{array}\right\}\right)
\end{aligned}$$

Avec une « distance » d définie On considère un opérateur de révision défini grâce à une distance de la forme de celles de l'exemple 2.4.1 (page 36) : pour $x, y \in \mathcal{U}$ par $d(x, y) = \sum_i w_i |y_i - x_i|$ où l'indice i parcourt l'ensemble des attributs et pour un attribut i, y_i et x_i sont respectivement les valeurs de l'attribut i dans les interprétations x et y. Avec des coefficient tels que $w_{\texttt{piridion}} > w_{\texttt{pomme}} + w_{\texttt{poire}}$ $w_{\texttt{sucre}} > w_{\texttt{saccharose}}$, $w_{\texttt{pomme}} > w_{\texttt{saccharose}}$ et $w_{\texttt{poire}} > w_{\texttt{saccharose}}$ l'adaptation par révision donne :

$$\texttt{CibleComplétée} = \mathrm{Mod}(\{\texttt{pâte} = 100,\ \texttt{pomme} = 0,\ \texttt{poire} = 300,\ \texttt{saccharose} = 15\})$$

Afin de conserver la quantité de piridion, la suppression des pommes est compensée par l'ajout de la même masse de poires. Et la masse de saccharose est réduite pour compenser la quantité de sucre plus importante contenue dans les poires.

3.2.3 Adaptation par révision et remémoration guidée par l'adaptation

On considère une base de cas $\{\texttt{Source}_i\}_{1 \leqslant i \leqslant n}$, et on définit l'ensemble $\texttt{SOURCE} = \bigcup_{1 \leqslant i \leqslant n} \texttt{Source}_i$. On suppose ici que l'opérateur de révision utilisé pour l'adaptation est défini à partir d'une distance d comme en section 2.3. La révision de `SOURCE` par `Cible`, modulo les connaissances du domaine, donne le même résultat que l'adaptation des cas sources remémorés suivant le principe de la remémoration guidée par l'adaptation. En effet, si on note $\texttt{CibleComplétée}_i$ le résultat de l'adaptation de $\texttt{Source}_i$:

$$\begin{aligned}
\texttt{CibleComplétée}_i &= (\texttt{CD} \cap \texttt{Source}_i) \dot{+} (\texttt{CD} \cap \texttt{Cible}) \\
&= \operatorname*{Min}_{d(\texttt{CD} \cap \texttt{Source}_i, \cdot)} (\texttt{CD} \cap \texttt{Cible})
\end{aligned}$$

Le coût de cette adaptation est de $d(\texttt{CD} \cap \texttt{Source}_i, \texttt{CD} \cap \texttt{Cible})$. Les cas remémorés sont donc ceux qui minimisent ce coût, soit les cas $\texttt{Source}_i$ tel que $d(\texttt{CD} \cap \texttt{Source}_i, \texttt{CD} \cap \texttt{Cible}) = d(\texttt{CD} \cap \texttt{SOURCE}, \texttt{CD} \cap \texttt{Cible})$. La révision de l'ensemble de la base de connaissance par `Cible`, vaut donc :

$$\left(\bigcup_i (\texttt{CD} \cap \texttt{Source}_i)\right) \dot{+} (\texttt{CD} \cap \texttt{Cible}) = \operatorname*{Min}_{d(\bigcup_i (\texttt{CD} \cap \texttt{Source}_i), \cdot)} (\texttt{CD} \cap \texttt{Cible})$$

D'après la proposition 1.2.3, page 11 :

$$\left(\bigcup_i (\texttt{CD} \cap \texttt{Source}_i)\right) \dot{+} (\texttt{CD} \cap \texttt{Cible}) = \bigcup_{\substack{1 \leqslant i \leqslant n \\ \texttt{Source}_i \text{ est remémoré}}} \operatorname*{Min}_{d(\texttt{CD} \cap \texttt{Source}_i, \cdot)} (\texttt{CD} \cap \texttt{Cible})$$
$$= \bigcup_{\substack{1 \leqslant i \leqslant n \\ \texttt{Source}_i \text{ est remémoré}}} \texttt{CibleComplétée}_i$$

En particulier si un seul cas source $\texttt{Source}_i$ est remémoré, le résultat de l'adaptation de $\texttt{Source}_i$ coïncide avec la révision de `SOURCE` par `Cible`. La remémoration est donc implicitement effectuée lors de la révision de `SOURCE`.

3.2.4 Adaptation par mise à jour

Plusieurs opérations de changement minimal de connaissances ont été présentées en section 2.1, dont deux traitent des changements à apporter à des connaissances pour les rendre cohérentes avec d'autres connaissances : la révision et la mise à jour.

D'après [Katsuno et Mendelzon, 1991a], la différence entre ces deux opérations tient à la cause des incohérences éventuelles entre les connaissances initiales et les nouvelles connaissances. Dans le cas de la révision, elles sont la conséquence d'erreurs ou d'approximations dans les connaissances initiales. Dans le cas de la mise à jour, elles résultent d'une évolution dans le temps du domaine décrit.

Intuitivement l'adaptation correspond à une révision : la solution du cas source est prise comme une solution approchée au problème cible qui doit être corrigée si elle est incohérente avec le contexte cible. Excepté pour certaines applications particulières, les contextes du cas source et du cas cible ne sont pas reliés chronologiquement, les incohérences éventuelles entre la solution source et le contexte cible ne sont donc pas la conséquence d'une évolution dans le temps.

Cette différence de nature entre révision et mise à jour se traduit par un signification différente de la notion de minimalité du changement. Lors de la révision d'une BC Ψ par une BC M, les modèles de M retenus sont ceux qui minimisent globalement le changement par rapport aux modèles de Ψ, c'est-à-dire les modèles de M qui correspondent à la correction minimale des modèles les moins erronés de Ψ. Pour un opérateur de révision $\dot{+}$ défini à l'aide d'une distance d :

$$\mathrm{Mod}(\Psi \dot{+} \mathrm{M}) = \operatorname*{Min}_{d(\mathrm{Mod}(\Psi), \cdot)} \mathrm{Mod}(\mathrm{M})$$

Lors de la mise à jour de Ψ par M, les modèles de M retenus sont ceux qui minimisent le changement relativement à l'un des modèles de Ψ, c'est-à-dire les modèles de M qui résultent d'un changement minimal de l'un des modèles de Ψ. Si on caractérise l'importance du changement

par la distance d, la mise à jour de Ψ par M, notée $\Psi \diamond \mathrm{M}$, vérifie :

$$\mathrm{Mod}(\Psi \diamond \mathrm{M}) = \bigcup_{I \in \mathrm{Mod}(\Psi)} \underset{d(I,\cdot)}{\mathrm{Min}}\, \mathrm{Mod}(\mathrm{M})$$

On peut remarquer que la révision donne un résultat plus précis que la mise à jour : $\mathrm{Mod}(\Psi \dot{+} \mathrm{M}) \subseteq \mathrm{Mod}(\Psi \diamond \mathrm{M})$.

[Katsuno et Mendelzon, 1991a] justifient cette forme de changement minimal pour la mise à jour par le fait que si plusieurs interprétations peuvent être faites de la BC Ψ, correspondant à plusieurs états initiaux possibles du monde décrit, l'importance du changement à apporter à une interprétation I de Ψ ne donne aucune indication quant à sa vraisemblance. Les résultats possibles de l'évolution doivent donc être comparés séparément pour chaque état initial. Au contraire, lors de la révision, si une interprétation I décrite par Ψ s'avère être plus éloignée des modèles de M qu'une autre interprétation J, on retiendra plutôt les corrections de J.

Nous avons fait comme hypothèse que dans un cas source $\mathtt{Source} = \mathtt{Pb_s} \times \mathtt{Sol_s}$, pour tout problème $\mathtt{pb} \in \mathtt{Pb_s}$ et toute solution $\mathtt{sol} \in \mathtt{Sol_s}$ $\mathtt{sol}$ est une solution à $\mathtt{pb}$. L'adaptation se fait suivant la même forme de changement minimal que la révision : si $\mathtt{sol}_1$ et $\mathtt{sol}_2$ sont des solution associées à un problème $\mathtt{pb}$ telles que $d((\mathtt{pb}, \mathtt{sol}_1), \mathtt{Cible}) < d((\mathtt{pb}, \mathtt{sol}_2), \mathtt{Cible})$, l'effort d'adaptation de $(\mathtt{pb}, \mathtt{sol}_2)$ étant plus élevé que pour $(\mathtt{pb}, \mathtt{sol}_1)$, on ne retient que le résultat de l'adaptation de $(\mathtt{pb}, \mathtt{sol}_1)$. En revanche, si on fait l'hypothèse plus faible que tout problème $\mathtt{pb} \in \mathtt{Pb_s}$, admet une solution dans $\mathtt{sol} \in \mathtt{Sol_s}$, un couple $(\mathtt{pb}, \mathtt{sol}_2) \in \mathtt{Source}$ ne peut pas être ignoré parce qu'un autre couple $(\mathtt{pb}, \mathtt{sol}_1) \in \mathtt{Source}$ est plus proche de $\mathtt{Cible}$. En effet il n'est pas garanti que $\mathtt{sol}_1$ soit une solution de $\mathtt{pb}$. Les adaptations des différents éléments de $\mathtt{Source}$ doivent donc être comparées séparément, comme pour une mise à jour. En pratique, les cas sources sont souvent ponctuels, c'est-à-dire que $\mathtt{Source} = \{(\mathtt{pb}, \mathtt{srce})\}$, et l'adaptation par la mise à jour d'un tel cas source donne le même résultat que l'adaptation par la révision.

Dans la suite, nous continuons à faire l'hypothèse que pour tout $(\mathtt{pb}, \mathtt{sol}) \in \mathtt{Source}$, $\mathtt{sol}$ est une solution de $\mathtt{pb}$, suivant le principe de la minimisation de l'effort d'adaptation, l'adaptation correspond donc à une révision du contexte source par le contexte cible. L'étude de l'adaptation suivant une autre forme de minimalité du changement est gardé en perspective.

3.3 Lien avec d'autres approches de l'adaptation

Nous avons présenté en section 3.2 une approche de l'adaptation définie comme la révision de l'expérience apportée par un cas source par le contexte du cas cible. Cette définition est justifiée par l'idée que nous défendons qui veut que la plupart des approches de l'adaptation en RÀPC hypothétique sont guidées par la minimisation d'un effort d'adaptation, et peuvent donc être formalisées par un opérateur de révision.

Nous montrons dans la suite que cette affirmation est vérifiée pour deux approches d'adaptation précédemment formalisées : l'adaptation par règles en section 3.3.1 et l'adaptation différen-

tielle en section 3.3.2.

3.3.1 L'adaptation par règles et l'adaptation par révision

Cette section reprend les travaux présentés dans [Cojan et Lieber, 2010a].

Présentation de l'adaptation par règles

On suppose ici que les cas sont ponctuels et que l'on distingue la description des problèmes de celle des solutions :

$$\texttt{Source} = \{(\texttt{pb}^{\texttt{s}}, \texttt{sol}^{\texttt{s}})\}$$
$$\texttt{Cible} = \{\texttt{pb}^{\texttt{c}}\} \times \mathcal{U}_{\texttt{sol}}$$

avec $\texttt{pb}^{\texttt{s}}, \texttt{pb}^{\texttt{c}} \in \mathcal{U}_{\texttt{pb}}$ et $\texttt{sol}^{\texttt{s}} \in \mathcal{U}_{\texttt{sol}}$.

Nous nous limitons ici aux règles d'adaptation appelées *reformulations* dans [Melis *et al.*, 1998]. Une telle règle est un couple $(\texttt{r}, \mathcal{A}_{\texttt{r}})$ où $\texttt{r}$ est une relation entre problèmes et où $\mathcal{A}_{\texttt{r}}$ est une fonction qui à un problème d'adaptation associe une solution. Cette règle d'adaptation s'interprète de la façon suivante :

$$\begin{array}{l} \textbf{si } \texttt{pb}^{\texttt{s}}\ \texttt{r}\ \texttt{pb}^{\texttt{c}} \\ \quad \textbf{alors } \texttt{sol}^{\texttt{c}} = \mathcal{A}_{\texttt{r}}(\texttt{pb}^{\texttt{s}}, \texttt{sol}^{\texttt{s}}, \texttt{pb}^{\texttt{c}}) \text{ résout probablement cible} \end{array} \tag{3.2}$$

On note $\texttt{RA}$ l'ensemble fini des règles d'adaptation du système de RÀPC considéré.

On peut composer les règles d'adaptation de la façon suivante. On appelle *chemin de similarité* de l'instance de problème $\texttt{pb}^{\texttt{s}}$ à l'instance de problème $\texttt{pb}^{\texttt{c}}$, un chemin reliant ces deux instances de problème par une composition de relations $\texttt{r}$ telles que $(\texttt{r}, \mathcal{A}_{\texttt{r}}) \in \texttt{RA}$. On peut l'écrire $\{(\texttt{pb}_{i-1}, \texttt{r}_i, \texttt{pb}_i)\}_{1 \leqslant i \leqslant q}$ où $\texttt{pb}_0 = \texttt{pb}^{\texttt{s}}$, $\texttt{pb}_q = \texttt{pb}^{\texttt{c}}$ et $\texttt{pb}_{i-1}\ \texttt{r}_i\ \texttt{pb}_i$ pour tout i. L'étape de construction du chemin de similarité est un problème algorithmique qui dépend de la forme des $\texttt{r}_i$ et qui peut se révéler complexe ; dans certains cas, il est indécidable (car la composition de deux relations décidables n'est pas nécessairement décidable). Dans [Lieber, 2002], un algorithme établissant un chemin de similarité durant la remémoration, qui combine un parcours de hiérarchie avec une recherche A* et qui a été implanté pour plusieurs applications, est présenté.

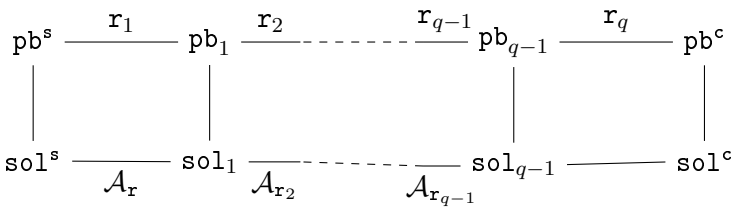
À $\texttt{pb}^{\texttt{s}}$ et $\texttt{pb}^{\texttt{c}}$ donnés, il peut y avoir plusieurs chemins de similarité. Pour choisir entre eux, on utilise une fonction de coût qui à $(\texttt{r}, \mathcal{A}_{\texttt{r}}) \in \texttt{RA}$ associe $\texttt{coût}(\texttt{r}) > 0$. Le coût d'un chemin de similarité se calcule de façon additive, c'est $\sum_i \texttt{coût}(\texttt{r}_i)$. Pour fixer les idées, on peut définir $\texttt{coût}(\texttt{r}) = -\log P_{\texttt{r}}$ où $P_{\texttt{r}}$ est une estimation de la probabilité que l'application de (3.2) résout effectivement $\texttt{pb}^{\texttt{c}}$. L'additivité du coût traduit alors une hypothèse d'indépendance. Dans le cas général, il peut y avoir plusieurs chemins de similarité de coût minimal de $\texttt{pb}^{\texttt{s}}$ à $\texttt{pb}^{\texttt{c}}$, conduisant à plusieurs instances de solutions de $\texttt{pb}^{\texttt{c}}$. On note $\texttt{Solutions}(\texttt{pb}^{\texttt{c}})$ cet ensemble de solutions. Si le système n'a pas de connaissances annexes pour choisir une instance de solution

dans $\texttt{Solutions}(\texttt{pb}^{\texttt{c}})$, alors le choix est effectué par l'utilisateur (ou laissé au hasard).

L'adaptation consiste d'abord à construire le chemin de similarité de moindre coût, puis à « suivre » ce chemin, en appliquant successivement les $\mathcal{A}_{\texttt{r}_i}$. Étant donné un chemin de similarité $cs = \{(\texttt{pb}_{i-1}, \texttt{r}_i, \texttt{pb}_i)\}_{1 \leqslant i \leqslant q}$ de $\texttt{pb}^{\texttt{s}} \in \mathcal{U}_{\texttt{pb}}$ à $\texttt{pb}^{\texttt{c}} \in \mathcal{U}_{\texttt{pb}}$ et $\texttt{sol}^{\texttt{s}} \in \mathcal{U}_{\texttt{sol}}$, soit $\texttt{sol}_i = \mathcal{A}_{\texttt{r}_i}(\texttt{pb}_{i-1}, \texttt{sol}_{i-1}, \texttt{sol}_i)$ pour $1 \leqslant i \leqslant q$. $\texttt{sol}_q = \texttt{sol}^{\texttt{c}}$ est une proposition d'instance de solution pour $\texttt{pb}^{\texttt{c}}$. On appelle *chemin d'adaptation* de $(\texttt{pb}^{\texttt{s}}, \texttt{sol}^{\texttt{s}}) \in \mathcal{U}$ à $(\texttt{pb}^{\texttt{c}}, \texttt{sol}^{\texttt{c}}) \in \mathcal{U}$ un ensemble de la forme :

$$ca = \{(\texttt{pb}_{i-1}, \texttt{sol}_{i-1}), (\texttt{r}_i, \mathcal{A}_{\texttt{r}_i}), (\texttt{pb}_i, \texttt{sol}_i)\}_{1 \leqslant i \leqslant q}$$

Son coût est le coût de cs. Ce coût est nul ssi $\texttt{pb}^{\texttt{s}} = \texttt{pb}^{\texttt{c}}$ (i.e. $q = 0$). Ce chemin peut être illustré par le schéma suivant :



Exemple 3.3.1. *L'objectif de l'application est de calculer une valeur approchée d'un logarithme décimal en s'appuyant sur une table de logarithmes qui constituera la base de cas. Un cas source est donc un couple* $(x_s, y_s) \in \mathcal{U}_{pb} \times \mathcal{U}_{sol}$ *où* $\mathcal{U}_{pb} = \mathbb{R}_+^*$ *et* $\mathcal{U}_{sol} = \mathbb{R}$ *(*$\mathbb{R}_+^* =]0; +\infty[$*) et où* $y_s \simeq \log x_s$ *à* $\varepsilon > 0$ *fixé près.*

Un problème cible est un réel strictement positif $x_{\texttt{c}} \in \mathbb{R}_+^*$ *dont on veut une valeur approchée du logarithme. Il est représenté par le cas* $\texttt{Cible} = \{x_{\texttt{c}}\} \times \mathbb{R}$.

Les connaissances générales qu'on souhaite utiliser sont les suivantes[11] *(pour* $t \in \mathbb{R}_+^*$*) :*

$$\log t \leqslant 0{,}44 \cdot (t - 1) \tag{3.3}$$

$$\log 1/t = -\log t \tag{3.4}$$

$$\log 10t = \log t + 1 \tag{3.5}$$

On se place dans un cadre de RÀPC *hypothétique : l'approche cherchée proposera une valeur approchée de* $\log c$ *sans qu'on contrôle l'erreur commise.*

L'équation (3.3) *exprime des connaissances du domaine :*

$$\texttt{CD} = \{(x, y) \in \mathcal{U} \mid y \leqslant 0{,}44 \cdot (x - 1)\}$$

Les équations (3.4) et (3.5) expriment des connaissances d'adaptation et peuvent se traduire par

11. La première inégalité est une approximation de du fait que $\log t \leqslant \frac{t-1}{\ln 10}$ où $0.43 \leqslant \ln 10 \leqslant 0.44$. Donc, sauf pour un petit intervalle $[1 - \varepsilon, 1]$ de valeurs de t, $\log t \leqslant 0{,}44 \cdot (t - 1)$, et pour $t \in [1 - \varepsilon, 1]$, l'écart est petit. Donc pour une application dont le but est d'approcher la valeur de log, cette approximation n'est pas excessive.

trois règles d'adaptation :

$$\begin{cases} \textit{si } \mathtt{pb}^c = 1/\mathtt{pb}^s \\ \quad \textbf{alors } \mathtt{sol}^c = -\mathtt{sol}^s \end{cases} \qquad \begin{pmatrix} x \; r \; y \;\; \textit{ssi} \;\; y = 1/x \\ \mathcal{A}_r(x,z,y) = -z \end{pmatrix}$$

$$\begin{cases} \textit{si } \mathtt{pb}^c = 10 \cdot \mathtt{pb}^s \\ \quad \textbf{alors } \mathtt{sol}^c = \mathtt{sol}^s + 1 \end{cases} \qquad \begin{pmatrix} x \; r \; y \;\; \textit{ssi} \;\; y = 10 \cdot x \\ \mathcal{A}_r(x,z,y) = z + 1 \end{pmatrix}$$

$$\begin{cases} \textit{si } \mathtt{pb}^c = 0,1 \cdot \mathtt{pb}^s \\ \textbf{alors } \mathtt{sol}^c = \mathtt{sol}^s - 1 \end{cases} \qquad \begin{pmatrix} x \; r \; y \;\; \textit{ssi} \;\; y = 0,1 \cdot x \\ \mathcal{A}_r(x,z,y) = z - 1 \end{pmatrix}$$

La première traduit (3.4), les deux dernières traduisent (3.5).

Par exemple, si $\mathtt{pb}^c = \dfrac{0,01}{\mathtt{pb}^s}$*, alors en appliquant un chemin de similarité constitué, dans un ordre quelconque, d'une application de la première règle et de deux applications de la troisième règle, on obtient* $\mathtt{sol}^c = -\mathtt{sol}^s - 2$*.*

De façon générale, si $\mathtt{pb}^c = 10^p \cdot (\mathtt{pb}^s)^k$ *(*$p \in \mathbb{Z}$*,* $k \in \{-1,+1\}$*), alors* $\mathtt{sol}^c = p + k \cdot \mathtt{sol}^s$*. Et si le coût d'application de chaque règle est de* 1*, alors le coût d'adaptation sera* $p + \dfrac{1-k}{2}$*.*

L'avantage de cette approche est qu'elle fournira un résultat précis. Son inconvénient est qu'elle ne permet de résoudre que les problèmes de la forme $10^p \cdot (\mathtt{pb}^s)^k$ *où* $p \in \mathbb{Z}$*,* $k \in \{-1,+1\}$ *et* $\{(\mathtt{pb}^s, \mathtt{sol}^s)\}$ *est un cas source.*

L'adaptation par règle sous forme de $\dot{+}$-adaptation. On se place sous les hypothèses de l'adaptation par règles. On peut noter de façon préalable que cette approche de l'adaptation ne tient pas compte des connaissances du domaine, ce qui revient à poser $\mathtt{CD} = \mathcal{U}$. Soit $d_{\mathtt{RA}}$, la distance sur $\mathcal{U} = \mathcal{U}_{\mathtt{pb}} \times \mathcal{U}_{\mathtt{sol}}$ définie par

$$d_{\mathtt{RA}}(u_1, u_2) = \min\{\mathtt{coût}(ca) \mid ca : \text{chemin d'adaptation de } u_1 \text{ à } u_2\} \tag{3.6}$$

où par convention $\min \emptyset = +\infty$. Si on ne dispose que d'un nombre fini de règles d'adaptations (ce que l'on suppose), $\{\mathtt{coût}(ca) \mid ca : \text{chemin d'adaptation de } u_1 \text{ à } u_2\}$ est discret, le minimum est donc atteint et si $d_{\mathtt{RA}}(u_1, u_2) < +\infty$, il existe un chemin d'adaptation de u_1 à u_2 de coût $d_{\mathtt{RA}}(u_1, u_2)$. Soit $\mathtt{Source} = \{\mathtt{srce}\}$ un cas source spécifique, avec $\mathtt{srce} = (\mathtt{pb}^s, \mathtt{sol}^s)$, et $\mathtt{Cible} = \{\mathtt{pb}^c\} \times \mathcal{U}_{\mathtt{sol}}$ un cas cible avec une partie problème spécifique. Soit $\mathtt{CibleComplétée}$ définie par l'équation (3.1) (page 88) avec l'opérateur de révision $\dot{+}_{d_{\mathtt{RA}}}$ défini grâce à la « distance » $d_{\mathtt{RA}}$. Une instance $u_{\mathtt{c}} = (x_{\mathtt{c}}, y_{\mathtt{c}})$ est dans $\mathtt{CibleComplétée}$ ($u_{\mathtt{c}} \in \mathtt{CibleComplétée}$) ssi il existe un chemin d'adaptation de coût minimal de $\mathtt{srce}$ à $u_{\mathtt{c}}$, ce qui équivaut à dire que $\mathtt{sol}^c$ est obtenue par adaptation le long d'un chemin de similarité de coût minimal. Autrement dit, $y_{\mathtt{c}} \in \mathtt{Solutions}(x_{\mathtt{c}})$ où $\mathtt{Solutions}(\mathtt{pb}^c)$ est le résultat de l'adaptation par règles de $(\mathtt{pb}^s, \mathtt{sol}^s)$ à $\mathtt{pb}^c$. Donc $\mathtt{CibleComplétée} = \{\mathtt{pb}^c\} \times \mathtt{Solutions}(\mathtt{pb}^c)$. Ainsi, la résolution du problème d'adaptation considéré à l'aide des règles d'adaptation coïncide avec la résolution de ce problème d'adaptation à l'aide de l'opérateur de révision $\dot{+}_{d_{\mathtt{RA}}}$.

Une conséquence de cela est qu'une adaptation par règles, conçue pour des cas sources spéci-

fiques, peut être généralisée immédiatement pour des cas sources généraux : il suffit d'appliquer la $\dot{+}_{d_{\mathtt{RA}}}$-adaptation sur de tels cas sources. De la même façon, on peut tenir compte de la présence de connaissances du domaine ($\mathtt{CD} \subsetneq \mathcal{U}$) mais cela suppose que les chemins d'adaptation ca de l'équation (3.6) soient cohérents avec ces connaissances, autrement dit, pour tout $(\mathtt{pb}_i, \mathtt{sol}_i)$ de ca, il faut que $(\mathtt{pb}_i, \mathtt{sol}_i) \in \mathtt{CD}$.

Associer adaptation par règles et $\dot{+}$-adaptation. Les deux approches de l'adaptation présentent des inconvénients.

L'inconvénient principal de l'approche par règles (ou de la $\dot{+}_{d_{\mathtt{RA}}}$-adaptation) est qu'elle dépend fortement de la disponibilité des règles d'adaptation. Certains travaux ont pour objectif l'obtention de ces règles, que ce soit par apprentissage automatique ([Hanney et Keane, 1996]), auprès d'experts ([d'Aquin *et al.*, 2006]) ou de façon semi-automatique selon les principes de l'extraction des connaissances et de la fouille de données ([Craw *et al.*, 2006], [d'Aquin *et al.*, 2007]). Néanmoins rares (ou inexistantes) sont les applications non jouets pour lesquelles les règles d'adaptation disponibles sont suffisantes pour traiter tous les problèmes d'adaptation qui seraient jugés triviaux par un expert.

Formellement, la $\dot{+}$-adaptation comprenant l'adaptation par règles, elle ne devrait pas avoir de limitations que n'a pas cette dernière. Néanmoins, comme nous l'avons vu en section 3.2.2, pour des opérateurs de révision définis grâce à des « distances » simples, l'adaptation par révision se limite à une adaptation par copie et correction.

Pour combiner ces deux approches, une première idée est de définir une $\dot{+}^d$-adaptation réalisant un choix entre les deux approches, l'adaptation par règles, représentée par une « distance » $d_{\mathtt{RA}}$, et l'adaptation conservatrice, représentée par une distance d_0 ne gérant pas d'interactions problème-solution :

$$d(u_1, u_2) = \min\left(K_{\mathtt{RA}} \cdot d_{\mathtt{RA}}(u_1, u_2), K_0 \cdot d_0(u_1, u_2)\right) \quad \text{pour } u_1, u_2 \in \mathcal{U}$$

Les coefficients $K_{\mathtt{RA}} > 0$ et $K_0 > 0$ sont choisis pour doser la priorité entre les deux types d'adaptation. Par exemple, si $K_{\mathtt{RA}}$ est petit devant K_0, cela signifie que l'adaptation par règles, quand elle s'applique (i.e., quand $d_{\mathtt{RA}}(\mathtt{Sol_s}, \mathtt{pb^c}) < +\infty$), est généralement préférée à la $\dot{+}_{d_0}$-adaptation.

Une deuxième idée de combinaison des deux approches de l'adaptation consiste à les composer (dans le même esprit que celui de la combinaison des règles d'adaptation) : on adapte le cas source par des règles en un cas intermédiaire entre le cas source et le cas cible, puis on applique la $\dot{+}_{d_0}$-adaptation. Cela peut se modéliser par une $\dot{+}^d$-adaptation avec

$$d(u_1, u_2) = \inf_{u \in \mathcal{U}} K_{\mathtt{RA}} \cdot d_{\mathtt{RA}}(u_1, u) + K_0 \cdot d_0(u, u_2) \quad \text{pour } u_1, u_2 \in \mathcal{U}$$

(si la borne inférieure de l'expression ci-dessus n'est pas atteinte, on se contentera d'une valeur de u qui donnera une valeur proche de cette borne).

Exemple 3.3.2. *Appliquer la première combinaison dans l'exemple 3.3.1 consiste à choisir, au cas (d'adaptation) par cas (d'adaptation), une des deux approches. En l'occurrence, l'adaptation par règles, quand elle donne un résultat (i.e.,* $d_{RA}(\mathtt{pb}^s, \mathtt{pb}^c) < +\infty$*, c.-à-d.* $\mathtt{pb}^c = 10^p \cdot (\mathtt{pb}^s)^s$ *pour un* $p \in \mathbb{Z}$ *et un* $s \in \{-1, 1\}$*) est préférable. Cela revient à faire*

$$
\begin{array}{ll}
\textbf{\textit{si}} & \mathtt{pb}^c = 10^p \cdot (\mathtt{pb}^s)^s \qquad (p \in \mathbb{Z}, s \in \{-1, +1\}) \\
\textbf{\textit{alors}} & \mathtt{sol}^c = \min(p + s \cdot \mathtt{sol}^s, 0{,}44 \cdot (\mathtt{pb}^c - 1)) \\
\textbf{\textit{sinon}} & \mathtt{sol}^c = \min(\mathtt{sol}^s, 0{,}44 \cdot (\mathtt{pb}^c - 1))
\end{array}
$$

Appliquer la deuxième combinaison, sous l'hypothèse $K_{RA} \ll K_0$*, revient à appliquer les règles d'adaptation en un cas* $\{z\}$ *le plus proche du cas cible et à appliquer une copie-correction. En l'occurrence,* $z = (z_1, z_2)$ *où* $z_1 = 10^p \cdot (\mathtt{pb}^s)^s$ *avec* p *et* s *choisis de façon à ce que* z_1 *soit le plus proche possible de* $\mathtt{pb}^c$ *au sens de* d_0*. Plus formellement :*

$$
\begin{array}{l}
\mathtt{sol}^c = \min\left(p + s \cdot \mathtt{sol}^s, 0{,}44 \cdot (\mathtt{pb}^c - 1)\right) \\
\textit{où } (p, s) \textit{ minimise } K_{RA}\left(p + \dfrac{1-s}{2}\right) + K_0 \left|\mathtt{pb}^c - 10^p \cdot (\mathtt{pb}^s)^s\right| \\
\textit{sur } \mathbb{Z} \times \{-1, +1\}
\end{array}
$$

3.3.2 L'adaptation différentielle et l'adaptation par révision

Cette section reprend les travaux présentés dans [Cojan et Lieber, 2011a].

Présentation de l'adaptation différentielle

L'adaptation différentielle s'inspire de l'équation de calcul différentiel suivante :

$$dy_j = \sum_i \frac{\partial y_j}{\partial x_i} \cdot dx_i \tag{3.7}$$

appliquée à une fonction différentiable $f : x \in \mathbb{R}^m \mapsto y \in \mathbb{R}^n$. Cette équation peut servir à calculer une approximation du premier ordre de f en interprétant les différentielles dx_i et dy_j comme des petites différences. Ce calcul peut être effectué grâce à la connaissance $\mathtt{CA} = \left\{\frac{\partial y_j}{\partial x_i}\right\}_{ij}$. $\mathtt{CA}$ est l'abréviation de *connaissances d'adaptation* : cette approximation du premier ordre peut être vue comme une adaptation. En effet, si $\mathcal{U}_{\mathtt{pb}} = \mathbb{R}^m$, $\mathcal{U}_{\mathtt{sol}} = \mathbb{R}^n$, $\mathtt{Source} = \{u_{\mathtt{s}}\}$ est un cas source spécifique avec $u_{\mathtt{s}} = (x^{\mathtt{s}}, y^{\mathtt{s}})$ et $y^{\mathtt{s}} = f(x^{\mathtt{s}})$, alors (3.7) permet de calculer $\mathtt{CibleComplétée} = \{(x^{\mathtt{c}}, y^{\mathtt{c}})\}$ avec $\{x^{\mathtt{c}}\} = \mathtt{Pb_c}$ et $y_j^{\mathtt{c}} = y_j^{\mathtt{s}} + \sum_i \frac{\partial y_j}{\partial x_i} \cdot (x_i^{\mathtt{c}} - x_i^{\mathtt{s}})$.

L'adaptation différentielle est présentée en détail dans [Fuchs *et al.*, 2006], nous proposons ici une version simplifiée. Cette simplification consiste à généraliser le calcul différentiel en substituant $\mathbb{R}$ par (éventuellement) d'autres ensembles de valeurs : $\mathcal{U}_{\mathtt{pb}} = V_1 \times \ldots \times V_m$, $\mathcal{U}_{\mathtt{sol}} = W_1 \times \ldots \times W_n$, où les V_i et les W_j sont différents types d'ensembles de valeurs. Un type $\mathcal{T}$ peut être numérique (p. ex., $\mathcal{T} = \mathbb{R}$ or $\mathcal{T} = \mathbb{Z}$) ou non (p. ex., types énumérés, ensembles des

parties d'un ensemble fini, etc.), sous condition de disposer d'une opération, notée $+$, telle que $(\mathcal{T},+)$ soit un groupe commutatif [12]. On considère donc $m+n$ groupes commutatifs $(V_i,+)$ et $(W_j,+)$ qui ne sont pas nécessairement différents. Soit $+$ définie sur $\mathcal{U}_{\texttt{pb}}$ (resp., sur $\mathcal{U}_{\texttt{sol}}$) par $a+b=(a_1+b_1,\ldots,a_m+b_m)$ (resp., $a+b=(a_1+b_1,\ldots,a_n+b_n)$). $(\mathcal{U}_{\texttt{pb}},+)$ et $(\mathcal{U}_{\texttt{sol}},+)$ sont des groupes commutatifs. Étant donné un cas spécifique $\texttt{Source}=\{(x^{\texttt{s}},y^{\texttt{s}})\}$ et un problème cible spécifique $\texttt{Pb}_{\texttt{c}}=\{x^{\texttt{c}}\}$, dx et dy dénotant respectivement $x^{\texttt{c}}-x^{\texttt{s}}$ et $y^{\texttt{c}}-y^{\texttt{s}}$.

L'adaptation différentielle consiste à ① calculer dy à partir de dx et ② calculer $y^{\texttt{c}}=y^{\texttt{s}}+dy$. L'étape ② est simple. Soit $D_{x^{\texttt{s}}}$ la fonction qui calcule l'étape ① : $D_{x^{\texttt{s}}}:dx\mapsto dy$. Dans le cadre du calcul différentiel numérique, $D_{x^{\texttt{s}}}$ est une transformation linéaire de matrice $\left[\frac{\partial y_j}{\partial x_i}(x^{\texttt{s}})\right]_{ij}$. Dans le cadre de l'adaptation différentielle générale, $D_{x^{\texttt{s}}}$ correspond aux fonctions $\frac{\partial y_j}{\partial x_i}(x^{\texttt{s}})$ qui associent à chaque $dx_i\in V_i$ une contribution $d_iy_j\in W_j$ à dy_j :

$$\begin{aligned}\frac{\partial y_j}{\partial x_i}(x^{\texttt{s}}):dx_i\mapsto d_iy_j \qquad dy_j&=\sum_i d_iy_j\\ D_{x^{\texttt{s}}}((dx_1,\ldots,dx_m))&=(dy_1,\ldots,dy_n)\\ y^{\texttt{c}}&=y^{\texttt{s}}+D_{x^{\texttt{s}}}(x^{\texttt{c}}-x^{\texttt{s}})\end{aligned}\tag{3.8}$$

On suppose que $\left(\frac{\partial y_j}{\partial x_i}(x^{\texttt{s}})\right)(0)=0$: si le descripteur du problème x_i ne change pas ($dx_i=0$) alors il ne contribue pas au changement du descripteur de la solution dy_j ($d_iy_j=0$). Cela entraîne que $D_{x^{\texttt{s}}}(0)=0$.

Exemple 3.3.3. *On reprend ici l'exemple 3.3.1. On cherche à approcher la fonction logarithme décimal par un système de* RÀPC. $\mathcal{U}_{\texttt{pb}}=\mathbb{R}_+^*$, $\mathcal{U}_{\texttt{sol}}=\mathbb{R}$, *un cas source est un couple* $(x,y)\in\mathcal{U}$ *tel que* $y\simeq\log x$. *Les connaissances du domaine prises en compte sont* $\log x\leqslant 0{,}44\cdot(x-1)$, *ce qui veut dire que :*

$$\texttt{CD}=\{(x,y)\in\mathcal{U}\mid y\leqslant 0{,}44\cdot(x-1)\}$$

Pour cet exemple, $D_{x^{\texttt{s}}}:dx\in\mathbb{R}\mapsto\varrho^{\texttt{s}}\cdot dx\in\mathbb{R}$, *avec* $\varrho^{\texttt{s}}\in\mathbb{R}$. *Ainsi, l'adaptation est simplement définie par*

$$y^{\texttt{c}}=y^{\texttt{s}}+\varrho^{\texttt{s}}\cdot(x^{\texttt{c}}-x^{\texttt{s}})\tag{3.9}$$

(l'adaptation se fait en suivant une droite et $\varrho^{\texttt{s}}$ *est la pente de cette droite). La figure 3.1 illustre cette adaptation. Le choix « optimal » de* $\varrho^{\texttt{s}}$ *est* $\varrho^{\texttt{s}}=\frac{d\log x^{\texttt{s}}}{dx^{\texttt{s}}}=\frac{1}{x^{\texttt{s}}\cdot\ln 10}$ *([13]). Le choix* $\varrho^{\texttt{s}}=0$ *correspond à l'adaptation par copie :* $y^{\texttt{c}}=y^{\texttt{s}}$.

Pour comparer avec une adaptation conservatrice, on considère l'opérateur de révision $\dot{+}^d$ *défini grâce à* d, *la distance* L_1 *de la base canonique de* $\mathbb{R}\times\mathbb{R}$ *(pour* $(x^1,y^1),(x^2,y^2)\in\mathcal{U}$, $d((x^1,y^1),(x^2,y^2))=|x^2-x^1|+|y^2-y^1|$*). La* $\dot{+}^d$*-adaptation de* `Source` *pour résoudre* `Cible`

12. Un groupe commutatif $(\mathcal{T},+)$ est tel que $+$ est une opération interne associative et commutative, $(\mathcal{T},+)$ a un élément neutre, noté 0, et chaque $a\in\mathcal{T}$ a un élément inverse noté $-a$ ($a+(-a)=0$). $b-a$ est une notation pour $b+(-a)$. Dans [Fuchs *et al.*, 2006] les hypothèses sur $(\mathcal{T},+)$ sont plus faibles.

13. C'est le choix optimal dans le sens où $y^{\texttt{c}}-(y^{\texttt{s}}+\varrho^{\texttt{s}}\cdot(x^{\texttt{c}}-x^{\texttt{s}}))=_{x^{\texttt{c}}\to x^{\texttt{s}}}o(x^{\texttt{c}}-x^{\texttt{s}})$ ssi $\varrho^{\texttt{s}}=\frac{1}{x^{\texttt{s}}\cdot\ln 10}$, ce qui est une conséquence directe de la définition des dérivées.

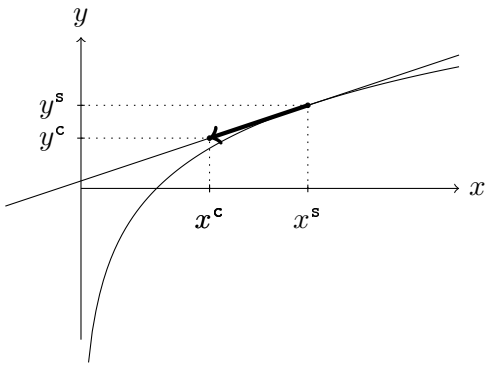


FIGURE 3.1 – Illustration de l'adaptation différentielle sur l'exemple. Elle consiste simplement à suivre la droite de pente ϱ^s (le choix optimal de ϱ^s est choisi ici : la droite est la tangente à la courbe en x^c). Aucune connaissance du domaine n'est considérée ($\mathtt{CD} = \mathcal{U}$).

procède par copie-correction de y^s : si (x^c, y^s) est cohérent avec $\mathtt{CD}$, c'est-à-dire si $y^s \leqslant 0{,}44 \cdot (x^c - 1)$, alors $y^c = y^s$, sinon cette solution est corrigée, $y^c = 0{,}44 \cdot (x^c - 1)$. Donc le résultat de l'adaptation conservatrice est :

$$y^c = \min(y^s, x^c - 1) \tag{3.10}$$

La figure 3.2 illustre cet exemple.

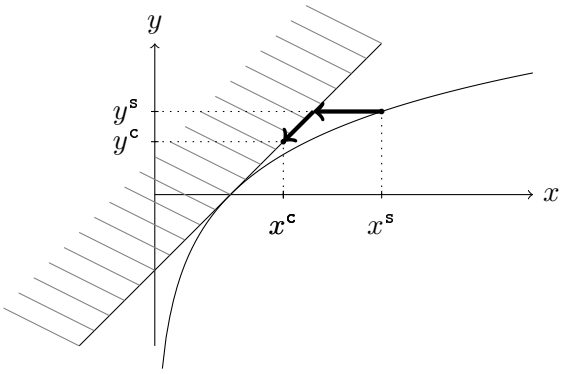


FIGURE 3.2 – Illustration sur l'exemple de l'adaptation par révision suivant un opérateur défini à l'aide de la distance L_1 de la base canonique. Cette adaptation correspond à une adaptation par copie et correction : tant que la copie est cohérente avec les connaissances du domaine elle est appliquée (cf. la flèche horizontale à droite), quand elle ne l'est plus, elle est corrigée suivant les connaissances du domaine (cf. l'autre flèche).

L'adaptation différentielle vue comme une adaptation par révision

La proposition suivante montre que l'adaptation différentielle peut être simulée par l'adaptation par révision :

Proposition 3.3.1. *Étant donné les connaissances d'adaptation différentielles exprimées par la fonction D_{x^s} paramétrée par $x^s \in \mathcal{U}_{pb}$, il existe un opérateur de révision $\dot{+}$ tel que le résultat de*

l'adaptation différentielle coïncide avec la $\dot{+}$-adaptation sans connaissances du domaine.

Plus précisément, soient $\mathtt{dist}_{pb}$ et $\mathtt{dist}_{sol}$ respectivement des « distances » sur $\mathcal{U}_{pb}$ et $\mathcal{U}_{sol}$, et soit d la « distance » (au sens des préliminaires, section 1.2.2, c'est-à-dire qu'elle respecte la séparation) telle que pour $u_s, u_c \in \mathcal{U}$, avec $u_s = (x^s, y^s)$ et $u_c = (x^c, y^c)$,

$$d(u_s, u_c) = \mathtt{dist}_{pb}(x^s, x^c) + \mathtt{dist}_{sol}(y^s, y^c - D_{x^s}(x^c - x^s)) \tag{3.11}$$

La $\dot{+}^d$-adaptation de $\mathtt{Source} = \{(x^s, y^s)\}$ pour résoudre $\mathtt{Cible} = \{x^c\} \times \mathcal{U}_{sol}$ donne donc $y^c = y^s + D_{x^s}(x^c - x^s)$ qui est le résultat de l'adaptation différentielle.

Démonstration. Montrons d'abord que d est une « distance ». Si $d(u^s, u^c) = 0$ alors (a) $\mathtt{dist}_{pb}(x^s, x^c) = 0$ et (b) $\mathtt{dist}_{sol}(y^s, y^c - D_{x^s}(x^c - x^s)) = 0$. Puisque $\mathtt{dist}_{pb}$ est une « distance », (a) entraîne que $x^s = x^c$. Donc $D_{x^s}(x^c - x^s) = 0$, et (b) entraîne que $\mathtt{dist}_{sol}(y^s, y^c - 0) = 0$. Puisque $\mathtt{dist}_{sol}$ est une « distance », $y^s = y^c$. Finalement, $u^s = u^c$ et d est une « distance ».

Soit y^c le résultat de la $\dot{+}^d$-adaptation de $\{(x^s, y^s)\}$ par l'instance du problème cible x^c. y^c minimise donc $\Delta = d((x^s, y^s), (x^c, y^c))$, où x^s, y^s et x^c sont fixés. (3.11) entraîne que $\Delta \geqslant \mathtt{dist}_{pb}(x^s, x^c)$. $\Delta = \mathtt{dist}_{pb}(x^s, x^c)$ n'est atteint que si $\mathtt{dist}_{sol}(y^s, y^c - D_{x^s}(x^c - x^s)) = 0$, ce qui équivaut, puisque $\mathtt{dist}_{sol}$ est une « distance », à $y^s = y^c - D_{x^s}(x^c - x^s)$, soit $y^c = y^s + D_{x^s}(x^c - x^s)$. Donc, si y^c est le résultat de la $\dot{+}^d$-adaptation avec d définie par (3.11) alors y^c est aussi le résultat de l'adaptation différentielle. La preuve de l'implication inverse est directe. □

Ce résultat enrichit mutuellement les deux approches de l'adaptation :

- L'adaptation par révision généralise l'adaptation différentielle dans le sens où elle permet de considérer des cas sources qui ne sont pas nécessairement spécifiques et, surtout, elle permet de prendre en compte les connaissances du domaine pour l'adaptation différentielle.
- L'adaptation par révision dépend du choix d'un opérateur de révision et les connaissances d'adaptation associées à l'adaptation différentielle peuvent servir à définir un opérateur de révision pertinent pour effectuer l'adaptation.

Exemple 3.3.4 (Suite de l'exemple 3.3.3). *Avec des « distances » $\mathtt{dist}_{pb}$ et $\mathtt{dist}_{sol}$ de la forme $|\cdot - \cdot|$ (c.-à-d. les « distances » L_1 sur les bases canoniques de $\mathcal{U}_{pb}$ et $\mathcal{U}_{sol}$), l'équation* (3.11) *donne*

$$d(u_s, u_c) = |x^c - x^s| + |y^c - (y^s + \varrho^s \cdot (x^c - x^s))| \tag{3.12}$$

Donc, sans connaissances du domaine, la $\dot{+}^d$-adaptation donne le résultat (3.9) *et avec les connaissances du domaine cette adaptation donne :*

$$y^c = \min(y^s + \varrho^s \cdot (x^c - x^s), x^c - 1)$$

La situation $\varrho^s = 0$ correspond à l'adaptation conservatrice (cf. équation (3.10)*). Plus généralement, la « distance » d définie par* (3.12) *est une « distance » L_1 dans la base $(\overrightarrow{e_1}, \overrightarrow{e_2})$ de $\mathbb{R}^2$ avec*

$\overrightarrow{e_1} = \overrightarrow{\varepsilon_1} + \varrho^s \overrightarrow{\varepsilon_2}$ *et* $\overrightarrow{e_2} = \overrightarrow{\varepsilon_2}$ *où* $(\overrightarrow{\varepsilon_1}, \overrightarrow{\varepsilon_2})$ *est la base canonique (*$\overrightarrow{\varepsilon_1} = (1,0)$ *et* $\overrightarrow{\varepsilon_2} = (0,1)$*) : la pente de* $\overrightarrow{e_1}$ *dans* $(\overrightarrow{\varepsilon_1}, \overrightarrow{\varepsilon_2})$ *vaut* ϱ^s*.* $(\overrightarrow{e_1}, \overrightarrow{e_2})$ *est une base* orthogonale *(par rapport à* $(\overrightarrow{\varepsilon_1}, \overrightarrow{\varepsilon_2})$*) ssi* $\varrho^s = 0$ *(ce qui correspond à* $(\overrightarrow{e_1}, \overrightarrow{e_2}) = (\overrightarrow{\varepsilon_1}, \overrightarrow{\varepsilon_2})$*). Ainsi,* $\dot{+}^d$ *avec la « distance »* d *définie par* (3.12) *est orthogonale (avec le sens de la définition 3.2.2, page 90) ssi* $(\overrightarrow{e_1}, \overrightarrow{e_2})$*, la base dans laquelle* d *est* L_1*, est orthogonale. Cela justifie* a posteriori *l'adjectif « orthogonal » de l'opérateur de révision.*

La figure 3.3 illustre cette adaptation.

Remarque 3.3.1. *Dans cet exemple, cette adaptation coïncide soit avec l'adaptation différentielle soit avec l'adaptation conservatrice. Ce n'est pas toujours le cas. Par exemple, supposons qu'on cherche à approcher par un système de* RÀPC *la valeur de la fonction sinus en* $\pi/2$ *en connaissant sa valeur en* 0*.* $\mathcal{U} = \mathcal{U}_{pb} \times \mathcal{U}_{sol}$ *avec* $\mathcal{U}_{pb} = \mathcal{U}_{sol} = \mathbb{R}$*,* $\mathtt{Source} = \{(0,0)\}$ *et le problème cible à résoudre est* $\mathtt{Pb_c} = \pi/2$*, on s'appuie sur la valeur de la dérivée du sinus en* 0 *qui vaut* 1 *et sur les connaissances du domaine qui établissent que la fonction sinus est à valeurs dans* $[-1,1]$ *:*

$$\mathtt{CD} = \{(x,y) \in \mathcal{U} \mid -1 \leqslant y \leqslant 1\}$$

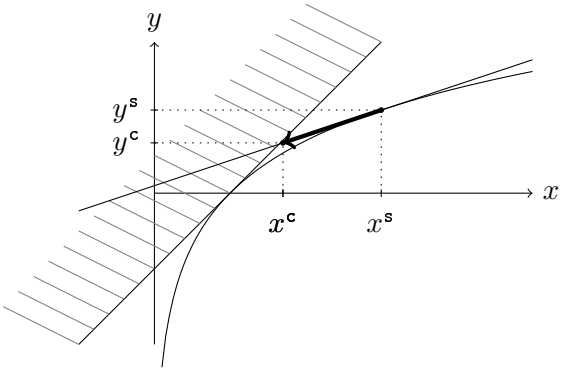
L'adaptation de $\mathtt{Source}$ *pour résoudre* $\mathtt{Pb_c}$ *donne* $\mathtt{CibleComplétée} = \{(x^c, y^c)\}$ *avec :*

$$x^c = \pi/2 \quad \text{et} \quad \begin{cases} y^c = 0 & \text{avec l'adaptation conservatrice.} \\ y^c = \pi/2 & \text{avec l'adaptation différentielle (sans tenir compte de } \mathtt{CD}\text{).} \\ y^c = 1 & \text{avec la } \dot{+}^d\text{-adaptation pour } d \text{ définie suivant (3.12).} \end{cases}$$

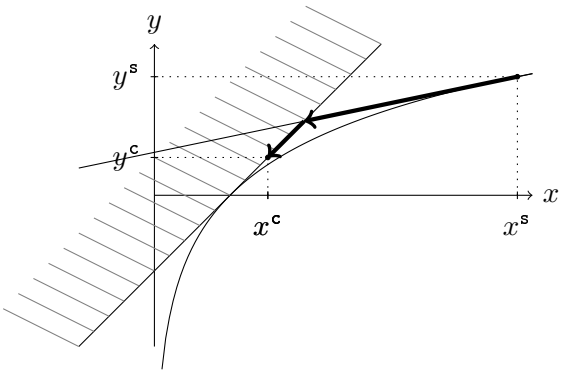
De l'adaptation conservatrice à l'adaptation différentielle optimale

Cette section traite d'adaptations qui sont « entre » l'adaptation conservatrice (AC) et l'adaptation différentielle optimale (ADO), étant donné des connaissances du domaine. L'ADO est une adaptation différentielle avec des connaissances complètes sur les $\left[\frac{\partial y_j}{\partial x_i}(x^s)\right]_{ij}$, c.-à-d., un choix de D_{x^s} donnant les résultats optimaux pour l'adaptation différentielle suivant des critères prédéfinis (pour l'exemple, ce choix optimal est expliqué dans la note 13 en bas de page 99). L'adaptation conservatrice correspond, elle, au choix arbitraire $\frac{\partial y_j}{\partial x_i}(x^s) = 0$ (pour chaque x^s, i et j).

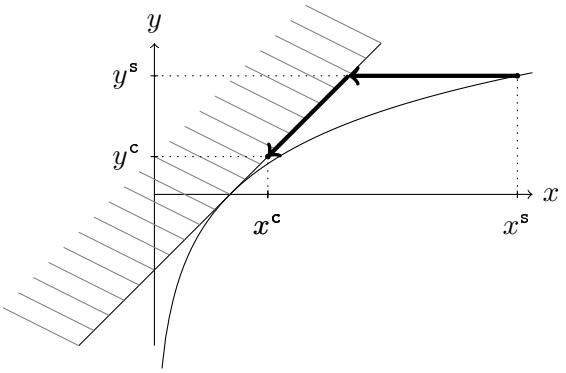
D'un côté, l'ADO peut être préférée à l'AC, puisqu'elle donne la meilleure adaptation parmi les adaptations différentielles. D'un autre côté, l'AC ne demande par d'effort d'acquisition de connaissances d'adaptation ni d'apprentissage alors que l'ADO en demande beaucoup. L'idée d'adaptations intermédiaires entre AC et ADO a donc émergé, elle consiste à définir une procédure d'acquisition de connaissances d'adaptation, pour affiner graduellement l'opérateur d'adaptation en partant de l'AC comme état initial, et en convergeant vers l'ADO. Cette procédure pourrait être implantée suivant les principes de l'acquisition opportuniste de connaissances d'adaptation (voir p. ex. [Cordier *et al.*, 2008]) ou de l'extraction opportuniste de connaissances d'adaptation (voir p. ex. [Badra *et al.*, 2009b]).



(a)



(b)



(c)

FIGURE 3.3 – Deux illustrations de l'adaptation par révision suivant un opérateur de révision intégrant des connaissances d'adaptation différentielle. Dans la situation (a), le résultat de l'adaptation différentielle est cohérent avec les connaissances du domaine, il n'y a donc pas besoin de le corriger (cela est équivalent à l'adaptation de la figure 3.1). Dans la situation (b), le résultat de l'adaptation différentielle n'est pas cohérent avec les connaissances du domaine, il faut donc le corriger. Le résultat de cette adaptation est équivalent à l'adaptation conservatrice représentée en (c).

3.4 Combinaison de cas par fusion de connaissances

Cette section reprend les travaux présentés dans [Cojan et Lieber, 2009a]. Il s'agit à l'origine de l'idée proposée par Pierre Marquis en janvier 2007. En utilisant la fusion contrainte de connaissances à la place de la révision, on peut généraliser l'adaptation par la révision en une combinaison de cas, où le cas cible n'est pas résolu à partir d'un seul mais de plusieurs cas sources.

3.4.1 Définition et propriétés

Soit $\mathtt{SCS} = \{\mathtt{Source}_1, \ldots, \mathtt{Source}_k\}$ un ensemble de cas sources, et $\triangle$ un opérateur de fusion contrainte sur $\mathcal{U}$. La $\triangle$-combinaison des cas sources SCS pour résoudre le cas Cible est obtenue en fusionnant les contextes sources sous contrainte du contexte cible :

$$\begin{aligned} \mathtt{CibleComplétée} &= \mathtt{CC}^{\mathtt{CD}}_{\triangle}(\{\mathtt{Source}_1, \ldots, \mathtt{Source}_k\}, \mathtt{Cible}) \quad \text{où} \\ \mathtt{CC}^{\mathtt{CD}}_{\triangle}(\{\mathtt{Source}_1, \ldots, \mathtt{Source}_k\}, \mathtt{Cible}) &= \triangle_{\mathtt{CD} \cap \mathtt{Cible}}(\{\mathtt{CD} \cap \mathtt{Source}_1, \ldots, \mathtt{CD} \cap \mathtt{Source}_k\}) \end{aligned} \tag{3.13}$$

La combinaison de cas par fusion est une généralisation de l'adaptation par révision : si $k = \mathrm{card}(\mathtt{SCS}) = 1$, alors la $\triangle$-combinaison de $\mathtt{SCS} = \{\mathtt{Source}_1\}$ pour résoudre Cible correspond à la $\dot{+}$-adaptation de $\mathtt{Source}_1$ pour résoudre Cible, où $\dot{+}$ est défini par l'égalité (2.13), page 72.

Propriétés de la combinaison par fusion contrainte

Comme en section 3.2.1 où les postulats de la révision sont associés à des propriétés attendues de l'adaptation, nous pouvons identifier certaines correspondances entre les postulats de la fusion et les propriétés attendues du résultat CibleComplétée de la fusion des cas $\mathtt{Source}_1, \ldots, \mathtt{Source}_n$ pour résoudre le cas Cible.

CibleComplétée doit résoudre Cible, ce qui se traduit par une spécialisation de Cible dans notre formalisme, cela correspond (modulo les connaissances du domaine CD) au postulat ($\triangle$-0).

($\triangle$-1) établit que CibleComplétée doit être cohérent à moins que Cible lui-même ne soit lui-même pas cohérent avec CD. C'est en quelque sorte une exigence de résultat : la combinaison doit aboutir à un résultat qui ait du sens.

Selon [Konieczny, 1999], les postulats ($\triangle$-5), ($\triangle$-6), ($\triangle$-7) et ($\triangle$-8) caractérisent le changement minimal des connaissances apportées par les sources d'information.

Comme les postulats (G5) et (G6) de la révision, les postulat ($\triangle$-7) et ($\triangle$-8) correspondent à la caractérisation de la minimisation d'un pré-ordre total (voir les propositions 1.2.1 et 1.2.2, page 9). Intuitivement, cela signifie qu'une fois qu'un compromis a été trouvé entre les solutions des cas sources, ce compromis est modifié au minimum pour résoudre Cible.

Les postulats ($\triangle$-5) et ($\triangle$-6) établissent que si les solutions obtenues par la combinaison de deux ensembles de cas sources sont cohérentes, alors la solution globale obtenue n'a pas besoin d'être modifiée.

Le postulat ($\triangle$-2) correspond à la situation extrême ou les solutions partielles des différents cas sources sont cohérentes entre elles avec `Cible` et avec `CD`, `CibleComplétée` est alors obtenu en assemblant ces solutions partielles sans qu'il ne soit nécessaire de leur apporter de changement.

Le postulat ($\triangle$-4) établit que la solution proposée par un cas source ne doit pas être arbitrairement favorisée par rapport aux autres. Si deux cas $\texttt{Source}_1$ et $\texttt{Source}_2$ sont séparément cohérents avec un cas `Cible`, c'est-à-dire qu'ils donnent chacun une solution à `Cible` sans effort d'adaptation, alors la combinaison de $\texttt{Source}_1$ et $\texttt{Source}_2$ pour résoudre `Cible` ne peut pas sélectionner l'une des deux solutions obtenues. Ce postulat est plus discutable, on pourrait envisager que lorsque deux alternatives sont également bonnes, on n'en retienne qu'une seule afin d'obtenir une solution plus précise.

3.4.2 Exemple de combinaison de cas

On reprend l'exemple de l'introduction où Garuda cherche une recette de mousse au chocolat sans œufs. Il ne trouve pas de recette satisfaisant ces critères mais il trouve trois recettes dont il peut s'inspirer, une recette de mousse au chocolat traditionnelle, avec des œufs : $\texttt{Source}_1$, une recette de chantilly sans œufs : $\texttt{Source}_2$ et une recette de crème au chocolat sans œufs : $\texttt{Source}_3$. Les recettes sont représentées ici dans un formalisme attributs-valeurs simples avec un attribut par ingrédient : `œuf`, `chocolat`, `crème`, `sucre` et `soja` dont les valeurs représentent les quantités contenues dans la recette des ingrédients respectifs. Un attribut supplémentaire, `vMousse`, représente le volume de mousse obtenue dans la recette, les trois attributs `estMousseux`, `contientŒuf` et `contientChocolat` dont les valeurs sont booléennes indiquent respectivement si la recette prépare un plat mousseux, un plat qui contient de l'œuf et un plat qui contient du chocolat.

Les attributs sont numérotés $\texttt{a}_1, \ldots, \texttt{a}_9$ pour faciliter leur énumération, les domaines associés à ces attributs sont respectivement $\mathcal{U}_1, \ldots, \mathcal{U}_9$. Pour éviter les problèmes de minima non atteints lors du calcul de la fusion (voir section 2.3), on prendra des espaces discrets (p.ex. $0,1 \cdot \mathbb{N} = \{0,1 \cdot m \mid m \in \mathbb{N}\}$) plutôt que $\mathbb{R}$. La table 3.1 donne la représentation de chaque recette dans ce formalisme. Si la case correspondant à un attribut contient le symbole '_', cela signifie qu'il n'y a pas de contrainte sur les valeurs de cet attribut, il peut prendre toutes les valeurs de son domaine.

Les connaissances du domaine établissent des relations entre les attributs `estMousseux`, `contientŒuf`, `contientChocolat` et `vMousse` et les attributs représentant les quantités d'ingrédients :

$$\texttt{CD} = R_{\texttt{vMousse}} \cap R_{\texttt{estMousseux}} \cap R_{\texttt{contientŒuf}} \cap R_{\texttt{contientChocolat}}$$

où $R_{\texttt{vMousse}}$ établit que l'on peut obtenir au plus 200 ml de mousse par œuf et 225 ml de mousse pour 170 ml de soja ($225/170 \simeq 1,3$) :

$$R_{\texttt{vMousse}} = \{(_, _, _, \texttt{vMousse}, \texttt{œuf}, \texttt{soja}, _, _, _) \mid \texttt{vMousse} \leqslant 200 \times \texttt{œuf} + 1,3 \times \texttt{soja}\}$$

		Cible	$\texttt{Source}_1$	$\texttt{Source}_2$	$\texttt{Source}_3$
$a_1 = \texttt{estMousseux}$	$\mathcal{U}_1 = \mathbb{B}$	Vrai	Vrai	Vrai	Faux
$a_2 = \texttt{contientŒuf}$	$\mathcal{U}_2 = \mathbb{B}$	Faux	Vrai	Faux	Faux
$a_3 = \texttt{contientChocolat}$	$\mathcal{U}_3 = \mathbb{B}$	Vrai	Vrai	Faux	Vrai
$a_4 = \texttt{vMousse}$ (ml)	$\mathcal{U}_4 = 0,1 \cdot \mathbb{N}$	_	200	225	0
$a_5 = \texttt{œuf}$	$\mathcal{U}_5 = \mathbb{N}$	_	1	0	0
$a_6 = \texttt{soja}$ (ml)	$\mathcal{U}_6 = 0,1 \cdot \mathbb{N}$	_	0	170	0
$a_7 = \texttt{chocolat}$ (g)	$\mathcal{U}_7 = 0,1 \cdot \mathbb{N}$	_	35	0	25
$a_8 = \texttt{crème}$ (g)	$\mathcal{U}_8 = 0,1 \cdot \mathbb{N}$	_	10	0	125
$a_9 = \texttt{sucre}$ (g)	$\mathcal{U}_9 = 0,1 \cdot \mathbb{N}$	_	65	50	_

TABLE 3.1 – Représentation du cas `Cible` correspondant à la recherche d'une recette de mousse au chocolat sans œufs et des recettes $\texttt{Source}_1$, $\texttt{Source}_2$ et $\texttt{Source}_3$ utilisées pour déterminer une solution à `Cible`. $\texttt{Source}_1$ est une recette de mousse au chocolat traditionnelle, $\texttt{Source}_2$ est une recette de chantilly sans œufs et $\texttt{Source}_3$ est une recette de crème au chocolat sans œufs.

$R_{\texttt{estMousseux}}$, $R_{\texttt{contientŒuf}}$ et $R_{\texttt{contientChocolat}}$ établissent que `estMousseux` (resp. `contientŒuf` et `contientChocolat`) sont `Vrai` seulement lorsque la recette est mousseuse (resp. contient des œufs et du chocolat) :

$$R_{\texttt{estMousseux}} = \{(\texttt{estMousseux}, _, _, \texttt{vMousse}, _, _, _, _, _) \mid \texttt{estMousseux} = \texttt{Vrai} \quad \text{ssi} \quad \texttt{vMousse} \neq 0\}$$

$$R_{\texttt{contientŒuf}} = \{(_, \texttt{contientŒuf}, _, _, \texttt{œuf}, _, _, _, _) \mid \texttt{contientŒuf} = \texttt{Faux} \quad \text{ssi} \quad \texttt{œuf} = 0\}$$

$$R_{\texttt{contientChocolat}} = \{(_, _, \texttt{contientChocolat}, _, _, _, \texttt{chocolat}, _, _) \mid \texttt{contientChocolat} = \texttt{Faux} \quad \text{ssi} \quad \texttt{chocolat} = 0\}$$

On considère la « distance » d sur $\mathcal{U}$ définie à l'aide des « distances » d_i sur les espaces $\mathcal{U}_i$, pour $x, y \in \mathcal{U}$ avec $x = (x_1, \ldots, x_9)$ et $y = (y_1, \ldots, y_9)$:

$$d(x, y) = \sum_{i=1}^{9} w_i \cdot d_i(x_i, y_i)$$

où les d_i sont les « distances » suivantes sur les espaces $\mathcal{U}_i$:

$$d_i(x_i, y_i) = \begin{cases} 0 & \text{si } x_i = y_i \\ 1 & \text{sinon} \end{cases} \qquad \text{pour } 1 \leqslant i \leqslant 3$$

$$d_i(x_i, y_i) = |y_i - x_i| \qquad \text{pour } 4 \leqslant i \leqslant 9$$

Avec l'opérateur de fusion $\triangle^{d,\Sigma}$ défini en section 2.7.2, et des coefficients w_i qui donnent plus d'importance à la préservation du volume de mousse qu'à celui des ingrédients[14] :

14. Le facteur 3 correspond au nombre de cas sources. Lors de la combinaison de k cas sources, pour assurer

$w_4 > 3 \cdot (w_5 + w_6)$. La combinaison des cas $\texttt{Source}_1$, $\texttt{Source}_2$ et $\texttt{Source}_3$ donne le résultat $\texttt{CibleComplétée}_1 = \texttt{CC}^{\texttt{CD}}_{\triangle}(\{\texttt{Source}_1, \texttt{Source}_2, \texttt{Source}_3\}, \texttt{Cible})$ représenté dans le tableau 3.2. Comme aucune contrainte ne lie les attributs `chocolat`, `crème` et `sucre` aux autres attributs, la minimisation de d^{Σ} est atteinte lorsque chacun de ces attributs minimise une fonction $x \mapsto |x-a|+|x-b|+|x-c|$ avec $a \leqslant b \leqslant c$, dont le minimum est atteint pour $x = b$. Si on ne tient pas compte des contraintes de `CD`, `vMousse` et `soja` correspondent aussi aux minima d'une fonction de ce type, la condition $w_4 > 3 \cdot (w_5 + w_6)$ donne priorité à la minimisation du changement de `vMousse`, donc `vMousse` $= 200$, `soja` prend alors la plus petite valeur qui lui permette de satisfaire `CD`. Le volume de mousse obtenu est proche de ceux des recettes $\texttt{Source}_1$ et $\texttt{Source}_2$ et pour obtenir ce volume de mousse, compte tenue de `CD` et du fait que l'on ne peut pas utiliser d'œufs, on a ajouté le soja nécessaire (165 ml de soja). Les quantités de chocolat et de crème sont proches de celles des recettes $\texttt{Source}_1$ et $\texttt{Source}_3$. Comme attendu, la recette obtenue s'inspire de la recette de chantilly sans œufs ($\texttt{Source}_2$) pour la mousse, et de la recette de crème au chocolat sans œufs ($\texttt{Source}_3$) pour le chocolat et la crème.

Cependant, l'opérateur de fusion $\triangle^{d,\Sigma}$ est sensible à la répartition des connaissances entre les BC fusionnées, [Konieczny, 1999] qualifie cette stratégie de fusion de « majoritaire » avec l'image du vote où le nombre de voix détermine l'adoption d'un choix. Par exemple, la $\triangle^{d,\Sigma}$-combinaison des multi-ensembles de cas $\{\texttt{Source}_2, \texttt{Source}_3\}$ et $\{\texttt{Source}_2, \texttt{Source}_3, \texttt{Source}_3\}$ donnent des résultats différents, respectivement $\texttt{CibleComplétée}_2$ et $\texttt{CibleComplétée}_3$ représentés dans le tableau 3.2. Cela implique que les aspects des solutions sources retenues sont celles qui sont défendues par suffisamment de cas sources, la répartition des cas sources est donc importante. Nous verrons en perspectives (section 3.4.5) une piste pour définir un opérateur de fusion plus intéressant pour la combinaison de cas.

	$\texttt{CibleComplétée}_1$	$\texttt{CibleComplétée}_2$	$\texttt{CibleComplétée}_3$
$\texttt{a}_1 = \texttt{estMousseux}$	Vrai	Vrai	Vrai
$\texttt{a}_2 = \texttt{contientŒuf}$	Faux	Faux	Faux
$\texttt{a}_3 = \texttt{contientChocolat}$	Vrai	Vrai	Vrai
$\texttt{a}_4 = \texttt{vMousse}$ (ml)	200	$0 \leqslant \texttt{a}_4 \leqslant 200$	$0, 1$
$\texttt{a}_5 = \texttt{œuf}$	0	0	0
$\texttt{a}_6 = \texttt{soja}$ (ml)	165	$0 \leqslant \texttt{a}_6 \leqslant 170$	$0, 1$
$\texttt{a}_7 = \texttt{chocolat}$ (g)	25	$0 \leqslant \texttt{a}_7 \leqslant 25$	25
$\texttt{a}_8 = \texttt{crème}$ (g)	10	$0 \leqslant \texttt{a}_8 \leqslant 125$	125
$\texttt{a}_9 = \texttt{sucre}$ (g)	$50 \leqslant \texttt{a}_9 \leqslant 60$	$0 \leqslant \texttt{a}_9 \leqslant 50$	50

TABLE 3.2 – Résultats de la combinaison de cas suivant l'opérateur $\triangle^{d,\Sigma}$. $\texttt{CibleComplétée}_1$ est obtenu par combinaison des cas $\texttt{Source}_1$, $\texttt{Source}_2$ et $\texttt{Source}_3$, $\texttt{CibleComplétée}_2$ par la combinaison des cas $\texttt{Source}_2$ et $\texttt{Source}_3$, et $\texttt{CibleComplétée}_3$ par la combinaison des cas $\texttt{Source}_2$, $\texttt{Source}_3$ et $\texttt{Source}_3$ (compté deux fois).

que la valeur de `vMousse` soit conservée plutôt que celle de `œuf` et `soja`, il faut que $w_4 > k \cdot (w_5 + w_6)$).

3.4.3 Lien avec le CCBI

Inférence crédible à partir de cas

Hypothèses. L'inférence crédible à partir de cas (*Credible case-based inference* ou CCBI) [Hüllermeier, 2007] est une approche du RÀPC qui suppose que la relation problème-solution $\rightsquigarrow$ est une fonction partielle : si $x \rightsquigarrow y$ et $x \rightsquigarrow y'$ alors $y = y'$. De plus, les cas sources sont des singletons : $\texttt{Source}_i = \{(x_i, y_i)\}$ et le cas cible est composé d'un singleton problème et d'une solution non spécifiée : $\texttt{Cible} = \{x_\texttt{c}\} \times \mathcal{U}_\texttt{sol}$.

Le CCBI modélise le principe suivant : « Des problèmes similaires ont des solutions similaires » par une fonction $h : [0; +\infty] \to [0; +\infty]$, le *profil de similarité*, qui contraint la distance entre solutions en fonction de la distance entre problèmes [15], pour la *plupart* des $x_1, x_2 \in \mathcal{U}_\texttt{pb}$, si $x_1 \rightsquigarrow y_1$ et $x_2 \rightsquigarrow y_2$ alors :

$$d_\texttt{sol}(y_1, y_2) \leqslant h(d_\texttt{pb}(x_1, x_2)) \tag{3.14}$$

où $d_\texttt{pb}$ est une « distance » symétrique sur $\mathcal{U}_\texttt{pb}$ et $d_\texttt{sol}$ est une « distance » symétrique sur $\mathcal{U}_\texttt{sol}$. Une méthode d'apprentissage de la fonction h à partir de la base de cas est aussi décrite dans [Hüllermeier, 2007] et il est aussi prouvé, sous certaines hypothèses techniques, que la probabilité qu'une contrainte exprimée par (3.14) soit violée converge vers 0 au fur et à mesure que la base de cas grandit.

Remarque 3.4.1. *Puisque l'on suppose qu'un problème n'a qu'une seule solution ($\rightsquigarrow$ est une fonction partielle), on suppose que $h(0) = 0$.*

Définition du CCBI. Étant donné un ensemble $\texttt{SCS}$ de cas sources $\texttt{Source}_i = \{(x_i, y_i)\}_{1 \leqslant i \leqslant n}$ et un problème cible $x_\texttt{c}$, si on suppose que l'inégalité (3.14) est vérifiée pour tout couple de cas, alors la solution $y_\texttt{c}$ de $x_\texttt{c}$ vérifie $d_\texttt{sol}(y_i, y_\texttt{c}) \leqslant h(d_\texttt{pb}(x_i, x_\texttt{c}))$. Autrement dit (Fig. 3.4) :

$$y_\texttt{c} \in \mathcal{C}_{\text{CCBI}} = \bigcap_{i=1}^{n} \{y \in \mathcal{U}_\texttt{sol} \mid d_\texttt{sol}(y_i, y) \leqslant h(d_\texttt{pb}(x_i, x_\texttt{c}))\} \tag{3.15}$$

Donc, $\texttt{CibleComplétée} = \{x_\texttt{c}\} \times \mathcal{C}_{\text{CCBI}}$ résout $\texttt{Cible}$.

Cette inférence est crédible mais n'est pas certaine puisque (3.14) n'est satisfaite que pour la plupart des $x, y \in \mathcal{U}_\texttt{pb}$.

La $\triangle^{d,\Sigma}$-combinaison de cas étend le CCBI

Proposition 3.4.1. *On fait ici l'hypothèse du* CCBI *à propos de la base de cas.*

15. En fait, le CCBI est définie dans [Hüllermeier, 2007] grâce à des mesures de similarité $\mathcal{S}_\texttt{pb}$ et $\mathcal{S}_\texttt{sol}$ sur $\mathcal{U}_\texttt{pb}$ et $\mathcal{U}_\texttt{sol}$, mais la définition présentée ici est équivalente. En effet, une mesure de similarité $\mathcal{S}$ sur $\mathcal{U}$ qui vérifie $\mathcal{S}(x, y) = 1$ si $x = y$ peut être définie grâce à une « distance » d sur $\mathcal{U}$ par $\mathcal{S}(x, y) = \frac{1}{1+d(x,y)}$ et inversement.

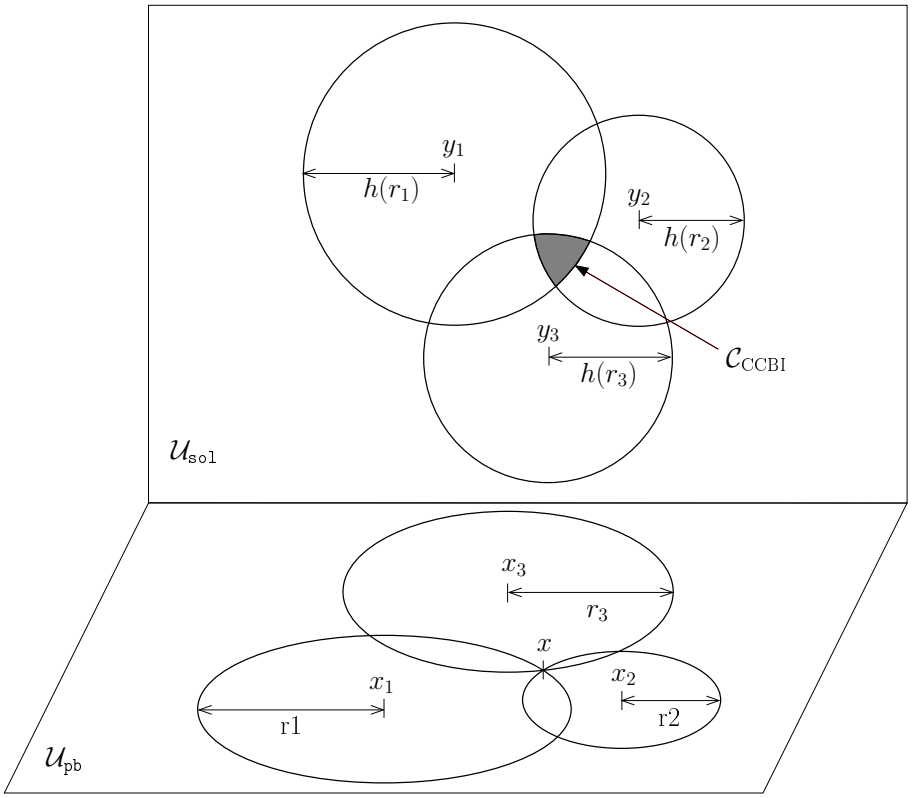


FIGURE 3.4 – Inférence crédible à partir de cas.

Soit d la « distance » sur $\mathcal{U} = \mathcal{U}_{pb} \times \mathcal{U}_{sol}$ définie pour $(x_1, y_1), (x_2, y_2) \in \mathcal{U}$ par

$$d((x_1, y_1), (x_2, y_2)) = \max\{h(d_{pb}(x_1, x_2)), d_{sol}(y_1, y_2)\}$$

Soit $\mathcal{C}_{CCBI}$ le résultat du CCBI*, et $\mathcal{C}_{\triangle^{d,\Sigma}}$ le résultat de la $\triangle^{d,\Sigma}$-combinaison des mêmes cas sources sans connaissances du domaine ($\mathtt{CD} = \mathcal{U}$) :*

$$\mathcal{C}_{CCBI} = \bigcap_{i=1}^{n} \{y \in \mathcal{U}_{sol} \mid d_{sol}(y_i, y) \leqslant h(d_{pb}(x_i, x_c))\}$$
$$\mathcal{C}_{\triangle^{d,\Sigma}} = \Phi_{sol}\left(\mathcal{CC}^{\mathcal{U}}_{\triangle^{d,\Sigma}}(\mathcal{SCS}, \mathcal{C}ible)\right)$$

où Φ_{sol} est la projection sur l'espace des solutions : étant donné un ensemble $\mathcal{C}as \subseteq \mathcal{U}$, $\Phi_{sol}(\mathcal{C}as) = \{y \in \mathcal{U}_{sol} \mid$ il existe $x \in \mathcal{U}_{pb}$ tel que $(x, y) \in \mathcal{C}as\}$. Si le CCBI *donne un résultat cohérent : $\mathcal{C}_{CCBI} \neq \emptyset$, alors il coïncide avec le résultat de la $\triangle^{d,\Sigma}$-combinaison de cas :*

$$\mathcal{C}_{CCBI} = \mathcal{C}_{\triangle^{d,\Sigma}}$$

Démonstration. Soit $y_{\mathtt{c}} \in \mathcal{C}_{\text{CCBI}}$, $d_{\mathtt{sol}}(y_i, y_{\mathtt{c}}) \leqslant h(d_{\mathtt{pb}}(x_i, x_{\mathtt{c}}))$ pour tout $1 \leqslant i \leqslant n$, donc

$$d((x_i, y_i), (x_{\mathtt{c}}, y_{\mathtt{c}})) = h\left(d_{\mathtt{pb}}(x_i, x_{\mathtt{c}})\right) \leqslant \min_{y \in \mathcal{U}_{\mathtt{sol}}} d((x_i, y_i), (x_{\mathtt{c}}, y))$$

et

$$\begin{aligned} d^{\Sigma}(\mathtt{SCS}, (x_{\mathtt{c}}, y_{\mathtt{c}})) &= \sum_{i=1}^{n} d((x_i, y_i), (x_{\mathtt{c}}, y_{\mathtt{c}})) \\ &\leqslant \sum_{i=1}^{n} \left(\min_{y \in \mathcal{U}_{\mathtt{sol}}} d((x_i, y_i), (x_{\mathtt{c}}, y)) \right) \\ &\leqslant \min_{y \in \mathcal{U}_{\mathtt{sol}}} \left(\sum_{i=1}^{n} d((x_i, y_i), (x_{\mathtt{c}}, y)) \right) = \min_{(x_{\mathtt{c}}, y) \in \mathcal{U}} d^{\Sigma}(\mathtt{SCS}, (x_{\mathtt{c}}, y)) \end{aligned}$$

Donc, $(x_{\mathtt{c}}, y_{\mathtt{c}}) \in \mathtt{CC}^{\mathcal{U}}_{\triangle^{d,\Sigma}}(\mathtt{SCS}, \mathtt{Cible})$ et $y_{\mathtt{c}} \in \mathcal{C}_{\triangle^{d,\Sigma}}$, ce qui montre que $\mathcal{C}_{\text{CCBI}} \subseteq \mathcal{C}_{\triangle^{d,\Sigma}}$.

Inversement, si $\mathcal{C}_{\text{CCBI}} \neq \emptyset$, (i.e., il existe un tel $y_{\mathtt{c}}$), alors $\min_{y \in \mathcal{U}_{\mathtt{sol}}} d^{\Sigma}(\mathtt{SCS}, (x_{\mathtt{c}}, y)) \leqslant d^{\Sigma}(\mathtt{SCS}, (x_{\mathtt{c}}, y_{\mathtt{c}}))$ et les inégalités précédentes sont des égalités. Donc, si $y \in \mathcal{U}_{\mathtt{sol}}$ minimise $d^{\Sigma}(\mathtt{SCS}, (x_{\mathtt{c}}, y))$, alors $\sum_{i=1}^{n} d((x_i, y_i), (x_{\mathtt{c}}, y)) = \sum_{i=1}^{n} d((x_i, y_i), (x_{\mathtt{c}}, y_{\mathtt{c}}))$. Comme pour tout $1 \leqslant i \leqslant n$ on a $d((x_i, y_i), (x_{\mathtt{c}}, y)) \geqslant d((x_i, y_i), (x_{\mathtt{c}}, y_{\mathtt{c}}))$, cela signifie que

$$d((x_i, y_i), (x_{\mathtt{c}}, y)) = d((x_i, y_i), (x_{\mathtt{c}}, y_{\mathtt{c}})) = h\left(d_{\mathtt{pb}}(x_i, x_{\mathtt{c}})\right)$$

c'est-à-dire $y \in \mathcal{C}_{\text{CCBI}}$, et on a bien $\mathcal{C}_{\triangle^{d,\Sigma}} \subseteq \mathcal{C}_{\text{CCBI}}$. □

Remarque 3.4.2. *Pour que d vérifie la séparation et soit donc effectivement une « distance », il faut que $h(v) = 0$ ssi $v = 0$. Le « si » est en accord avec les hypothèses (voir la remarque 3.4.1), pour le « seulement si » il suffit d'ajouter une valeur $\varepsilon \cdot x$ avec $\varepsilon > 0$ à $h(x)$: $h_{\varepsilon}(x) = h(x) + \varepsilon \cdot x$, puisque $h(x) \geqslant 0$ pour tout x, $h_{\varepsilon}(x) = 0$ implique que $\varepsilon \cdot x = 0$ et donc $x = 0$. Si h est un* profil de similarité *(i.e. il vérifie l'inégalité* (3.14)*, page 108) h_{ε} en est un aussi, mais il engendre un résultat moins précis pour le* CCBI*, on choisira donc un « petit » ε.*

On peut se demander si la proposition 3.4.1 est vraie aussi pour d'autres opérateurs de fusion contrainte. La propriété suivante montre qu'elle est vraie pour une famille d'opérateurs de ce type paramétrée par une fonction poids sur $2^{\mathcal{U}}$:

Proposition 3.4.2. *Soit M un multi-ensemble de sous-ensembles de $\mathcal{U}$, $y \in \mathcal{U}$, d une « distance » sur $\mathcal{U}$ et une fonction poids W qui associe un nombre réel $W(A) > 0$ à tout ensemble $A \in 2^{\mathcal{U}}$. On note $d^{\Sigma,W}(M, y) = \sum_{A \in M} W(A).d(A, y)$. En particulier, $d^{\Sigma} = d^{\Sigma,\mathbf{1}}$ avec $\mathbf{1}(A) = 1$ pour tout $A \in 2^{\mathcal{U}}$. Soit $\triangle^{d,\Sigma,W}$ l'opérateur de fusion contrainte défini par $\triangle^{d,\Sigma,W}_{IC}(M) = \{y \in IC \mid d^{\Sigma,W}(M, y) = d^{\Sigma,W}(M, IC)\}$. En particulier, $\triangle^{d,\Sigma} = \triangle^{d,\Sigma,\mathbf{1}}$.*
Avec ces définitions, la propriété 3.4.1 reste vraie si on remplace $\triangle^{d,\Sigma}$ par tout $\triangle^{d,\Sigma,W}$.

3.4.4 Lien avec d'autres approches de combinaison de cas en RàPC

Nous présentons ici quelques approches de combinaison utilisées dans des systèmes de RàPC que nous comparons ensuite suivant différents critères à la combinaison par fusion contrainte.

IDIOM [Smith *et al.*, 1995]. IDIOM est un outil d'architecte pour l'agencement des pièces dans un appartement. Les cas sources sont des exemples d'agencement de plusieurs pièces. Plusieurs cas peuvent être assemblés en respectant certaines contraintes (ouverture des portes, fenêtres, etc.) pour former des agencements plus complexes.

COMPOSER [Purvis et Pu, 1996]. COMPOSER est un système de RàPC heuristique pour résoudre des problèmes de satisfaction de contraintes, par exemple pour des problèmes d'assemblage de pièces. Les cas sources donnent des solutions partielles qui doivent ensuite être combinées, à l'aide d'un outil de résolution de contraintes, pour obtenir une solution globale cohérente.

Déjà Vu [Smyth *et al.*, 2001]. Déjà Vu est un système de conception de programmes informatiques par RàPC. Les cas sources sont organisés dans une hiérarchie suivant des relations de composition : un cas abstrait donne une description globale d'un programme dont les différentes parties sont précisées par d'autres cas. Pour réaliser un programme qui exécute certaines tâches, un cas source abstrait est d'abord remémoré et adapté, la solution abstraite obtenue est ensuite précisée en faisant appel à des cas sources plus précis pour résoudre les sous-tâches. La cohérence du programme global est assurée par le cas abstrait qui décrit les interactions entre les différentes composantes du programme.

RàPC décentralisé [d'Aquin *et al.*, 2005]. Le raisonnement à partir de cas décentralisé ou DzCBR (*Decentralized Case-Based Reasoning*) est une approche du RàPC faisant intervenir plusieurs points de vue. Par exemple, les traitements pour le cancer sont décidés en concertation par des médecins de différentes spécialités (chirurgien, chimiothérapeute, radiothérapeute, etc.). Chaque médecin s'appuie sur des traitements déjà appliqués à d'autres patients (les cas médicaux) et des connaissances propres à sa spécialité pour proposer un traitement. Un cas cible est considéré dans le contexte des différents points de vue avec les connaissances du domaine et les connaissances d'adaptation propres à chaque point de vue. Une solution locale est élaborée à partir d'un cas source local. Des règles, appelées passerelles, expriment les interactions entre les solutions locales des différents points de vue pour assurer la cohérence de la solution globale obtenue.

Comparaison avec la combinaison par fusion contrainte

Cette comparaison s'appuie sur les critères retenus dans la revue des approches de combinaison par [Gebhardt *et al.*, 1997] : la décomposition et recomposition des cas, la réutilisation

simultanée ou itérative des cas sources, la manière d'assurer la cohérence du résultat.

Suivant les approches de combinaison, les cas sources sont décomposés puis recomposés de différentes manières, pour la combinaison. Dans l'approche DzCBR, une décomposition fixe du domaine est établie suivant les différents points de vue. Une solution partielle est calculée dans chaque point de vue par un processus de RÀPC distinct, et la solution est l'union des ces solutions partielles. Dans DÉJÀ VU, les cas se décomposent aussi en plusieurs parties, mais ici la décomposition est donnée par la hiérarchie de la base de cas, elle dépend donc des cas remémorés. Dans IDIOM comme dans COMPOSER, il n'y a pas de décomposition des cas sources, ils sont remémorés de manière à « couvrir » le problème cible. La combinaison par fusion contrainte, telle que présentée ici, ne présente pas de décomposition. Une perspective intéressante serait de voir si certaines formes de décomposition peuvent être obtenues par la fusion en structurant l'espace des cas.

Alors que COMPOSER et DzCBR réutilisent les cas sources simultanément, DÉJÀ VU et IDIOM le font itérativement. Dans DÉJÀ VU, la résolution commence par l'adaptation d'un cas source abstrait et est itérée sur les sous-problèmes obtenus. Dans IDIOM une solution est construite en incorporant itérativement des cas sources. La fusion contrainte traite les différentes sources simultanément, la combinaison par fusion fait donc partie du premier groupe. Il faudrait étudier la possibilité de représenter les combinaisons itératives par une itération de révisions [16].

Le dernier critère retenu pour comparer les approches est la manière d'assurer la cohérence de la solution. Dans DzCBR, la solution globale est cohérente si les solutions locales des différents points de vue sont cohérentes avec les connaissances du domaine des points de vue et avec les conséquences, par l'intermédiaire des passerelles, des solutions obtenues dans les autres points de vue. Lorsque les solutions locales sont cohérentes, la solution globale l'est aussi, sinon l'adaptation doit être poursuivie dans les points de vue qui posent problème. IDIOM et COMPOSER utilisent un algorithme de résolution de conflit. Dans COMPOSER, les conflits sont résolus une fois que tous les cas sources ont été collectés. Dans IDIOM, les corrections sont faites au fur et à mesure qu'un nouveau cas source est ajouté. Dans la combinaison par fusion, les incohérences éventuelles entre cas sont traitées par l'opérateur de fusion. Cela motive l'étude des relations entre résolution de conflits et fusion des connaissances.

3.4.5 Perspectives

Comme nous l'avons noté en section 3.4.2 à propos de l'exemple de combinaison de recettes de cuisine pour obtenir une recette de mousse au chocolat sans œufs, l'opérateur de révision $\triangle^{d,\Sigma}$ défini en section 2.7.2 établit un compromis de majorité entre les cas sources, or il serait plus intéressant dans le cadre du RÀPC de tirer le meilleur parti des différents cas sources. Dans l'exemple, pour obtenir une recette de mousse au chocolat sans œufs, on devrait s'inspirer des recettes $\texttt{Source}_1$ et $\texttt{Source}_2$ pour obtenir une mousse onctueuse (donc les valeurs de `vMousse`,

16. C.-à-d. de calculer d'abord $\texttt{CibleComplétée}_1 = (\texttt{CD} \cap \texttt{Source}_1) \dot{+} (\texttt{CD} \cap \texttt{Cible})$ puis itérativement $\texttt{CibleComplétée}_{i+1} = (\texttt{CD} \cap \texttt{Source}_i) \dot{+} (\texttt{CD} \cap \texttt{CibleComplétée}_i)$.

`œuf` et `soja`), et des recettes $\texttt{Source}_1$ et $\texttt{Source}_3$ pour obtenir un enrobage au chocolat sans œufs (ce qui correspond aux valeurs de `chocolat`, `œuf` et `crème`).

Une première idée pour obtenir un opérateur de fusion vérifiant ce comportement est de minimiser le changement, par rapport à l'un des cas sources, séparément pour chaque composante, et d'agréger le résultat ensuite. Étant donné un espace $\mathcal{U} = \mathcal{U}_1 \times \ldots \times \mathcal{U}_n$, la combinaison des cas $\texttt{Source}_1, \ldots, \texttt{Source}_k$ pour résoudre `Cible` est obtenue en minimisant, pour chaque composante $\mathcal{U}_j$, la fonction :

$$u_j \in \mathcal{U}_j \mapsto \min_{i=1,\ldots,k} \left(\min_{x \in \texttt{Source}_i} d_j(x_j, u_j) \right) = \min_{\substack{i=1,\ldots,k \\ x \in \texttt{Source}_i}} d_j(x_j, u_j)$$

où chaque d_j est une « distance » sur $\mathcal{U}_j$. Ces fonctions à minimiser sont agrégées, par exemple en faisant leur somme, la combinaison des cas $\texttt{Source}_1, \ldots, \texttt{Source}_k$ pour résoudre `Cible` est donc déterminée par les minima de la fonction $\Sigma^{\min}$ telle que pour tout $u = (u_1, \ldots, u_n)$:

$$\Sigma^{\min}(\{\texttt{Source}_1, \ldots, \texttt{Source}_k\}, u) = \sum_{j=1}^{n} \left(\min_{\substack{i=1,\ldots,k \\ x \in \texttt{Source}_i}} d_j(x_j, u_j) \right)$$

Ou, pour reprendre les notations de la fusion de connaissances, pour un multi-ensemble fini $\mathcal{M}$ de sous-ensembles de $\mathcal{U}$ et un élément $u \in \mathcal{U}$:

$$\Sigma^{\min}(\mathcal{M}, u) = \sum_{j=1}^{n} \left(\min_{\substack{A \in \mathcal{M} \\ x \in A}} d_j(x_j, u_j) \right)$$

On note $\triangle^{\Sigma^{\min}}$ l'opérateur de fusion obtenu par minimisation de $\Sigma^{\min}$, pour toute BC IC et tout multi-ensemble fini $\mathcal{B}$ de BC :

$$\mathrm{Mod}(\triangle_{IC}^{\Sigma^{\min}}(\mathcal{B})) = \underset{\Sigma^{\min}(\mathrm{Mods}(\mathcal{B}),\cdot)}{\mathrm{Min}} \mathrm{Mod}(IC)$$

La figure 3.5 représente schématiquement le résultat de la combinaison de cas définie à l'aide de $\triangle^{\Sigma^{\min}}$.

Exemple 3.4.1. *Avec l'exemple de la section 3.4.2, la combinaison des cas* $\texttt{Source}_2$ *et* $\texttt{Source}_3$ *pour résoudre* `Cible` *par la combinaison définie à l'aide de* $\triangle^{\Sigma^{\min}}$ *donne le résultat* $\texttt{CibleComplétée}_4$ *présenté dans le tableau 3.3. Pour chaque variable, les valeurs retenues sont celles de l'un des cas sources. Pour* `vMousse` *et* `chocolat` *les contraintes de* `CD` *qui les lient respectivement à* `estMousseux` *et* `contientChocolat` *élimine les valeurs* 0*. La contrainte de* `CD` *reliant* `vMousse`*,* `œuf` *et* `soja` *implique que* `soja` *ne peut pas prendre la valeur* 0*, le changement minimal est obtenu pour la valeur* 170*. Les valeurs de* $\texttt{Source}_2$ *sont bien reprises pour les attributs concernant la mousse, et les valeurs de* $\texttt{Source}_3$ *pour l'enrobage au chocolat.*

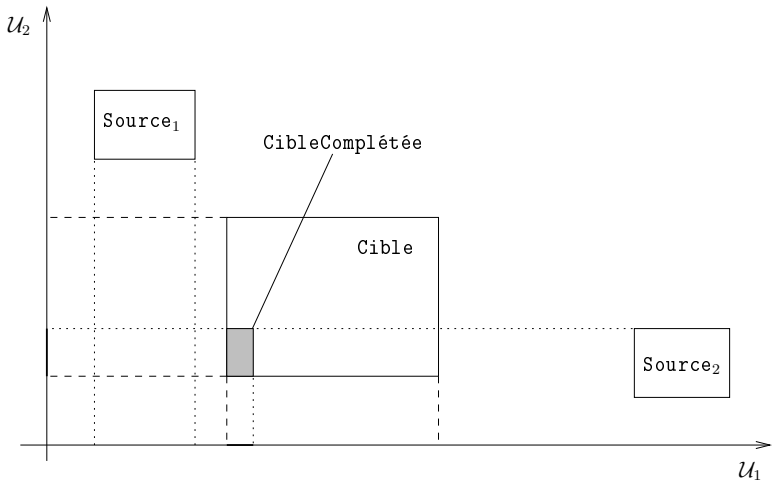


FIGURE 3.5 – Représentation schématique du résultat de la combinaison des cas Source$_1$ et Source$_2$ pour résoudre Cible dans l'espace $\mathcal{U} = \mathcal{U}_1 \times \mathcal{U}_2$.

La répétition de Source$_3$ *ne change pas le résultat :*

$$CC^{CD}_{\triangle^{\Sigma\min}}(\{Source_2, Source_3, Source_3\}, Cible) = CibleComplétée_4$$
$$= CC^{CD}_{\triangle^{\Sigma\min}}(\{Source_2, Source_3\}, Cible)$$

La combinaison des cas Source$_1$, Source$_2$ *et* Source$_3$ *pour résoudre* Cible *donne le résultat* CibleComplétée$_5$ *présenté dans le tableau 3.3. Ici aussi, les valeurs de* Source$_1$ *et* Source$_2$ *sont reprises pour les attributs concernant la mousse, et les valeurs de* Source$_1$ *et* Source$_3$ *pour l'enrobage au chocolat.*

	CibleComplétée$_4$	CibleComplétée$_5$
a$_1$ = estMousseux	Vrai	Vrai
a$_2$ = contientŒuf	Faux	Faux
a$_3$ = contientChocolat	Vrai	Vrai
a$_4$ = vMousse (ml)	225	200 ou 225
a$_5$ = œuf	0	0
a$_6$ = soja (ml)	170	170
a$_7$ = chocolat (g)	25	25 ou 35
a$_8$ = crème (g)	0 ou 125	0 ou 10 ou 125
a$_9$ = sucre (g)	_	_

TABLE 3.3 – Résultats de la combinaison de cas suivant l'opérateur $\triangle^{\Sigma\min}$. CibleComplétée$_4$ est obtenu par combinaison des cas Source$_2$ et Source$_3$, CibleComplétée$_5$ par combinaison des cas Source$_2$ et Source$_3$.

Il serait intéressant de le comparer la combinaison de cas obtenue aux approches présentées en

section 3.4.4. Il faudrait cependant voir comment généraliser la définition de $\Sigma^{\min}$ à des domaines $\mathcal{U}$ qui ne se décomposent pas de manière fixe en $\mathcal{U} = \mathcal{U}_1 \times \ldots \times \mathcal{U}_n$, par exemple lorsque les cas se décomposent de manière hiérarchique comme pour DÉJÀ VU.

L'opérateur de fusion $\triangle^{\Sigma^{\min}}$ ne satisfait pas tous les postulats, en particulier il ne satisfait pas ($\triangle$-2). Pour obtenir un opérateur de fusion satisfaisant les postulats, on peut par exemple considérer de prendre les minima lexicographiques selon les distances croissantes vis-à-vis des différents cas $\texttt{Source}_i$, en s'inspirant de la fonction d'agrégation GMax utilisée par [Konieczny, 1999].

Chapitre 4

Application de l'adaptation par révision à TAAABLE

Dans ce chapitre nous présentons une contribution de cette thèse au système TAAABLE qui participe au *Computer Cooking Contest* et dont le but est de proposer des recettes de cuisine en réponse à des requêtes. Nous présentons TAAABLE et les règles du concours en section 4.1. Nous expliquons en section 4.2 comment nous avons appliqué les outils d'adaptation par révision pour l'adaptation des quantités d'ingrédients. Et pour finir nous donnons en section 4.3 quelques idées pour d'autres applications possibles des travaux de cette thèse pour TAAABLE.

4.1 Présentation de TAAABLE

4.1.1 Le *Computer Cooking Contest*

Le *Computer Cooking Contest* (CCC, http://liris.cnrs.fr/ccc/ccc2011/) est un concours qui a lieu tous les ans depuis 2008, la première année lors de la conférence ECCBR[17] (*European Conference on Case-Based Reasoning*) et les suivantes lors des conférences ICCBR (*International Conference on Case-Based Reasoning*).

Les participants doivent présenter un système qui exploite une base de recettes de cuisine fournie par les organisateurs du CCC et doit répondre à des demandes de recette du type :

> « Je veux une crêpe avec des bananes mais je n'aime pas le chocolat. »

Ces requêtes portent sur les ingrédients désirés ou au contraire à éviter, sur le type de plats voulu (entrée, plat, dessert, tarte, soupe, ...), et sur une origine (asiatique, méditerranéenne, ...). Des régimes alimentaires peuvent aussi être spécifiés (allergie aux oléagineux, végétarien, ...). Les règles du concours n'imposent pas d'interface capable de recevoir des requêtes sous cette forme, une personne peut l'interpréter dans le langage du système. Les recettes proposées par le système en réponse à cette requête doivent être tirées de la base de recettes fournie, mais elles peuvent avoir été adaptées.

17. Cette conférence a depuis fusionné avec ICCBR.

En plus de cette épreuve principale, des *challenges* supplémentaires sont proposés, par exemple l'adaptation d'une recette spécifique pour satisfaire des exigences de l'utilisateur.

Une première évaluation, scientifique, des participants a lieu lors de l'atelier CCC où chaque équipe doit soumettre un article présentant le fonctionnement de leur système, le comité scientifique juge de l'originalité et de la qualité des approches. Une deuxième évaluation a lieu lors du déroulement du concours où un jury note les résultats fournis par les systèmes en réponse à une liste de requêtes. Le jury est composé de membres de la communauté RÀPC et, certaines années, d'un chef cuisinier.

En 2008 (TAAABLE 1) et 2009, TAAABLE a reçu le deuxième prix. En 2010, TAAABLE 3 a reçu le premier prix et le prix de l'adaptation.

4.1.2 TAAABLE 1

Le système TAAABLE (http://taaable.fr) fonctionne suivant le principe du RÀPC où les recettes sont vues comme des cas sources et la requête, comme un cas cible. Pour réaliser le premier système TAAABLE [Badra *et al.*, 2008], plusieurs travaux ont été effectués : acquisition de connaissances du domaine (culinaires) pour assister le raisonnement sur les recettes de cuisine, indexation des recettes fournies par le CCC, et l'implantation d'une première version du moteur de RÀPC.

Acquisition de connaissances culinaires. Pour guider la remémoration et l'adaptation des recettes de cuisine, des connaissances du domaine ont été acquises et rassemblées dans une ontologie culinaire. Pour la première version de TAAABLE, il s'agit essentiellement de hiérarchies des ingrédients, des types de plats et des origines. Ces connaissances ont été extraites de sites web puis validées manuellement.

Cette ontologie O est représentée dans le formalisme logique par des formules en logique propositionnelle, par exemple `pomme` $\Rightarrow$ `fruit`, `banane` $\Rightarrow$ `fruit`, `platCrêpe` $\Rightarrow$ `dessert`.

Indexation des recettes. Pour cette première version de TAAABLE, les recettes ne sont indexées que par les critères intervenant dans les requêtes. C'est-à-dire que pour le raisonnement, les recettes sont assimilées à la liste de leurs ingrédients (sans tenir compte des quantités), au type de plat préparé et à son origine éventuelle. Des outils d'indexation automatique ont été développés pour identifier pour chaque recette la liste de ses ingrédients, l'indexation des types de plats et de leur origine a été faite semi-automatiquement.

Dans le moteur de RÀPC, les recettes de cuisine sont représentées en logique propositionnelle, avec une variable par type d'ingrédient (`banane`, `fruit`, etc.), une variable par type de plat (`platCrêpe`, `dessert`, etc.), une variable par origine (`breton`, `français`, `asiatique`, etc.). Ainsi, `banane` représente la classe des recettes avec de la banane, `breton`, la classe des recettes d'origine bretonne, etc. Une recette est donc représentée par la conjonction des variables correspondant aux ingrédients qu'elle contient, des variables des types de plats et d'origine avec lesquels elle

a été annotée, et la négation des autres variables. Par exemple la recette *Baked apple pancake* de crêpes avec des pommes est représentée par la formule suivante (quelques ingrédients ont été enlevés pour simplifier) :

$$\texttt{BakedApplePancake} = \texttt{platCrêpe} \wedge \texttt{farine} \wedge \texttt{lait} \wedge \texttt{saccharose} \wedge \texttt{œuf} \wedge \texttt{pomme} \wedge \textit{rienDAutre}$$

où *rienDAutre* est la conjonction de négations de variables correspondant aux ingrédients non contenus dans la recette et les types et origines par lesquels elle n'est pas annotée [18]. `saccharose` correspond au sucre en poudre, on utilise ce terme pour le différencier du sucre qui peut être aussi contenu dans les ingrédients.

Implantation d'un moteur de RÀPC. Le cas cible, qui correspond à la requête, est représenté par une formule dans le même formalisme que les recettes. La requête « Je veux une crêpe avec des bananes mais je n'aime pas le chocolat. » est représentée par la formule :

$$\texttt{Cible} = \texttt{platCrêpe} \wedge \texttt{banane} \wedge \neg\texttt{chocolat}$$

Le moteur de RÀPC de TAAABLE doit alors fournir une formule `CibleComplétée` qui représente une recette de cuisine et qui répond à la requête, c'est-à-dire $\texttt{CibleComplétée} \models_O \texttt{Cible}$. Pour cela une recette de la base est remémorée puis adaptée en substituant des ingrédients. Par exemple la recette *Baked apple pancake* est adaptée en remplaçant les pommes par des bananes :

$$\texttt{CibleComplétée} = \texttt{platCrêpe} \wedge \texttt{farine} \wedge \texttt{lait} \wedge \texttt{saccharose} \wedge \texttt{œuf} \wedge \texttt{banane} \wedge \textit{rienDAutre}$$

Ce résultat est obtenu en généralisant `Cible` tant qu'il n'existe pas de recette `Recette` telle que $\texttt{Recette} \models_O \texttt{Cible}$. Ces généralisations sont obtenues en généralisant les termes de `Cible`. Un coût est donné aux généralisations, la généralisation la moins coûteuse ici est celle de `banane` par `fruit`. Le résultat est $\texttt{Cible}' = \texttt{platCrêpe} \wedge \texttt{fruit} \wedge \neg\texttt{chocolat}$, et $\texttt{BakedApplePancake} \models_O \texttt{Cible}'$ est remémorée. Pour adapter `BakedApplePancake`, on détermine quel terme spécialise `fruit`, c'est `pomme` ($\texttt{pomme} \models_O \texttt{fruit}$), `pomme` est alors substituée par `banane` pour obtenir `CibleComplétée`.

4.1.3 TAAABLE 2

Deux nouveautés sont apportées dans TAAABLE 2 [Badra *et al.*, 2009a] : le transfert des connaissances exploitées par TAAABLE dans un wiki sémantique, WIKITAAABLE (http://wikitaaable.loria.fr) et l'apprentissage de règles d'adaptation.

WIKITAAABLE. L'ontologie du domaine et les recettes de cuisine ont été transférées dans le wiki sémantique utilisant le moteur *Semantic Media Wiki* [Völkel *et al.*, 2007, SMW, 2011].

18. *rienDAutre* est la conjonction des littéraux $\neg v$ où les variables v sont telles que $\texttt{BakedApplePancake} \not\models v$, donc par exemple le terme $\neg\texttt{fruit}$ ne fera pas partie de *rienDAutre* vu que $\models_O \texttt{pomme} \Rightarrow \texttt{fruit}$.

Chaque recette est représentée par une page du wiki [Cordier *et al.*, 2009]. La figure 4.1 présente la page de la recette *Baked apple pancake*.

Baked apple pancake

Ingredients

- 2 Granny smith apples; peeled, cored; thinly sliced category:granny_smith_apple (2, ?, peeled,cored,thinly sliced, ?, ?)
- 1/2 c Granulated sugar category:granulated_sugar (1/2, c, ?, ?, ?)
- 1/2 c Packed brown sugar category:brown_sugar (1/2, c, packed, ?, ?)
- 2 ts Cinnamon category:cinnamon (2, tsp, ?, ?, ?)
- 1 c Milk category:milk (1, c, ?, ?, ?)
- 4 lg Eggs category:egg (4, ?, large, ?, ?)
- 1 c All-purpose flour category:all-purpose_flour (1, c, ?, ?, ?)
- 1 pn Salt category:salt (1, pinch, ?, ?, ?)
- 1/2 Stick unsalted butter category:unsalted_butter (1/2, ?, stick, ?, ?)

Preparation

- 1. Heat oven to 425 degrees. Lightly oil a 10-inch cast-iron skillet. Toss apples with both sugars and cinnamon in a large bowl. 2. Mix milk and eggs in a blender or food processor; add flour and salt and mix to combine. 3. Melt butter in prepared skillet over medium heat. Add apple mixture; cook, stirring often, until sugar melts, about 5 minutes. Remove from heat and pour batter over. 4. Bake until pancake is puffed and golden, 20 to 30 minutes. Serve at once. Yield: 2 to 4 servings.

FIGURE 4.1 – Page WIKITAAABLE de la recette *Baked apple pancake*.

Les pages de recettes contiennent des liens vers les pages correspondant aux ingrédients, aux types de plat et aux origines. Ces liens sont étiquetés, ce qui permet de définir formellement des relations entre la recette et les ingrédients qu'elle contient et l'annoter par les types de plat et les origines. Les pages correspondant aux ingrédients, aux types et aux origines sont des pages « catégorie » qui permettent de définir une hiérarchie dans *Semantic Media Wiki*, toujours sous la forme de liens étiquetés par des relations « sous-catégorie de » et « super-catégorie de ». La figure 4.2 présente la page des pommes.

Semantic Media Wiki permet de consulter les relations par des requêtes SPARQL. Il est ainsi possible de parcourir la hiérarchie d'ingrédients et de connaître pour chaque recette les ingrédients qu'elle contient. Le moteur de RÀPC a ainsi accès aux mêmes connaissances que précédemment [19].

L'intérêt du wiki est de permettre à une communauté d'utilisateurs d'enrichir la base de recettes et de les annoter. Les outils d'annotation automatique utilisés dans TAAABLE 1 pour indexer les recettes sont exécutés par des robots d'indexation (*bots*) qui annotent automatiquement les pages du wiki.

19. Pour l'instant le moteur de RÀPC n'accède pas aussi directement aux connaissances du wiki. Il exploite un export RDF représentant l'ensemble des relations du wiki.

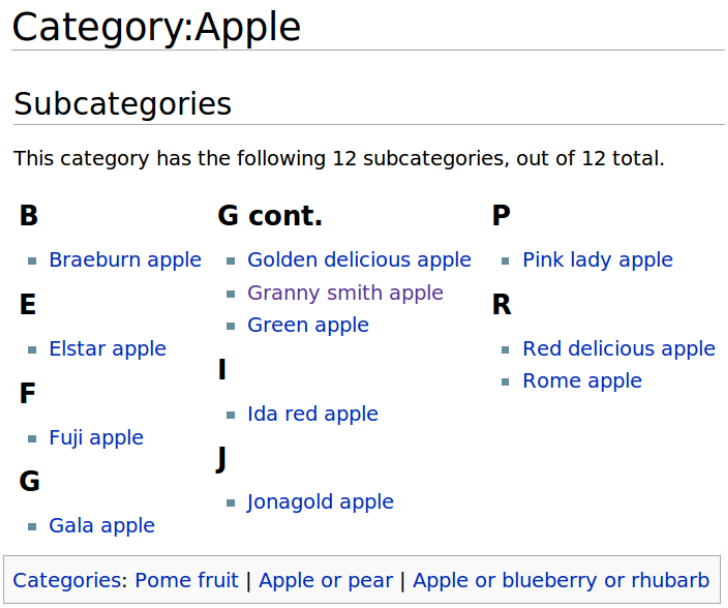


FIGURE 4.2 – Page WIKITAAABLE de la catégorie d'ingrédients *Apple*. La partie *Subcategories* liste les catégories d'ingrédient directement inclus dans la catégorie *Apple*. L'encadré en bas liste les catégories englobant directement *Apple*.

Acquisition de règles d'adaptation. En analysant les variations entre les ingrédients utilisés par les recettes, on peut identifier des motifs fréquents qui suggèrent des substitutions d'ingrédients lors de l'adaptation d'autres recettes [Badra, 2009]. Plus précisément, les motifs recherchés sont des triplets de formules ($\texttt{contexte}, \texttt{initial}, \texttt{substitut}$) où `contexte` représente les points communs (ingrédients, types, origine) entre deux recettes, `initial` représente les ingrédients présents dans la première recette qui sont substitués par les ingrédients `substitut` dans la deuxième. La variation entre une recette $\texttt{Recette}_1$ et une recette $\texttt{Recette}_2$ vérifie un tel motif si :

$$\texttt{Recette}_1 \models_O \texttt{contexte} \qquad \texttt{Recette}_1 \models_O \texttt{initial} \qquad \texttt{Recette}_1 \wedge \texttt{substitut} \models_O \bot$$
$$\texttt{Recette}_2 \models_O \texttt{contexte} \qquad \texttt{Recette}_2 \wedge \texttt{initial} \models_O \bot \qquad \texttt{Recette}_2 \models_O \texttt{substitut}$$

On obtient ainsi des substitutions plus complexes que celles qui sont obtenues par similarité dans la classification d'ingrédients. Par exemple ([Badra *et al.*, 2009a]), le motif ($\texttt{platSalade} \wedge \neg\texttt{pommeDeTerre}, \texttt{vinaigre}, \texttt{jusCitron} \wedge \texttt{sel}$) suggère de remplacer le vinaigre dans une salade sans pomme de terre par du jus de citron et du sel. Bien que les motifs obtenus soient (relativement) fréquents, les substitutions obtenues ne sont pas toutes pertinentes, une validation par l'utilisateur est demandée avant qu'elles soient exploitées par le moteur de RÀPC. Cette validation est proposée de manière opportuniste : elle est demandée lorsque la substitution peut être exploitée pour répondre à l'utilisateur, il peut ainsi juger de sa pertinence en jugeant du résultat obtenu pour l'exemple d'adaptation auquel il est confronté.

4.1.4 TAAABLE 3

Deux nouveautés ont été apportées à TAAABLE 3 [Blansché *et al.*, 2010] : l'adaptation des quantités d'ingrédients et l'adaptation de la préparation textuelle.

Adaptation des quantités d'ingrédients. L'adaptation des quantités est effectuée lorsque les substitutions d'ingrédients ont été déterminées. L'adaptation de la quantité des ingrédients de la recette *Baked apple pancake* pour substituer les pommes par des bananes est présentée dans la figure 4.3. Les pommes sont remplacées par la même masse de bananes, et puisque les bananes sont plus sucrées que les pommes, la quantité de saccharose (sucre en poudre) a été réduite.

Ingredient	Initial Quantity	New Quantity
Salt	=	1 grams
Unsalted butter	=	0 whole (0.0 grams)
Milk	=	1 cup (244.0 grams)
Granulated sugar	100.0 grams	80.0 grams
Egg	=	4 whole (200.0 grams)
Granny smith apple	2 whole (446.0 grams)	0 whole (0.0 grams)
Banana	0 grams (0.0 grams)	446 grams
Brown sugar	=	0 cup (110.0 grams)
Cinnamon	=	2 tsp (5.0 grams)
All-purpose flour	=	1 cup (250.0 grams)

FIGURE 4.3 – Résultat de l'adaptation des quantités d'ingrédients de la recette *Baked apple pancake*. Les quantités d'ingrédients de la recette initiale sont données dans la colonne centrale (*Initial Quantity*), les quantités après adaptation dans la colonne de droite (*New Quantity*). Les pommes sont remplacées par la même masse de bananes et la quantité de saccharose (sucre en poudre) a été réduite pour compenser le fait que les bananes sont naturellement plus sucrées que les pommes. La quantité des autres ingrédients reste inchangée.

Pour obtenir cette adaptation, on exploite des connaissances supplémentaires : les quantités d'ingrédients des recettes de la base fournie par le CCC et les données nutritionnelles de chaque catégorie d'ingrédient. Les quantités de chaque ingrédient utilisé dans une recette, ainsi que les unités dans lesquelles sont exprimées ces quantités, étaient déjà présentes dans WIKITAAABLE. Elles apparaissent comme attribut du lien sémantique entre la page de la recette et la page de la catégorie d'ingrédient (voir figure 4.1). Les données nutritionnelles des ingrédients ont été ajoutées dans leurs pages respectives (voir la figure 4.4). Les correspondances entre unités de volume et masse en grammes et, lorsque l'ingrédient peut être mesuré en pièces (par exemple, une pomme), la masse d'une pièce, sont aussi disponibles sur les pages des ingrédients.

Nous donnons plus de détails sur l'adaptation des quantités d'ingrédients en section 4.2.

Nutritional values

Nutritional value per 100 g (3.5 oz)	
Energy	52 kcal (220 kJ)
Carbohydrates	13.81 g
Sugars	10.39 g
Dietary fiber	2.4 g
Fat	0.17 g
Protein	0.26 g
Water	85.56 g
Vitamin A (equiv.)	3 µg (0%)
Thiamine (Vit. B1)	0.017 mg (1%)
Riboflavin (Vit. B2)	0.026 mg (2%)
Niacin (Vit. B3)	0.091 mg (1%)
Pantothenic acid (Vit. B5)	0.061 mg (1%)
Vitamin B6	0.041 mg (3%)
Folate (Vit. B9)	3 µg (1%)
Vitamin C	4.6 mg (8%)
Calcium	6 mg (1%)
Iron	0.12 mg (1%)
Magnesium	5 mg (1%)
Phosphorus	11 mg (2%)
Potassium	107 mg (2%)
Sodium	1 mg (0%)
Zinc	0.04 mg (0%)

Percentages are relative to US recommendations for adults.
Source: USDA Nutrient database
Corresponding ingredient: APPLES,RAW,WITH SKIN

Nutritional values

Nutritional value per 100 g (3.5 oz)	
Energy	89 kcal (370 kJ)
Carbohydrates	22.84 g
Sugars	12.23 g
Dietary fiber	2.6 g
Fat	0.33 g
Protein	1.09 g
Water	74.91 g
Vitamin A (equiv.)	3 µg (0%)
Thiamine (Vit. B1)	0.031 mg (2%)
Riboflavin (Vit. B2)	0.073 mg (5%)
Niacin (Vit. B3)	0.665 mg (4%)
Pantothenic acid (Vit. B5)	0.334 mg (7%)
Vitamin B6	0.367 mg (28%)
Folate (Vit. B9)	20 µg (5%)
Vitamin C	8.7 mg (15%)
Calcium	5 mg (1%)
Iron	0.26 mg (2%)
Magnesium	27 mg (7%)
Phosphorus	22 mg (3%)
Potassium	358 mg (8%)
Sodium	1 mg (0%)
Zinc	0.15 mg (2%)

Percentages are relative to US recommendations for adults.
Source: USDA Nutrient database
Corresponding ingredient: BANANAS,RAW

FIGURE 4.4 – Données nutritionnelles extraites des pages *Apple* et *Banana* de WIKITAAABLE. Ces données sont issues de la base de données libre de droits de l'*United States Department of Agriculture* (USDA, http://www.ars.usda.gov/).

Adaptation de la préparation textuelle. Dans les versions précédentes de TAAABLE, l'adaptation des recettes ne concernait que la liste des ingrédients, le système indiquait à l'utilisateur quels ingrédients utiliser, mais conservait le texte de la préparation de la recette source, sans tenir compte des substitutions d'ingrédients. En 2010, un module d'adaptation de la préparation a été développé en s'appuyant sur les travaux de [Dufour-Lussier *et al.*, 2010].

L'adaptation de la préparation est effectuée simultanément à l'adaptation des quantités. Elle est aussi effectuée une fois qu'une recette source a été sélectionnée et les substitutions d'ingrédients déterminées. Le résultat de l'adaptation de la préparation de la recette *Baked apple pancake* pour substituer les pommes par des bananes est présenté dans la figure 4.5. Bien que dans cet exemple l'adaptation textuelle se limite au remplacement de la chaîne de caractères « *apple* » par « *banana* », certaines adaptations textuelles sont plus sophistiquées.

Preparation Adaptation

1. Heat oven to 425 degrees. Lightly oil a 10 - inch cast - iron skillet. Toss ~~apples~~ bananas with both sugars and cinnamon in a large bowl. 2. Mix milk and eggs in a blender or food processor; add flour and salt and mix to combine. 3. Melt butter in prepared skillet over medium heat. Add ~~apple~~ banana mixture; cook, stirring often, until sugar melts, about 5 minutes. Remove from heat and pour batter over. 4. Bake until pancake is puffed and golden, 20 to 30 minutes. Serve at once. Yield: 2 to 4 servings.

FIGURE 4.5 – Résultat de l'adaptation de la préparation de la recette *Baked apple pancake*. Pour faire apparaître les modifications, les parties du texte de départ qui ont été supprimées sont rayées, les parties ajoutées sont soulignées.

4.2 Contribution à TAAABLE 3 : adaptation des quantités d'ingrédients

L'adaptation des quantités d'ingrédients met en œuvre l'adaptation par révision. Elle intervient après la remémoration d'une recette source et une première étape d'adaptation de la représentation propositionnelle de cette recette source. On dispose donc d'une recette source et de substitutions (booléennes) d'ingrédients. Chaque substitution remplace un ingrédient `ingIni` de la recette source par un autre ingrédient `ingFin`.

On dispose donc des données suivantes pour effectuer l'adaptation des quantités :

- La formule $\texttt{Source}_\texttt{p}$ en logique propositionnelle représentant la recette source. C'est une conjonction de littéraux correspondant à des types d'ingrédients, un littéral positif correspond à un type d'ingrédient utilisé dans la recette, un littéral négatif correspond à un type d'ingrédient non utilisé dans la recette. On ignore ici les types de plats et les origines. Dans l'exemple on a :

$$\texttt{Source}_\texttt{p} = \texttt{farine} \wedge \texttt{lait} \wedge \texttt{saccharose} \wedge \texttt{œuf} \wedge \texttt{pomme} \wedge \mathit{rienDAutre}$$

  où *rienDAutre* est la conjonction de négations de variables correspondant aux ingrédients non contenus dans la recette.
- La formule $\texttt{Cible}_\texttt{p}$ en logique propositionnelle représentant la requête de l'utilisateur. Il s'agit aussi d'une conjonction de littéraux, un littéral positif correspond à un type d'ingrédient que l'utilisateur veut utiliser dans la recette, un littéral négatif correspond à un type d'ingrédient qu'il veut éviter. Dans l'exemple on a :

$$\texttt{Cible}_\texttt{p} = \texttt{banane} \wedge \neg\texttt{chocolat}$$

- Un ensemble $\{(\texttt{ingIni}_i, \texttt{ingFin}_i)\}_i$ de substitutions d'ingrédients. Dans l'exemple, il n'y a

qu'une substitution $(\texttt{ingIni}_1, \texttt{ingFin}_1)$ avec :

$$\texttt{ingIni}_1 = \texttt{pomme} \qquad\qquad \texttt{ingFin}_1 = \texttt{banane}$$

4.2.1 Formalisme de représentation

On se place dans un formalisme attributs-valeurs simples. Les attributs correspondent aux quantités des différents ingrédients. On restreint la liste des ingrédients considérés à ceux qui sont dans la recette source, dans le cas cible, ou qui apparaissent dans les relations de substitutions. On y inclut aussi les ingrédients les généralisant. Dans l'exemple on considère les types d'ingrédients suivants : `farine`, `produitCéréalier`, `lait`, `produitLaitier`, `œuf`, `saccharose`, `chocolat`, `assaisonnement`, `pomme`, `piridion`, `banane`, `fruit`, `ingrédient`.

Les quantités d'ingrédients données dans les recettes sont exprimées dans différentes unités (gramme, millilitre, cuillerées, pièce, ...). Suivant l'unité utilisée, les attributs sont à valeurs dans $\mathbb{R}_+$ (grammes, millilitre) ou dans $\mathbb{N}$ (cuillerées, pièces). Pour exprimer des contraintes entre les quantités de différents ingrédients, on convertit les quantités en grammes. Pour cela, on introduit si nécessaire, un attribut supplémentaire par ingrédient qui représente sa quantité en grammes. La quantité d'un ingrédient peut donc être représentée par plusieurs attributs, les conversions entre les valeurs de ces attributs sont assurées par les connaissances du domaine (nous y revenons plus loin).

En plus des quantités d'ingrédients, on prend aussi en compte leurs valeurs nutritionnelles. La liste de ces grandeurs nutritionnelles est donnée dans les pages d'ingrédients (voir figure 4.4). Un attribut est introduit pour chaque grandeur nutritionnelle. Les grandeurs qui ont une influence importante sur le goût et l'apparence de la recette finale sont les teneurs en sucres, en corps gras et en eau, nous leur donnons donc plus d'importance dans le calcul de l'adaptation qu'aux autres grandeurs. Les autres grandeurs ont cependant un intérêt diététique et une perspective serait de prendre en compte ces valeurs pour répondre à des demandes de régimes particuliers par les utilisateurs.

Certaines grandeurs nutritionnelles ne peuvent pas être exprimées en grammes (par exemple pour l'énergie contenue dans les ingrédients qui est exprimée en calories), on ne considère donc qu'un seul attribut par grandeur nutritionnelle avec l'unité dans laquelle elle est exprimée (voir figure 4.4).

D'autres grandeurs pourraient être considérées pour l'adaptation des quantités, par exemple l'acidité des ingrédients ou des valeurs caractérisant la puissance des goûts. Mais nous manquons de connaissances pour les calculer.

Les recettes sont donc représentées à l'aide d'attributs $\mathtt{a}_1, \ldots, \mathtt{a}_n$ qui prennent respectivement leurs valeurs dans les ensembles $\mathcal{U}_1, \ldots, \mathcal{U}_n$ où $\mathcal{U}_i = \mathbb{R}_+$ ou $\mathcal{U}_i = \mathbb{N}$. Une recette est donc représentée par une valeur dans $\mathcal{U} = \mathcal{U}_1 \times \ldots \times \mathcal{U}_n$. Dans l'exemple, les attributs reprendront les notations des variables propositionnelles, avec en indice l'unité. Par exemple, l'attribut correspondant à la quantité de pommes en grammes est noté $\texttt{pomme}_\mathtt{g}$, l'attribut correspondant à

la quantité de pommes en unités (pièces) est notée $\texttt{pomme}_\texttt{u}$. Comme nous allons le voir dans la suite, les connaissances culinaires dont nous disposons peuvent s'exprimer par des contraintes linéaires entre les attributs. On est donc dans la situation décrite en section 2.4.2, où pour certaines « distances » d, le calcul de la révision de connaissances suivant l'opérateur $\dot{+}^d$, et donc celui de la $\dot{+}^d$-adaptation, est réductible à un problème de programmation linéaire.

4.2.2 Formulation des connaissances du domaine et du cas source

Les connaissances dont nous disposons pour cette adaptation sont les suivantes :

- Le cas source $\texttt{Source}_\texttt{N}$ qui représente la recette source dans le formalisme décrit dans la section précédente.
- Les conversions d'unité.
- La hiérarchie de types d'ingrédients.
- Les données nutritionnelles des ingrédients.

Les trois dernières connaissances sont des connaissances du domaine, elles sont représentées par la base de connaissances CD.

Représentation de $\texttt{Source}_\texttt{N}$

La recette source est représentée par un ensemble de formules de la forme $(\texttt{a} = v)$ où v est la valeur indiquée dans la recette pour l'ingrédient représenté par l'attribut a. Pour tout ingrédient n'apparaissant pas dans la recette, c'est-à-dire tel que le terme correspondant $\neg p$ est dans $\texttt{Source}_\texttt{p}$, la formule $(p_\texttt{g} = 0)$ est ajoutée à $\texttt{Source}_\texttt{N}$.

La recette *Baked apple pancake* est représentée par la BC suivante :

$$\texttt{Source}_\texttt{N} = \left\{ \begin{array}{l} (\texttt{farine}_\texttt{coupe} = 1), (\texttt{lait}_\texttt{coupe} = 1), (\texttt{œuf}_\texttt{u} = 4), (\texttt{saccharose}_\texttt{coupe} = 1), \\ (\texttt{pomme}_\texttt{u} = 2), (\texttt{banane}_\texttt{g} = 0) \end{array} \right\}$$

Conversion d'unités

Pour tout ingrédient exprimé dans deux unités, avec un attribut $\texttt{ing}_{[\texttt{unité}]}$ qui représente sa quantité dans l'unité [unité] et un attribut $\texttt{ing}_\texttt{g}$ qui représente sa quantité en grammes, on ajoute la formule suivante à CD :

$$(\texttt{ing}_\texttt{g} = \alpha \cdot \texttt{ing}_{[\texttt{unité}]})$$

où α est la masse en grammes correspondant à 1 [unité] de ing.

Remarque 4.2.1. *Ce que nous appelons « conversion » a un sens plus large qu'en physique puisque nous convertissons des valeurs entre unités non homogènes. Par exemple,* 1*coupe de farine pèse* 250*g. Pour les ingrédients mesurés à l'unité (pièce), dans l'exemple, les œufs, les pommes et les bananes, on considère la masse typique d'une unité, ces valeurs sont aussi issues de la base de données USDA (voir figure 4.4). Compte tenu des valeurs disponibles, les qualificatifs*

donnés aux ingrédients sont pris en compte. Par exemple, la recette Baked Apple Pancake *contient des grosses pommes, qui pèsent* $223g$ *selon les données USDA, une petite pomme ne pèse que* $101g$ *selon les mêmes données.*

Les conversions suivantes sont utilisées dans l'exemple :

$(\texttt{pomme}_{\texttt{g}} = 223 \cdot \texttt{pomme}_{\texttt{u}})$	Une (grosse) pomme pèse 223 grammes.
$(\texttt{farine}_{\texttt{g}} = 120 \cdot \texttt{farine}_{\texttt{coupe}})$	Une coupe de farine pèse 120 grammes.
$\vdots$	

Hiérarchie d'ingrédients

La hiérarchie des types d'ingrédients s'exprime par des contraintes linéaires entre attributs d'ingrédients liés par la relation de généralisation. La quantité contenue dans une recette d'un type d'ingrédient général est égale à la somme des quantités des ingrédients particuliers qu'elle utilise.

Ainsi, si un type ingrédient `ingGen` regroupe les types spécifiques d'ingrédients $\texttt{ing1}, \ldots, \texttt{ing}k$, alors on ajoute la formule suivante à `CD` :

$$(\texttt{ingGen}_{\texttt{g}} = \texttt{ing1}_{\texttt{g}} + \ldots + \ldots \texttt{ing}k_{\texttt{g}})$$

Remarque 4.2.2. *Par « types spécifiques » d'ingrédients, on entend types d'ingrédients pour lesquels il n'y a de type plus spécifique dans la liste d'ingrédients retenus. Dans l'exemple, la valeur de* $fruit_g$ *est déterminée par l'égalité suivante :*

$$fruit_g = pomme_g + banane_g$$

Le terme $piridion_g$ *n'apparaît pas dans cette somme, car comme il généralise* $pomme$ *on a* $piridion_g = pomme_g$ *et la masse des pommes serait comptée deux fois dans* $fruit_g$.

Nous avons décidé de prendre les égalités

$$\begin{cases} fruit_g = pomme_g + banane_g \\ piridion_g = pomme_g \end{cases} \quad \textit{plutôt que} \quad \begin{cases} fruit_g = piridion_g + banane_g \\ piridion_g = pomme_g \end{cases}$$

pour éviter les problèmes posés par les hiérarchies multiples. Par exemple, les brocolis sont classés

sous les types `légumeFleur` *et* `chou`[20]*, donc si on écrivait les relations suivantes :*

$$\begin{aligned} \texttt{légume}_g &= \texttt{légumeFleur}_g + \texttt{chou}_g \\ \texttt{légumeFleur}_g &= \texttt{broccoli}_g \\ \texttt{chou}_g &= \texttt{broccoli}_g \end{aligned}$$

la masse de brocolis serait comptée deux fois dans la masse des légumes. On préférera donc écrire $\texttt{légume}_g = \texttt{broccoli}_g$ *à la place de la première égalité.*

Remarque 4.2.3. *Certaines recettes ajoutent un petit problème supplémentaire, les quantités de deux types d'ingrédients* $\texttt{ing}1$ *et* $\texttt{ing}2$ *sont spécifiées alors que* $\texttt{ing}1$ *est plus général que* $\texttt{ing}2$*. Cela provoque une incohérence entre les égalités. Par exemple la recette* Vegetables in crock pot *contient* 1*coupe de brocoli (soit* 91*g) et* 1 *chou (de* 1248*g), or dans la classification de* TAAABLE*, les brocolis sont des choux. On obtient donc les égalités contradictoires suivantes :*

$$\begin{aligned} \texttt{chou}_g &= \texttt{broccoli}_g \\ \texttt{chou}_g &= 1248 \\ \texttt{broccoli}_g &= 91 \end{aligned}$$

Ce problème est contourné en introduisant un nouvel attribut $\texttt{ing}1'_g$ *à qui on donne la valeur spécifiée dans la recette pour* $\texttt{ing}1$ *et qui est traité comme un ingrédient spécifique :*

$$\texttt{ing}1_g = \texttt{ing}1'_g + \texttt{ing}2_g$$

si $\texttt{ing}2$ *est le seul type d'ingrédient généralisé par* $\texttt{ing}1$*. Et le terme* $\texttt{ing}1'_g$ *apparaît aussi dans les égalités déterminant les valeurs* $\texttt{ing}_g$ *pour tout* $\texttt{ing}$ *généralisant* $\texttt{ing}1$*.*

Donc dans le cas de la recette Vegetables in crock pot*, les égalités précédentes sont remplacées par les suivantes :*

$$\begin{aligned} \texttt{chou}_g &= \texttt{chou}'_g + \texttt{broccoli}_g \\ \texttt{chou}'_g &= 1248 \\ \texttt{broccoli}_g &= 91 \end{aligned}$$

qui sont bien cohérentes ($\texttt{chou}_g = 1248 + 91 = 1339$*).*

Données nutritionnelles

On dispose pour chaque type d'ingrédient de ses valeurs nutritionnelles. Par exemple, la teneur en sucre des ingrédients de la recette sont approximativement de 0.1g par gramme de

20. `chou` n'est pas un une sous-classe de `légumeFleur` (les choux pommés ne sont pas des fleurs), et `légumeFleurn`'est pas non plus une sous-classe de `chou`(les artichauts sont des légumes fleur).

pomme, 0.12g par gramme de banane, 1g par gramme de saccharose, 0.05g par gramme de lait, de 0g par gramme de farine et d'œuf. Ces quantités s'additionnent, on obtient alors la quantité de sucre contenue dans la recette :

$$\begin{aligned}\texttt{sucre}_{\texttt{g}} = 0 \cdot \texttt{farine}_{\texttt{g}} + 0.05 \cdot \texttt{lait}_{\texttt{g}} + 0 \cdot \texttt{œuf}_{\texttt{g}} \\ + 1 \cdot \texttt{saccharose}_{\texttt{g}} + 0.1 \cdot \texttt{pomme}_{\texttt{g}} + 0.12 \cdot \texttt{banane}_{\texttt{g}}\end{aligned}$$

Formellement, pour toute grandeur nutritionnelle représentée par l'attribut `vNut`, si on note $\alpha_{\texttt{ing}}^{\texttt{vNut}}$ la valeur de `vNut` contenue dans 1g de `ing`, on ajoute l'égalité suivante à `CD` :

$$\texttt{vNut} = \alpha_{\texttt{ing1}}^{\texttt{vNut}} \cdot \texttt{ing1}_{\texttt{g}} + \ldots + \alpha_{\texttt{ing}k}^{\texttt{vNut}} \cdot \texttt{ing}k_{\texttt{g}}$$

Comme pour les égalités de la hiérarchie d'ingrédients, seuls les ingrédients spécifiques interviennent dans ces calculs ($\texttt{ing1}, \ldots, \texttt{ing}k$ sont les ingrédients spécifiques de la recette).

4.2.3 Formulation du cas cible

La requête exprimée par l'utilisateur est représentée dans $\texttt{Cible}_{\texttt{p}}$ par des contraintes « booléennes » que nous ne savons pas toutes exprimer sous forme de contraintes linéaires dans le formalisme attributs-valeurs simples utilisé ici.

Un terme $\neg\texttt{ing}$ dans $\texttt{Cible}_{\texttt{p}}$ qui correspond à la demande par l'utilisateur de ne pas mettre d'ingrédient de type `ing` dans la recette, peut être exprimé par la contrainte $\texttt{ing}_{\texttt{g}} = 0$.

En revanche, nous ne savons pas exprimer une demande d'ingrédient, représentée par un terme `ing` dans $\texttt{Cible}_{\texttt{p}}$. En effet, la contrainte $\texttt{ing}_{\texttt{g}} > 0$ n'est pas permise (n'est pas dans le langage) car elle délimite un ensemble non fermé de l'espace. Même une contrainte approchée $\texttt{ing}_{\texttt{g}} \geqslant \varepsilon$, avec un « petit » $\varepsilon > 0$ ne donne pas de résultat satisfaisant. Suivant la « distance » de l'exemple 2.4.1 (page 36), la distance minimale de la valeur $\texttt{ing}_{\texttt{g}} = 0$ à l'ensemble délimité par $\texttt{ing}_{\texttt{g}} \geqslant \varepsilon$ est atteint pour $\texttt{ing}_{\texttt{g}} = \varepsilon$. Avec $\varepsilon = 1$, on obtient une crêpe avec 1g de banane et $2 \times 223 - 1 = 445$g de pomme.

Pour obtenir un résultat intéressant, on s'appuie sur les substitutions obtenues par l'étape d'adaptation précédente. Pour chaque substitution ($\texttt{ingIni}, \texttt{ingFin}$), on ajoute la formule suivante à `CD` :

$$\texttt{subs} = \texttt{ingIni}_{\texttt{g}} + \texttt{ingFin}_{\texttt{g}}$$

où `subs` est un nouvel attribut. Et la formule suivante est ajoutée à $\texttt{Cible}_{\texttt{N}}$:

$$\texttt{ingIni}_{\texttt{g}} = 0$$

On suppose que l'on donne plus d'importance aux différences de valeurs de `subs` qu'aux différences de valeurs de $\texttt{ingIni}_{\texttt{g}}$ et $\texttt{ingFin}_{\texttt{g}}$. C'est le cas, par exemple, avec la « distance » de l'exemple 2.4.1 (page 36) et un coefficient de `subs` supérieur à la somme des coefficients de

$\texttt{ingIni}_\texttt{g}$ et $\texttt{ingFin}_\texttt{g}$. Ainsi, si on ne considère aucune autre contrainte, le changement minimal d'une valeur $u \in \mathcal{U}$ qui vérifie $\texttt{ingIni}_\texttt{g} = x$ ($x \in \mathbb{R}$) et $\texttt{ingFin}_\texttt{g} = 0$, à l'ensemble des valeurs $v \in \mathcal{U}$ qui vérifient $\texttt{ingIni}_\texttt{g} = 0$ est atteint pour des valeurs vérifiant $\texttt{ingFin}_\texttt{g} = x$. Le résultat attendu de l'adaptation est obtenu par l'effet de « compensation » de l'adaptation conservatrice (voir la section 3.2.2).

4.2.4 Choix d'une distance et calcul

On utilise la « distance » de l'exemple 2.4.1 (page 36), pour $x, y \in \mathcal{U}$ avec $x = (x_1, \ldots, x_n)$ et $y = (y_1, \ldots, y_n)$:

$$d(x, y) = \sum_{i=1}^{n} w_i |y_i - x_i|$$

Chaque attribut correspond à une composante de $\mathcal{U} = \mathcal{U}_1 \times \ldots \times \mathcal{U}_n$.

Quelques critères guident le choix des coefficients w_i :

- On ne prend en compte que les quantités d'ingrédients en grammes, donc pour tout i tel que $\texttt{a}_i$ correspond à la quantité d'un ingrédient dans une autre unité, $w_i = 0$.
- Si les attributs $\texttt{a}_{i1}, \ldots, \texttt{a}_{ik}$ représentent les quantités en grammes des ingrédients spécifiques généralisés par un ingrédient `ingGen` tel que $\texttt{ingGen}_\texttt{g} = \texttt{a}_{i0}$, alors :

  $$w_{i0} \geqslant w_{i1} + \ldots + w_{ik}$$

  Cette condition permet d'assurer l'effet de « compensation » de l'adaptation conservatrice (voir la section 3.2.2).

En pratique, les coefficients de d sont calculés en fonction de la base de recette, w_i est égal au nombre de recettes qui utilisent un ingrédient correspondant à $\texttt{a}_i$.

4.3 Contributions possibles à des versions futures de TAAABLE

L'application des outils d'adaptation à TAAABLE a soulevé quelques problèmes et quelques compromis ont dû être fait pour obtenir un système fonctionnel. Cependant, ces solutions peuvent être largement améliorées. On distingue dans la suite deux niveaux de perspectives, celles qui concernent l'adaptation des quantités et celles qui concernent l'adaptation en général pour TAAABLE.

Perspectives concernant l'adaptation des quantités. La technique utilisée pour formuler les demandes d'ingrédients par l'utilisateur n'est pas complètement satisfaisante. L'entière substitution d'un ingrédient de la recette source par l'ingrédient demandé peut être excessive. Par exemple pour adapter une tarte aux pommes pour qu'elle contienne des poires, on pourrait ne remplacer que la moitié des pommes par des poires. La proportion de moitié de pommes et moitié de poires résulte d'un choix arbitraire, mais on peut considérer une contrainte moins forte sur

la substitution d'ingrédients pour laisser d'autres critères peser sur le calcul des quantités. Une piste serait de modéliser cette substitution par la « distance » plutôt que par des contraintes. Étant donné une substitution $\texttt{ing1} \rightarrow \texttt{ing2}$, on modifie la « distance » d en « croisant » la comparaison des quantités de ing1 et de ing2 entre le cas source et le cas cible. On suppose pour simplifier que les attributs correspondant aux ingrédients ing1 et ing2 sont respectivement $\texttt{a}_1$ et $\texttt{a}_2$ ($\texttt{ing1}_\texttt{g} = \texttt{a}_1$ et $\texttt{ing2}_\texttt{g} = \texttt{a}_2$). Pour $u, v \in \mathcal{U}$, on défini :

$$d(x, y) = w_1|y_2 - x_1| + w_2|y_1 - x_2| + w_3|y_3 - x_3| + \ldots + w_n|y_n - x_n|$$

Par exemple supposons que $n = 3$ et que le type d'ingrédient ing3 regroupe les types d'ingrédients ing1 et ing2, CD contient donc la contrainte $\texttt{ing3}_\texttt{g} = \texttt{ing1}_\texttt{g} + \texttt{ing2}_\texttt{g}$, et $w_3 > w_1 + w_2$. La distance minimale entre $x = (446, 0, 446)$ et $y = (y_1, y_2, y_3)$ vaut

$$d(x, y) = w_1|y_2 - 446| + w_2|y_1 - 0| + w_3|y_3 - 446|$$

elle est minimale pour $y_2 = 446$, $y_1 = 0$ et $y_3 = 446$. Si on ajoute la contrainte $\texttt{ing2}_\texttt{g} \leqslant \texttt{ing1}_\texttt{g}$ à CD, c'est-à-dire que l'on ne considère que les valeurs y telles que $y_2 \leqslant y_1$ (x vérifie bien cette contrainte), la distance minimale entre x et $y \in \mathrm{Mod}(\texttt{CD})$ est atteinte pour $y = (223, 223, 446)$. On a substitué la moitié de ing1 par ing2, c'est la quantité maximale de ing2 que l'on peut introduire en conservant la quantité de ing3, sous les contraintes imposées par CD.

Remarque 4.3.1. *La fonction d obtenue n'est plus une « distance » puisqu'elle ne vérifie plus l'axiome de séparation, et $\dot{+}^d$ ne vérifie pas le postulat* (G2).

Les règles de substitution acquises par fouille de la base de cas dans TAAABLE 2 ne sont pas exploitées dans TAAABLE 3 car elles ne substituent pas un ingrédient par un autre mais un ensemble d'ingrédients par un autre ensemble d'ingrédients. Pour modéliser ces substitutions plus complexes sur des quantités numériques, des connaissances supplémentaires sont nécessaires, comme les proportions entre les ingrédients introduits. Avec l'exemple pour la substitution présentée en section 4.1.3 (page 121), où le vinaigre est substitué par du jus de citron et du sel, il est important de savoir que la proportion de sel doit être petite par rapport à celle de jus de citron. Il faut donc aussi revoir comment modéliser ces substitutions dans notre formalisme attributs-valeurs simples.

Le choix des coefficients de la « distance » est fixe et est choisi pour donner les meilleurs résultats globalement. Mais l'importance relative des différents critères intervenant dans le calcul de la distance peut varier suivant le contexte (demandes de l'utilisateur ou type de la recette adaptée). Par exemple, lors de l'adaptation d'une recette de confiture, la quantité d'eau contenue dans les ingrédients a peu d'importance par rapport à leurs teneurs en sucre. En revanche, lors de l'adaptation d'une recette de tarte, la quantité d'eau contenue dans les ingrédients est importante : si on n'en tient pas compte, on risque d'obtenir une tarte molle.

Perspectives concernant l'adaptation en général pour TAAABLE. Actuellement, les connaissances utilisées pour l'adaptation sont une classification culinaire des ingrédients, leurs valeurs nutritionnelles et les quantités d'ingrédients dans chaque recette. Pour obtenir des adaptations de recettes plus satisfaisantes, d'autres critères sont à prendre en compte. Par exemple, la connaissance des transformations qui peuvent être effectuées sur les ingrédients (couper, presser, griller, ...), permettrait de substituer des ingrédients par des ingrédients pouvant être transformés de la même manière. La connaissance de propriétés chimiques des ingrédients permettrait de choisir des ingrédients et les actions qui leur sont appliquées pour produire des effets désirés. Cela permettrait, par exemple, de proposer un substitut différent aux blancs d'œufs pour une mousse au chocolat, où du soja soyeux convient parce qu'il reste cru, que pour un gâteau, où une levure convient mieux.

La représentation de ces connaissances demande un formalisme plus expressif, comme une logique de descriptions $\mathcal{ALC}$ (D)[21] ou un fragment. Nos travaux portant sur la révision dans ce formalisme permettraient d'effectuer l'adaptation en prenant directement en compte ces connaissances.

21. $\mathcal{ALC}(D)$ est l'extension d'$\mathcal{ALC}$ par un domaine concret D. Par exemple, D peut être le domaine des intervalles de réels ce qui permettrait d'exprimer des concepts tels que celui des recettes contenant moins de 100g de sucre.

Conclusion

Les travaux de cette thèse s'appuient sur un rapprochement entre le raisonnement effectué pour la révision des connaissances et le raisonnement effectué pour l'adaptation de cas dans le cadre du RÀPC hypothétique. Dans les deux cas, il s'agit de modifier les connaissances *a minima* afin de les rendre cohérentes avec d'autres connaissances. Dans le cadre de la révision, on cherche à modifier au minimum les connaissances dont on dispose initialement pour ne pas les remettre en cause inutilement. Dans le cadre de l'adaptation, en l'absence de connaissances permettant de vérifier une solution candidate à un problème, la modification d'une solution connue risque de la rendre moins bonne. On cherche donc à la modifier le moins possible pour résoudre un nouveau problème.

L'adaptation par la révision. Ce rapprochement entre révision et adaptation motive la définition de l'adaptation par la révision où l'adaptation d'un cas source pour résoudre un cas cible est obtenue en révisant le cas source par le cas cible. L'intérêt de cette approche de l'adaptation est de bénéficier des travaux sur la révision. Alors que le processus d'adaptation est peu formalisé en RÀPC et souvent développé *ad hoc*, la théorie de la révision est bien formalisée et il existe plusieurs représentations de la notion de changement minimal de connaissances qui permettent de définir un opérateur de révision, notamment sous forme de « distance » entre interprétations. Dans le chapitre 3 nous avons étudié le lien entre les postulats de la théorie de la révision et les propriétés attendues de l'adaptation.

Nous défendons l'idée que l'adaptation par la révision donne un cadre formel permettant de représenter un grande partie des formes d'adaptation utilisées pour le RÀPC hypothétique. Pour défendre cette idée, nous avons montré que plusieurs approches de l'adaptation précédemment définies peuvent être exprimées sous forme d'adaptation par la révision. Ces travaux devraient permettre d'exploiter plusieurs formes de connaissances d'adaptation dans un même système de RÀPC. Souvent, les connaissances d'adaptation s'expriment naturellement sous une autre forme qu'une « distance » entre interprétations. Il est donc utile de pouvoir exprimer les connaissances d'adaptations sous une forme naturelle pour les utilisateurs et de les traduire sous forme d'opérateur de révision pour le calcul de l'adaptation. Dans le même ordre d'idées, il serait utile de pouvoir expliquer les résultats de l'adaptation à l'aide des connaissances d'adaptation formulées par les utilisateurs.

L'utilisation de différentes connaissances d'adaptation pose aussi la question du choix entre

les différentes solutions qu'elles offrent et de la manière de les associer pour tirer le meilleur profit de chacune. Un premier essai a été fait en section 3.3.1 (page 97) avec l'association entre adaptation par règles et adaptation conservatrice. Il faudrait étudier plus généralement, comment associer différentes formes d'adaptation par révision et quelle stratégie adopter quand plusieurs approches d'adaptation donnent des solutions différentes.

Dans la discussion de la section 3.2.4, nous avons envisagé d'utiliser pour l'adaptation, une forme de minimalité du changement légèrement différente de celle de la révision et utilisée pour la mise à jour. Cela ne change pas le résultat de l'adaptation d'un cas source ponctuel, ce qui correspond à la situation la plus courante en RÀPC. Mais elle semble mieux correspondre à la signification habituellement donnée à un cas source non ponctuel. Cette intuition est encore peu justifiée et il faudrait comparer les résultats obtenus par l'adaptation de cas sources suivant les deux formes de minimalité du changement pour juger. Il faudrait ensuite déterminer comment les travaux sur la révision dans des formalismes plus expressifs que la logique propositionnelle peuvent être étendus à la mise à jour.

Révision pour l'adaptation. Bien que dans l'article de référence [Alchourrón *et al.*, 1985] la révision soit définie dans un cadre plus large, les travaux sur la révision se limitent souvent à la logique propositionnelle. Or elle est peu utilisée en RÀPC où la représentation des connaissances du domaine et des cas demandent des formalismes plus expressifs. Les formalismes les plus utilisés en RÀPC sont les formalismes attributs-valeurs simples ou des formalismes plus structurés comme les formalismes objet ou des formalismes logiques. Dans le but d'appliquer l'adaptation par révision dans ces formalismes, nous avons étendu, en section 2.3, les travaux sur la révision dans un formalisme logique plus large que la logique propositionnelle et qui couvre notamment les formalismes attributs-valeurs simples. Nous avons ensuite proposé en section 2.4 un opérateur de révision dans un formalisme attributs-valeurs simples dont le calcul est réductible à un problème de programmation linéaire lorsque les connaissances sont représentées par des contraintes linéaires. Nous avons aussi présenté en section 2.5 une version préliminaire d'un opérateur de révision d'ABox exprimées dans la logique de descriptions $\mathcal{ALC}$ et dont le calcul s'appuie sur l'algorithme des tableaux.

Nous avons plusieurs perspectives pour poursuivre ces travaux. Concernant la révision dans le formalisme attributs-valeurs simples présenté en section 2.4.2 nous avons donné une piste pour réduire le calcul de la révision dans le formalisme présenté dans [Schwind, 2010] à un problème de programmation linéaire. Nous avons proposé en section 2.6 quelques pistes pour corriger l'algorithme de révision sur les ABox d'$\mathcal{ALC}$, notamment pour que le résultat de cet algorithme corresponde à la révision suivant un opérateur défini à l'aide d'une « distance » entre interprétations. Un autre développement envisagé serait d'étendre cet algorithme à la révision de TBox en $\mathcal{ALC}$. Il serait ensuite intéressant de l'étendre à d'autres LD plus expressives, notamment avec l'introduction de domaines concrets, ce qui permettrait de faire le lien avec la révision sur les formalismes attributs-valeurs simples.

Combinaison de cas par fusion contrainte. Nous nous sommes aussi intéressés à une généralisation de l'adaptation, la combinaison de cas, qui réutilise plusieurs cas sources pour résoudre un cas cible. Nous proposons en section 3.4 une généralisation de l'adaptation par révision en combinaison de cas à l'aide de la théorie de la fusion contrainte de connaissances. Cette opération qui consiste à déterminer un compromis entre les connaissances provenant de plusieurs sources tout en satisfaisant des contraintes, généralise la révision. En nous appuyant sur les travaux de [Konieczny, 1999], nous montrons que les opérateurs de révision que nous avons défini sur les formalismes attributs-valeurs simples et $\mathcal{ALC}$ peuvent être étendus en opérateurs de fusion contrainte. Comme pour l'adaptation par la révision, nous considérons que la combinaison par fusion contrainte donne un cadre formel permettant de représenter d'autres formes de combinaison déjà utilisées en RàPC. Par exemple, nous montrons en section 3.4.3 que le *Credible Case-Based Inference* [Hüllermeier, 2007] peut être exprimé sous forme de combinaison par fusion.

Nous avons vu, cependant, que les fonctions d'agrégation habituellement utilisées pour la fusion contrainte de connaissances ne donnent pas une forme de compromis intéressante pour le RàPC. Une piste pour obtenir un opérateur de fusion plus approprié à la combinaison de cas a été proposée en section 3.4.5, il reste à tester sur des cas concrets qu'il donne bien les résultats souhaités.

Adaptation par révision dans Taaable. Pour finir nous avons présenté dans le chapitre 4 l'application Taaable. Il s'agit d'un système de démonstration dont le but est de remémorer et adapter des recettes de cuisine. Nous avons appliqué l'adaptation par révision à ce système pour l'adaptation des quantités d'ingrédients. Cela nous a permis de tester cette approche de l'adaptation sur un exemple concret et d'identifier des améliorations possibles.

Plusieurs travaux de cette thèse pourraient être appliqués dans de prochaines versions de Taaable, en particulier l'utilisation de connaissances représentées dans une LD expressive ainsi que l'exploitation de différents types de connaissances d'adaptations. Cela permettrait de tester ces travaux sur des problèmes concrets.

Perspectives

Dans ce mémoire nous avons énoncé plusieurs perspectives que nous avons évoqué dans les différentes parties de la conclusion. À court terme, il faudrait poursuivre les travaux concernant la définition et l'implantation de l'opérateur de révision sur la logique de descriptions $\mathcal{ALC}$. Pour faire la synthèse avec les travaux sur la révision dans les formalismes attributs-valeurs simples, il serait ensuite utile d'étendre cet opérateur de révision pour prendre en compte l'extension de $\mathcal{ALC}$ par des domaines concrets. Le formalisme obtenu permet de représenter les connaissances pour une grande variété de domaines d'applications, pour lesquelles il serait possible d'utiliser l'adaptation par la révision. On pourrait ensuite étendre encore cet opérateur de révision pour couvrir les standards de représentation des connaissances du Web sémantique où de nombreuses

connaissances sont disponibles.

Cela permettrait, à plus long terme, de développer un système général pour l'adaptation en s'appuyant sur l'adaptation par la révision. Ce système devrait fournir un mode d'adaptation générique qui puisse être enrichi par différentes formes de connaissances d'adaptation. Dans ce but, il faudrait poursuivre les travaux d'unification des approches d'adaptation et établir comment incorporer des connaissances d'adaptation données sous différentes formes dans l'adaptation par la révision. Cela soulève différentes question intéressantes d'acquisition et de gestion de ces connaissances d'adaptation, et plus généralement des connaissances exploitées par l'adaptation. Nous pourrions nous intéresser à des techniques d'acquisition opportuniste de connaissances [Cordier *et al.*, 2008], et de gestion collaboratives de connaissances notamment pour les connaissances partagées sur le Web sémantique.

Index

$\mathtt{Cible_p}$, 124
Cn, 7
$\mathcal{C}_{\mathrm{CCBI}}$, 108
CD, voir connaissances du domaine
$\mathcal{U}$, 82
$\mathcal{U}_{\mathtt{pb}}$, 82
$\mathcal{U}_{\mathtt{sol}}$, 82
$\triangle_{IC}(\mathcal{B})$, 70
$\Longrightarrow.$, 48
IC, 70
Min, 9
Mod, 6
$\longrightarrow.$, 44
$\mathtt{Source_p}$, 124
$\overline{A}$, 13
$\leqslant^d_{\cdot}$, 24
$\mathbb{B}$, 7
$\mathtt{CC}^{\mathtt{CD}}_{\triangle}(,,)$ 104
$\mathcal{C}$, 63
$dist(A, B)$, 11
$dist(A, x)$, 11
d_{H}, 22
d^{Σ}, 72
$\equiv$, 7
$\wedge$, 5
$\Sigma^{\min}$, 113
form(.), 21
$\Rightarrow$, 5
(ingIni, ingFin), 124
instances, 63
Ω, 6
$\models$, 6
$\not\models$, 6
$\neg$, 5
$\triangle^{\Sigma^{\min}}$, 113
$\triangle^{d,\Sigma}_{IC}(\mathcal{B})$, 73
$\vee$, 5
$2^{\mathcal{U}}$, 6
$\mathcal{P}$, 26
$(A1) - (A3)$, 21
($\triangle$-0) – ($\triangle$-8), 71
(R1) – (R6), 20
(G1) – (G6), 25
(G4′), 34
$(\overline{\mathrm{G1}}) - (\overline{\mathrm{G6}})$, 29
$\dot{+}$, 19
$\dot{+}_{\mathrm{D}}$, 23
$\mathtt{p}^{-}$, 59
$\mathcal{R}$, 63

ABox, 41
    complète, 44
    disjonction d'ABox, 43
    fermée, 44
    ouverte, 44
adaptation, 80
    chemin d'adaptation, 95
    coût, 82
    conservatrice, 90
    différentielle, 86, 98
    effort, 82
    généralisation/spécialisation, 86
    par copie, 89
    règles, 94
    reformulation, 86, voir adaptation par règles

adhérence, 13
AGM, 19
AGraph, voir graphe assertionnel
$\mathcal{ALC}$, 40
    inférences, 42
    sémantique, 42
    syntaxe, 40
assertion, voir formule assertionnelle
assertional box, voir ABox
Assignation fidèle, 20
attribut, 35

base de cas, 79
base de connaissances, 5
BC, voir base de connaissances
Booléens, 7

cas, 79
    cas cible, 79
    cas source, 79
    ponctuel, 82
CCBI, 108
chemin de similarité, 94
combinaison de cas, 104
    CCBI, 108
    COMPOSER, 111
    DÉJÀ VU, 111
    DZCBR, 111
    $\triangle$-combinaison de cas, 104
    IDIOM, 111
concept, 40
concept atomique, 40
conflit, 44
Connaissances, 5
    dynamiques, 81
    statiques, 81
connaissances d'adaptation, 81
connaissances du domaine, 81
contraction de connaissances, 18
contraintes d'intégrité, 70

débogage de base de connaissances, 19
DBC voir disjonction de BC 43
disjonction de BC, 43
« distance », 11
distance de Hamming, 22
DL-Lite, 59
domaine, 35
DZCBR, 111

entraîne, 7
équivalent, 7

FNN, voir forme normale négative
forme normale négative, 43
formule, 5
formule assertionnelle, 41
formule terminologique, 41
fusion contrainte de connaissances, 17, 69
    postulats, 71
fusion de connaissances, 16

graphe assertionnel, 48

incohérent, 7
inférence crédible à partir de cas, 108, voir CCBI
instance, 41
interprétation, 6

logique, 5
logique propositionnelle
    sémantique, 7
    syntaxe, 5
logiques de descriptions
    expressives, 40

méthode des tableaux, 43
MIPS, 58
mise à jour de connaissances, 18
modèle, 6
monde, 6
multi-ensemble, 70

Ontologie, 19

opérateur de révision, 19

postulats de révision, 20
pré-ordre, 8
    total, 8
profil de similarité, 108

réflexivité, 8
révision
    de base de connaissances, 33
révision de connaissances, 15
révision de Dalal, 23
rôle, 41
rôle inverse, 59
RÀPC, 79
    RÀPC heuristique, 80
    RÀPC hypothétique, 80
RÀPC décentralisé, voir DzCBR
remémoration, 79
remémoration guidée par l'adaptation, 82
représentation des connaissances
    attributs-valeurs simples, 83
$\dot{+}^d$, 24

TBox, 41
terminological box, voir TBox
théorie, 7
transitivité, 8

vérifier (une formule), 6

Bibliographie

[Aamodt et Plaza, 1994] Aamodt, A. et Plaza, E. (1994). Case-based Reasoning : Foundational Issues, Methodological Variations, and System Approaches. *AI Communications*, 7(1):39–59.

[Alchourrón *et al.*, 1985] Alchourrón, C. E., Gärdenfors, P. et Makinson, D. (1985). On the logic of theory change : partial meet functions for contraction and revision. *Journal of Symbolic Logic*, 50:510–530.

[Baader *et al.*, 2003] Baader, F., Calvanese, D., McGuinness, D. L., Nardi, D. et Patel-Schneider, P. F., éditeurs (2003). *The Description Logic Handbook : Theory, Implementation, and Applications*. Cambridge University Press.

[Baader et Hanschke, 1991] Baader, F. et Hanschke, P. (1991). A scheme for integrating concrete domains into concept languages. *In IJCAI*, pages 452–457.

[Badra, 2009] Badra, F. (2009). *Extraction de connaissances d'adaptation en raisonnement à partir de cas*. Thèse, Université Henri Poincaré - Nancy I.

[Badra *et al.*, 2008] Badra, F., Bendaoud, R., Bentebitel, R., Champin, P.-A., Cojan, J., Cordier, A., Després, S., Jean-Daubias, S., Lieber, J., Meilender, T., Mille, A., Nauer, E., Napoli, A. et Toussaint, Y. (2008). Taaable : Text Mining, Ontology Engineering, and Hierarchical Classification for Textual Case-Based Cooking. *In Workshop Proceedings of the 9th Conference on Case-Based Reasoning (ECCBR-08)*, pages 219–228. Tharax Verlag.

[Badra *et al.*, 2009a] Badra, F., Cojan, J., Cordier, A., Lieber, J., Meilender, T., Mille, A., Molli, P., Nauer, E., Napoli, A., Skaf-Molli, H. et Toussaint, Y. (2009a). Knowledge Acquisition and Discovery for the Textual Case-Based Cooking system WikiTaaable. *In Workshops of the 8th International Conference on Case-Based Reasoning (ICCBR-09)*.

[Badra *et al.*, 2009b] Badra, F., Cordier, A. et Lieber, J. (2009b). Opportunistic Adaptation Knowledge Discovery. *In Case-Based Reasoning Research and Development (ICCBR 2009)*. 60–74.

[Bergmann et Wilke, 1996] Bergmann, R. et Wilke, W. (1996). On the Role of Abstraction in Case-Based Reasoning. *In* Smith, I. et Faltings, B., éditeurs : *Advances in Case-Based Reasoning – Third European Workshop, EWCBR'96*, LNAI 1168, pages 28–43. Springer Verlag, Berlin.

[Berners-Lee *et al.*, 2001] Berners-Lee, T., Hendler, J. et Lassila, O. (2001). The Semantic Web. *Scientific American.*

[Blansché *et al.*, 2010] Blansché, A., Cojan, J., Dufour-Lussier, V., Lieber, J., Molli, P., Nauer, E., Skaf-Molli, H. et Toussaint, Y. (2010). Taaable 3 : Adaptation of Ingredient Quantities and of Textual Preparations. *In Workshops of the 18th International Conference on Case-Based Reasoning (ICCBR-10).*

[Branting et Aha, 1995] Branting, L. K. et Aha, D. W. (1995). Stratified Case-Based Reasoning : Reusing Hierarchical Problem Solving Episodes. *In Proceedings of the 14th International Joint Conference on Artificial Intelligence (IJCAI'95), Montréal*, volume 1, pages 384–390.

[Chou et Winslett, 1994] Chou, S.-c. T. et Winslett, M. (1994). A model-based belief revision system. *J. Autom. Reasoning*, 12(2):157–208.

[Cojan et Lieber, 2008a] Cojan, J. et Lieber, J. (2008a). Conservative adaptation in metric spaces. *In Advances in Case-Based Reasoning, 9th European Conference, ECCBR 2008, Trier, Germany. Proceedings*, pages 135–149.

[Cojan et Lieber, 2008b] Cojan, J. et Lieber, J. (2008b). Formalisation de l'adaptation conservatrice dans les espaces métriques. *In 16ème atelier RàPC*, pages 92–107.

[Cojan et Lieber, 2009a] Cojan, J. et Lieber, J. (2009a). Belief Merging-based Case Combination. *In* David C. Wilson et Lorraine McGinty, éditeurs : *8th International Conference on Case-Based Reasoning - ICCBR 2009 Case-Based Reasoning Research and Development*, volume 5650 de *Lecture Notes in Computer Science*, pages 105–119, Seattle United States. Springer Berlin.

[Cojan et Lieber, 2009b] Cojan, J. et Lieber, J. (2009b). Une approche de l'adaptation en raisonnement à partir de cas fondée sur l'optimisation sous contraintes. *In* Cholvy, L. et Konieczny, S., éditeurs : *Journées d'Intelligence Artificielle Fondamentale*, Marseille France.

[Cojan et Lieber, 2010a] Cojan, J. et Lieber, J. (2010a). Adapter des cas en utilisant un opérateur de révision ou des règles. *In* Laurence Cholvy, S. K., éditeur : *Journée Intelligence Artificielle Fondamentale*, Strasbourg France.

[Cojan et Lieber, 2010b] Cojan, J. et Lieber, J. (2010b). Un algorithme d'adaptation avec des cas exprimés dans la logique de descriptions $\mathcal{ALC}$. *In 18ème atelier RàPC.*

[Cojan et Lieber, 2010c] Cojan, J. et Lieber, J. (2010c). An Algorithm for Adapting Cases Represented in an Expressive Description Logic. *In* Stefania Montani et Isabelle Bichindaritz, éditeurs : *9th International Conference on Case-Based Reasoning - ICCBR 2010 Case-Based Reasoning Research and Development*, Lecture Notes in Computer Science, Alessandria Italy. Springer Berlin.

[Cojan et Lieber, 2011a] Cojan, J. et Lieber, J. (2011a). Adaptation par révision et adaptation différentielle : comparaison de deux approches de l'adaptation. *In 19ème atelier Français de Raisonnement à Partir de Cas*, Chambéry France. Fadi Badra and Amélie Cordier.

[Cojan et Lieber, 2011b] Cojan, J. et Lieber, J. (2011b). An Algorithm for Adapting Cases Represented in $\mathcal{ALC}$. *In IJCAI 2011 (to be published)*. AAAI.

[Condotta *et al.*, 2010] Condotta, J.-F., Kaci, S., Marquis, P. et Schwind, N. (2010). Majority merging : from boolean spaces to affine spaces. *In* Coelho, H., Studer, R. et Wooldridge, M., éditeurs : *ECAI*, volume 215 de *Frontiers in Artificial Intelligence and Applications*, pages 627–632. IOS Press.

[Cordier *et al.*, 2008] Cordier, A., Fuchs, B., Lana de Carvalho, L., Lieber, J. et Mille, A. (2008). Opportunistic Acquisition of Adaptation Knowledge and Cases – The IakA Approach. *In Advances in Case-Based Reasoning, 9th European Conference, ECCBR-2008, Trier, Germany. Proceedings*, Lecture Notes in Artificial Intelligence 5239, pages 150–164. Springer.

[Cordier *et al.*, 2009] Cordier, A., Lieber, J., Molli, P., Nauer, E., Skaf-Molli, H. et Toussaint, Y. (2009). Wikitaaable : A semantic wiki as a blackboard for a textual case-based reasoning system. *In SemWiki 2009 – 4th Semantic Wiki Workshop*, Heraklion, Greece.

[Coulon *et al.*, 1990] Coulon, D., Boivieux, J.-F., Bourrelly, L., Bruneau, L., Chouraqui, E., David, J.-M., Lu, C. R., Py, M., Savelli, J., Séroussi, B. et Vrain, C. (1990). Le raisonnement par analogie en intelligence artificielle. *In* B. Bouchon-Meunier, éditeur : *Actes des 3èmes journées nationales du PRC-GDR Intelligence Artificielle*, pages 45–88.

[Craw *et al.*, 2006] Craw, S., Wiratunga, N. et Rowe, R. C. (2006). Learning adaptation knowledge to improve case-based reasoning. *Artificial Intelligence*, 170(16-17):1175–1192.

[Dalal, 1988] Dalal, M. (1988). Investigations into a theory of knowledge base revision : Preliminary report. *In AAAI*, pages 475–479.

[Dantzig, 1963] Dantzig, G. B. (1963). *Linear Programming and Extensions*. Princeton University Press, Princeton, New Jersey.

[d'Aquin *et al.*, 2007] d'Aquin, M., Badra, F., Lafrogne, S., Lieber, J., Napoli, A. et Szathmary, L. (2007). Case Base Mining for Adaptation Knowledge Acquisition. *In* Veloso, M. M., éditeur : *Proceedings of the 20th International Joint Conference on Artificial Intelligence (IJCAI'07)*, pages 750–755. Morgan Kaufmann, Inc.

[d'Aquin *et al.*, 2005] d'Aquin, M., Lieber, J. et Napoli, A. (2005). Decentralized case-based reasoning for the semantic web. *In International Semantic Web Conference*, pages 142–155.

[d'Aquin *et al.*, 2006] d'Aquin, M., Lieber, J. et Napoli, A. (2006). Adaptation Knowledge Acquisition : a Case Study for Case-Based Decision Support in Oncology. *Computational Intelligence (an International Journal)*, 22(3/4):161–176.

[Dixon et Wobcke, 1993] Dixon, S. et Wobcke, W. (1993). The implementation of a first-order logic agm belief revision system. *In ICTAI*, pages 40–47.

[Dufour-Lussier *et al.*, 2010] Dufour-Lussier, V., Lieber, J., Nauer, E. et Toussaint, Y. (2010). Text adaptation using formal concept analysis. *In* Bichindaritz, I. et Montani, S.,

éditeurs : *Case-Based Reasoning Research and Development*, volume 6176 de *Lecture Notes in Artificial Intelligence*, pages 96–110, Alessandria, Italie. Springer-Verlag. The original publication is available at www.springerlink.com.

[Flouris *et al.*, 2008] Flouris, G., Manakanatas, D., Kondylakis, H., Plexousakis, D. et Antoniou, G. (2008). Ontology change : classification and survey. *Knowledge Eng. Review*, 23(2):117–152.

[Flouris *et al.*, 2006] Flouris, G., Plexousakis, D. et Antoniou, G. (2006). On generalizing the agm postulates. *In* Penserini, L., Peppas, P. et Perini, A., éditeurs : *STAIRS*, volume 142 de *Frontiers in Artificial Intelligence and Applications*, pages 132–143. IOS Press.

[Fuchs *et al.*, 2000] Fuchs, B., Lieber, J., Mille, A. et Napoli, A. (2000). An Algorithm for Adaptation in Case-Based Reasoning. *In Proceedings of the 14th European Conference on Artificial Intelligence (ECAI-2000), Berlin, Germany*, pages 45–49.

[Fuchs *et al.*, 2006] Fuchs, B., Lieber, J., Mille, A. et Napoli, A. (2006). A general strategy for adaptation in Case-Based Reasoning. Rapport technique RR-LIRIS-2006-016, LIRIS UMR 5205 CNRS/INSA de Lyon/Université Claude Bernard Lyon 1/Université Lumière Lyon 2/École Centrale de Lyon.

[Fuhrmann, 1991] Fuhrmann, A. (1991). Theory contraction through base contraction. *Journal of Philosophical Logic*, 20:175–203. 10.1007/BF00284974.

[G. Flouris, 2005] G. Flouris, D. Plexousakis, G. A. (2005). On Applying the AGM theory to DLs and OWL. *In 4th International Semantic Web Conference (ISWC-05)*, pages 216–231. LNCS 3729 Springer.

[Gebhardt *et al.*, 1997] Gebhardt, F., Voss, A., Gräther, W. et Schmidt-Belz, B. (1997). *Reasoning with complex cases.* Kluwer, Boston.

[Gärdenfors, 1992] Gärdenfors, P., éditeur (1992). *Belief Revision.* Cambridge University Press.

[Haase *et al.*, 2005] Haase, P., van Harmelen, F., Huang, Z., Stuckenschmidt, H. et Sure, Y. (2005). A framework for handling inconsistency in changing ontologies. *In 4th International Semantic Web Conference*, volume 3729 de *Lecture Notes in Computer Science*, pages 353–367. Springer.

[Halaschek-Wiener *et al.*, 2006] Halaschek-Wiener, C., Katz, Y. et Parsia, B. (2006). Belief base revision for expressive description logics. *In* Grau, B. C., Hitzler, P., Shankey, C. et Wallace, E., éditeurs : *OWLED*06 Workshop on OWL : Experiences and Directions*, volume 216. CEUR-WS.org.

[Hanney et Keane, 1996] Hanney, K. et Keane, M. T. (1996). Learning Adaptation Rules From a Case-Base. *In* Smith, I. et Faltings, B., éditeurs : *Advances in Case-Based Reasoning – Third European Workshop, EWCBR'96*, LNAI 1168, pages 179–192. Springer Verlag, Berlin.

[Hansson, 1991] Hansson, S. O. (1991). *Belief Base Dynamics.* Thèse de doctorat, Uppsala University.

[Horrocks, 1997] Horrocks, I. (1997). *Optimising Tableaux Decision Procedures for Description Logics*. Thèse de doctorat, University of Manchester.

[Hüllermeier, 2007] Hüllermeier, E. (2007). Credible Case-Based Inference Using Similarity Profiles. *IEEE Transaction on Knowledge and Data Engineering*, 19(6):847–858.

[Kalyanpur, 2006] Kalyanpur, A. (2006). *Debugging and Repair of OWL Ontologies*. Thèse de doctorat, University of Maryland College Park.

[Kalyanpur *et al.*, 2007] Kalyanpur, A., Parsia, B., Horridge, M. et Sirin, E. (2007). Finding all justifications of owl dl entailments. *In* Aberer, K., Choi, K.-S., Noy, N. F., Allemang, D., Lee, K.-I., Nixon, L. J. B., Golbeck, J., Mika, P., Maynard, D., Mizoguchi, R., Schreiber, G. et Cudré-Mauroux, P., éditeurs : *ISWC/ASWC*, volume 4825 de *Lecture Notes in Computer Science*, pages 267–280. Springer.

[Kalyanpur *et al.*, 2006] Kalyanpur, A., Parsia, B., Sirin, E. et Grau, B. C. (2006). Repairing unsatisfiable concepts in owl ontologies. *In* Sure, Y. et Domingue, J., éditeurs : *ESWC*, volume 4011 de *Lecture Notes in Computer Science*, pages 170–184. Springer.

[Karmarkar, 1984] Karmarkar, N. (1984). A new polynomial-time algorithm for linear programming. *Combinatorica*, 4(4):373–396.

[Katsuno et Mendelzon, 1991a] Katsuno, H. et Mendelzon, A. (1991a). On the Difference Between Updating a Knowledge Base and Revising It. *In* Allen, J. F., Fikes, R. et Sandewall, E., éditeurs : *KR'91 : Principles of Knowledge Representation and Reasoning*, pages 387–394. Morgan Kaufmann, San Mateo, California.

[Katsuno et Mendelzon, 1991b] Katsuno, H. et Mendelzon, A. (1991b). Propositional knowledge base revision and minimal change. *Artificial Intelligence*, 52(3):263–294.

[Kolodner, 1993] Kolodner, J. (1993). *Case-Based Reasoning*. Morgan Kaufmann, Inc.

[Konieczny, 1999] Konieczny, S. (1999). *Sur la Logique du Changement : Révision et Fusion de Bases de Connaissances*. Thèse de doctorat, Université des Sciences et Technologies de Lille, Villeneuve d'Ascq.

[Konieczny *et al.*, 2004] Konieczny, S., Lang, J. et Marquis, P. (2004). DA^2 merging operators. *Artificial Intelligence*, 157(1-2):49–79.

[Lieber, 2002] Lieber, J. (2002). Strong, Fuzzy and Smooth Hierarchical Classification for Case-Based Problem Solving. *In* van Harmelen, F., éditeur : *Proceedings of the 15th European Conference on Artificial Intelligence (ECAI-02), Lyon, France*, pages 81–85. IOS Press, Amsterdam.

[Lieber, 2007] Lieber, J. (2007). Application of the Revision Theory to Adaptation in Case-Based Reasoning : the Conservative Adaptation. *In Proceedings of the 7th International Conference on Case-Based Reasoning (ICCBR-07)*, Lecture Notes in Artificial Intelligence 4626, pages 239–253. Springer, Belfast.

[Melis *et al.*, 1998] Melis, E., Lieber, J. et Napoli, A. (1998). Reformulation in Case-Based Reasoning. *In* Smyth, B. et Cunningham, P., éditeurs : *Fourth European Workshop on Case-Based Reasoning, EWCBR-98*, Lecture Notes in Artificial Intelligence 1488, pages 172–183. Springer.

[Meyer *et al.*, 2005] Meyer, T., Lee, K. et Booth, R. (2005). Knowledge integration for description logics. *In Proceedings of the AAAI'05*, pages 645–650.

[Nebel, 1992] Nebel, B. (1992). *Syntax-Based Approaches to Belief Revision*, chapitre 3, pages 52–88. Cambridge University Press.

[Orwell, 1945] Orwell, G. (1945). *Animal Farm : A Fairy Story*. Secker and Warburg (London).

[Pellet, 2011] Pellet (2011). Clark & Parsia. dernière consultation : juin 2011. http://clarkparsia.com/pellet/.

[Purvis et Pu, 1996] Purvis, L. et Pu, P. (1996). An Approach to Case Combination. *In* Voss, A., éditeur : *Proc. of the ECAI'96 Workshop : Adaptation in Case-Based Reasoning*, pages 43–46.

[Qi et Du, 2009] Qi, G. et Du, J. (2009). Model-based revision operators for terminologies in description logics. *In* Boutilier, C., éditeur : *IJCAI*, pages 891–897.

[Qi *et al.*, 2008] Qi, G., Haase, P., Huang, Z. et Pan, J. Z. (2008). A kernel revision operator for terminologies. *In* Baader, F., Lutz, C. et Motik, B., éditeurs : *Description Logics*, volume 353 de *CEUR Workshop Proceedings*. CEUR-WS.org.

[Qi *et al.*, 2006] Qi, G., Liu, W. et Bell, D. A. (2006). Knowledge base revision in description logics. *In* Fisher, M., van der Hoek, W., Konev, B. et Lisitsa, A., éditeurs : *JELIA*, volume 4160 de *Lecture Notes in Computer Science*, pages 386–398. Springer.

[Reiter, 1987] Reiter, R. (1987). A theory of diagnosis from first principles. *Artif. Intell.*, 32(1):57–95.

[Richter, 1998] Richter, M. M. (1998). Introduction. *In* Lenz, M., Bartsch-Spörl, B., Burkhard, H.-D. et Wess, S., éditeurs : *Case-Based Reasoning Technologies. From Foundations to Applications*, Lecture Notes in Artificial Intelligence 1400, chapitre 1, pages 1–15. Springer.

[Riesbeck et Schank, 1989] Riesbeck, C. K. et Schank, R. C. (1989). *Inside Case-Based Reasoning*. Lawrence Erlbaum Associates, Inc., Hillsdale, New Jersey.

[Robert, 2007] Robert, D. L., éditeur (2007). *Le Petit Robert*. Dictionnaires Le Robert.

[Schlobach et Cornet, 2003] Schlobach, S. et Cornet, R. (2003). Non-standard reasoning services for the debugging of description logic terminologies. *In* Gottlob, G. et Walsh, T., éditeurs : *IJCAI*, pages 355–362. Morgan Kaufmann.

[Schlobach *et al.*, 2007] Schlobach, S., Huang, Z., Cornet, R. et van Harmelen, F. (2007). Debugging incoherent terminologies. *J. Autom. Reasoning*, 39(3):317–349.

[Schwind, 2010] Schwind, N. (2010). *Fusion de réseaux de contraintes qualitatives*. Thèse de Doctorat d'Université, Université d'Artois.

[Sirin *et al.*, 2007] Sirin, E., Parsia, B., Grau, B. C., Kalyanpur, A. et Katz, Y. (2007). Pellet : A practical owl-dl reasoner. *J. Web Sem.*, 5(2):51–53.

[Smith *et al.*, 1995] Smith, I., Lottaz, C. et Faltings, B. (1995). Spatial composition using cases : IDIOM. *In* Veloso, M. et Aamodt, A., éditeurs : *Case-Based Reasoning Research and Development – ICCBR-95 Proceedings, Sesimbra, Portugal*, LNAI 1010, pages 88–97. Springer Verlag, Berlin.

[SMW, 2011] SMW (2011). Semantic MediaWiki. dernière consultation : juin 2011. http://semantic-mediawiki.org/.

[Smyth et Keane, 1998] Smyth, B. et Keane, M. T. (1998). Adaptation-guided retrieval : Questioning the similarity assumption in reasoning. *Artif. Intell.*, 102(2):249–293.

[Smyth *et al.*, 2001] Smyth, B., Keane, M. T. et Cunningham, P. (2001). Hierarchical case-based reasoning integrating case-based and decompositional problem-solving techniques for plant-control software design. *IEEE Trans. Knowl. Data Eng.*, 13(5):793–812.

[Völkel *et al.*, 2007] Völkel, M., Krötzsch, M., Vrandecic, D., Haller, H. et Studer, R. (2007). Semantic Wikipedia. *Journal of Web Semantics*, 5(4).

[W3C OWL Working Group, 2009] W3C OWL Working Group (2009). OWL 2 Web Ontology Language Document Overview. W3C Recommendation. http://www.w3.org/TR/owl2-overview/.

[Wang *et al.*, 2009] Wang, K., Wang, Z., Topor, R. W., Pan, J. Z. et Antoniou, G. (2009). Concept and Role Forgetting in $\mathcal{ALC}$ Ontologies. *In* Bernstein, A., Karger, D. R., Heath, T., Feigenbaum, L., Maynard, D., Motta, E. et Thirunarayan, K., éditeurs : *International Semantic Web Conference*, volume 5823 de *Lecture Notes in Computer Science*, pages 666–681. Springer.

[Wang *et al.*, 2008] Wang, Z., Wang, K., Topor, R. W. et Pan, J. Z. (2008). Forgetting concepts in dl-lite. *In* Bechhofer, S., Hauswirth, M., Hoffmann, J. et Koubarakis, M., éditeurs : *ESWC*, volume 5021 de *Lecture Notes in Computer Science*, pages 245–257. Springer.

[Weber *et al.*, 2005] Weber, R. O., Ashley, K. D. et Brüninghaus, S. (2005). Textual case-based reasoning. *Knowledge Eng. Review*, 20(3):255–260.

[Williams, 1997] Williams, M.-A. (1997). Anytime belief revision. *In Proceedings of the 15th international joint conference on Artifical intelligence - Volume 1*, pages 74–79, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

[Zadeh, 1965] Zadeh, L. A. (1965). Fuzzy Sets. *Information and Control*, 8:338–353.